Ramaiah Institute of Technology, Bangalore-560 054

(Autonomous Institute, Affiliated to VTU, Belgaum)

Department of Electronics and Instrumentation Engineering

LAB MANUAL

EMBEDDED CONTROLLERS LAB

Sub code: EIL47

Prepared by: Elavaar Kuzhali.S.

Reviewed by: M.K.Pushpa

M.Jyothirmayi

Course Contents and Lecture Schedule

Laboratory Classes

Session No	Topics
1.	Introduction: i) Swap two numbers without using an intermediate register ii) To find the factorial of a given number iii) Convert word of little endian format to big endian format.
2.	Generate 12 bit Hamming code from a given 8 bit code
3.	i) Move a string from given memory location to another location ii)To Add N numbers of data stored consecutively in memory location iii) Translate the given C code to assembly. for (i=0;i<8;i++){a[i]=b[7-i];}
4.	i) Move a block of data from memory location to another location using LOAD multiple and STORE multiple instructions. ii) Exchange a block of data between memory locations.
5.	i) Arrange a given set of data in ascending order ii) Arrange a given set of data in descending order.
6.	i) Implement subroutine nesting using stack ii) To implement ARM –THUMB interworking to find the smallest. iii) To handle SWI instruction in the program
7.	To familiarize I/O ports of LPC 2148 --- on/off control of LEDs using switches
8.	To display a given string using the LCD display interface
9.	Interface key pad and to display the key pressed on LCD
10.	Waveform generation using the internal DAC of LPC 2148.
11.	To convert a given analog voltage to digital using ADC of LPC 2148.
12.	Using timers to generate a specified delay
13.	Using timer/counter/capture module of LPC 2148 to count the number of pulses and display on LCD.
14.	Use of UART of LPC 2148 for transmitting and receiving data

Course Outcomes:

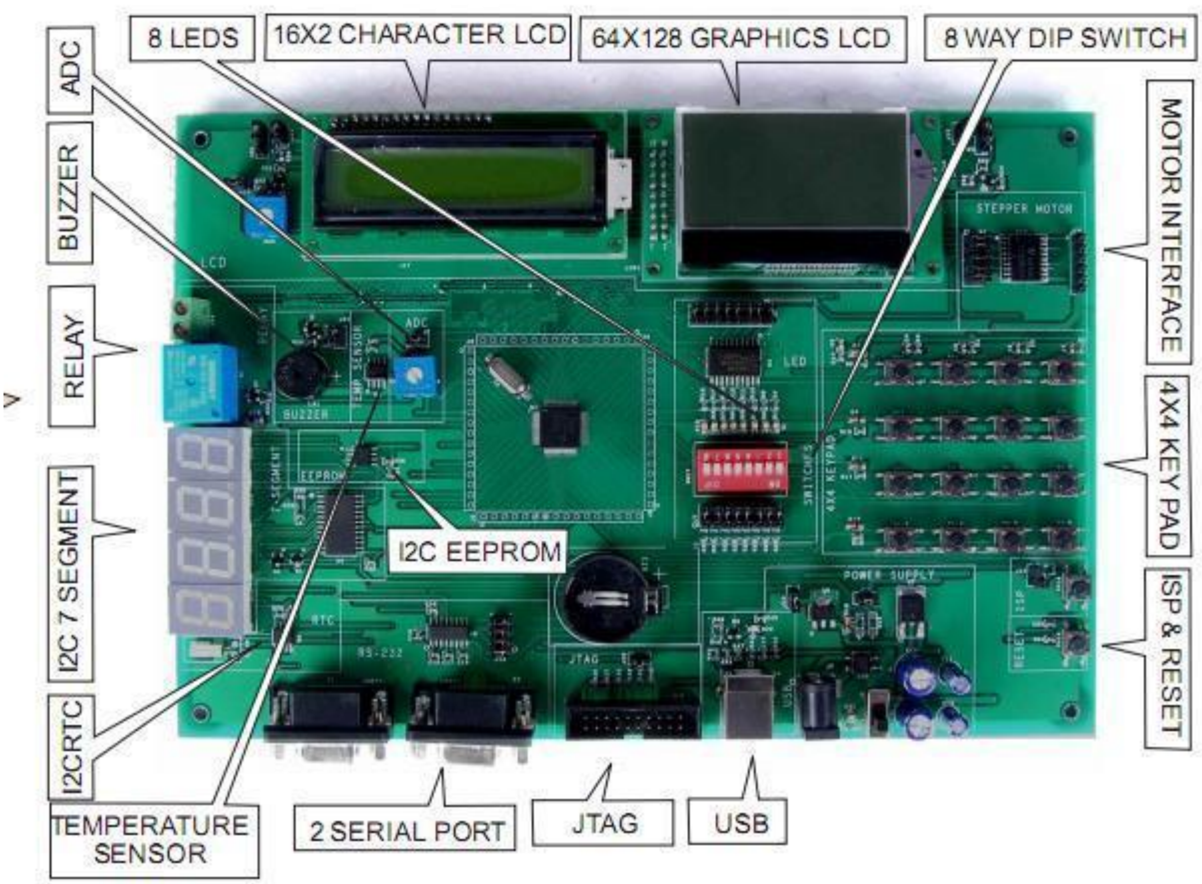
The students should be able to:

1. Design, implement, and debug ARM/THUMB assembly level programs using Keil software. **(PO-1,2,3,4,5,9,10,12, PSO-1,3)**
2. Design, develop, analyze and debug embedded C programs to interact with various built in peripherals of ARM7 LPC2418 kit. **(PO-1,2,3,4,5,9,10,12, PSO-1,3)**
3. Write programs to handle exceptions and interrupts in ARM processor. **(PO-1,2,3,4,5,9,10,12, PSO-1,3)**

Course Outcomes	Program Outcomes(POs)													PROGRAM SPECIFIC OUTCOMES (PSOs):		
	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12		PSO1	PSO2	PSO3
1.	3	3	3	3	3				2	3		3		3		3
2.	3	3	3	3	3				2	3		3		3		3
3.	2	3	3	3	3				2	3		3		3		3

Procedure for using KEIL Software

1. Board Layout

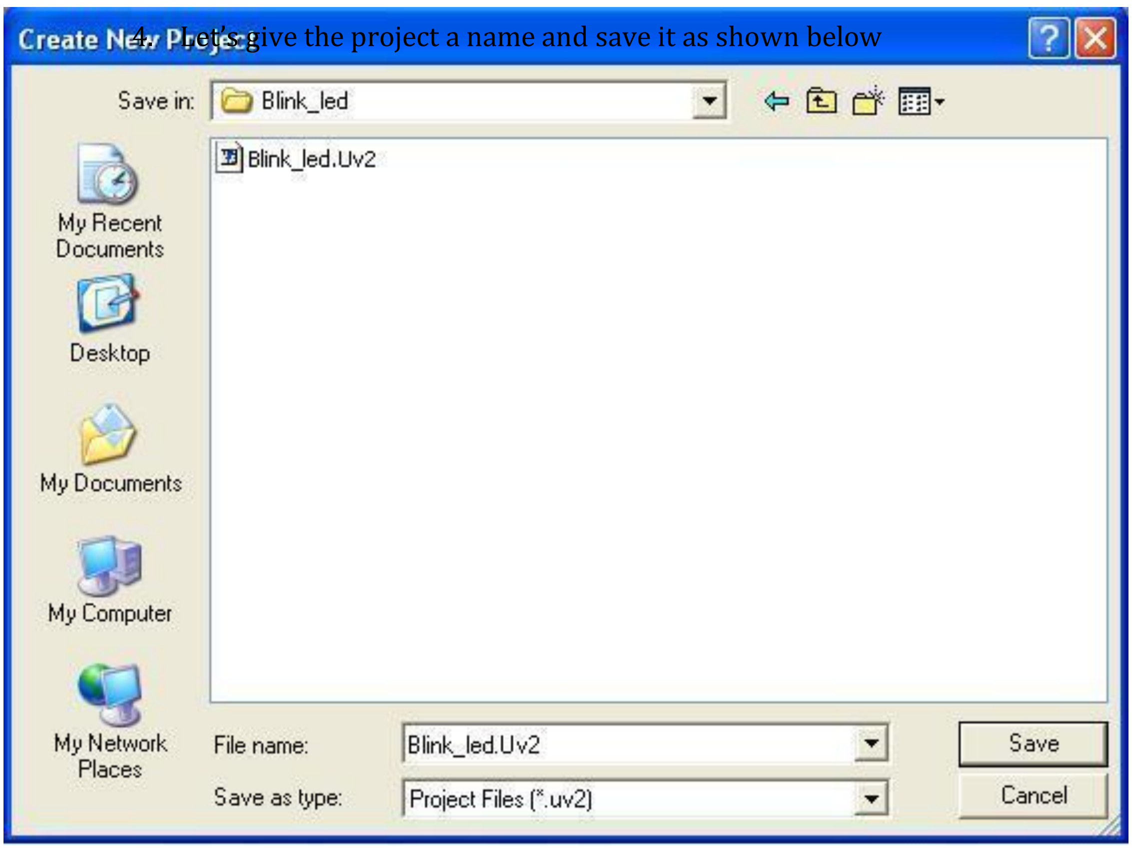


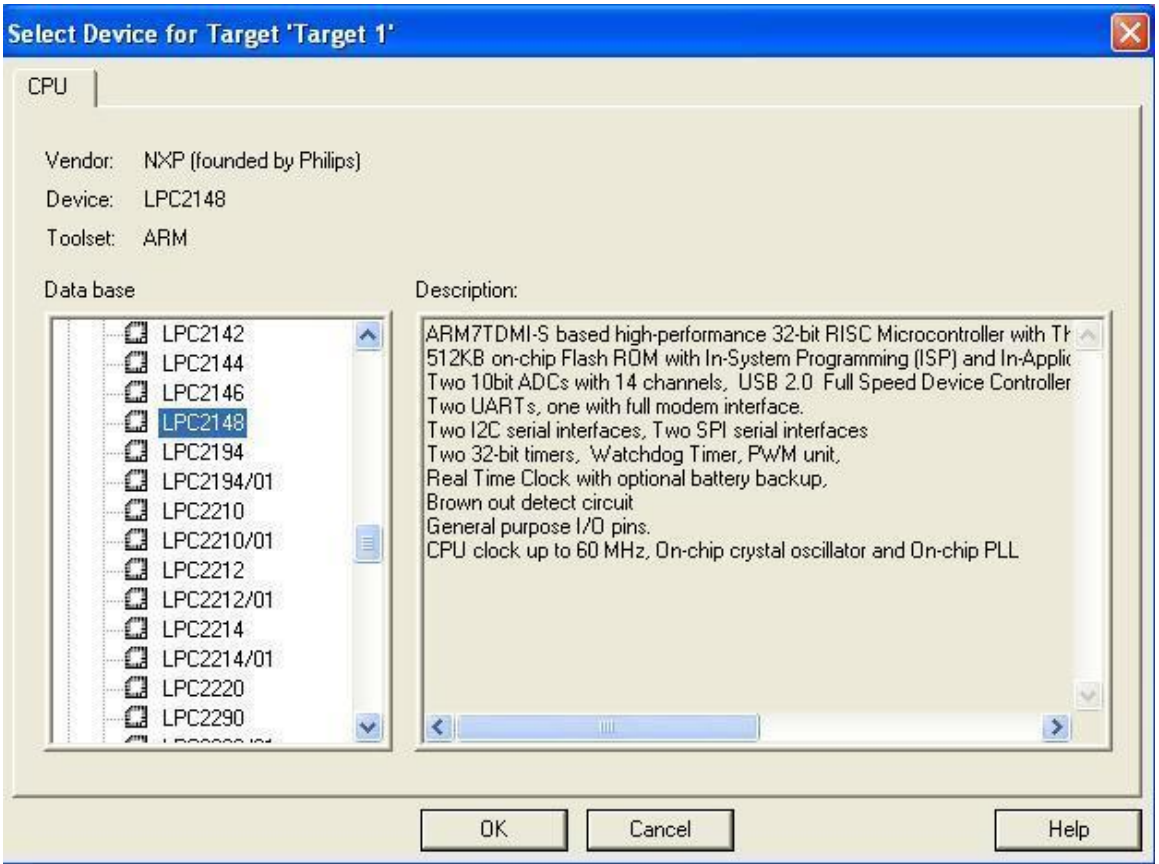
2. **Creating Project in KEIL and Generating HEX file**

- The LPC2148-EDU kit is compatible with many commercial and open source tools.
- The kit comes along with a evaluation version of KEIL IDE with 32K Code limit.
- To create a KEIL project for the experiments.
- The necessary „.c" and „.h" files required for the project are kept in the specified folder
  1. Run the KEIL IDE and go the
  2. Select the CPU target as LPC2148 under the option of NXP in the list

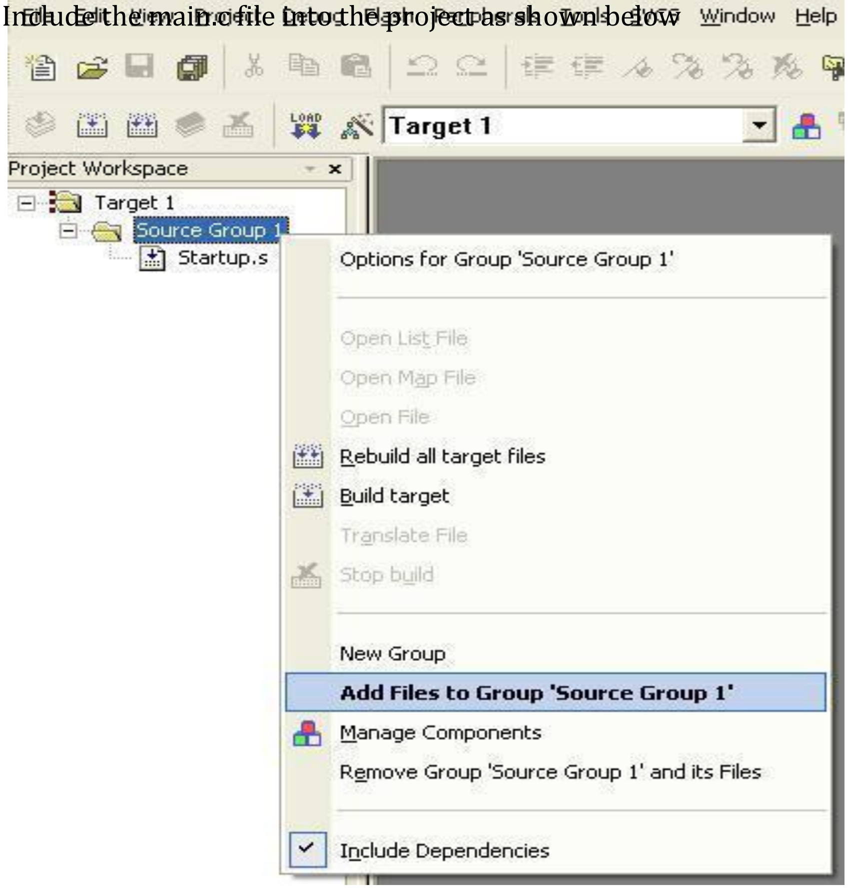
3.

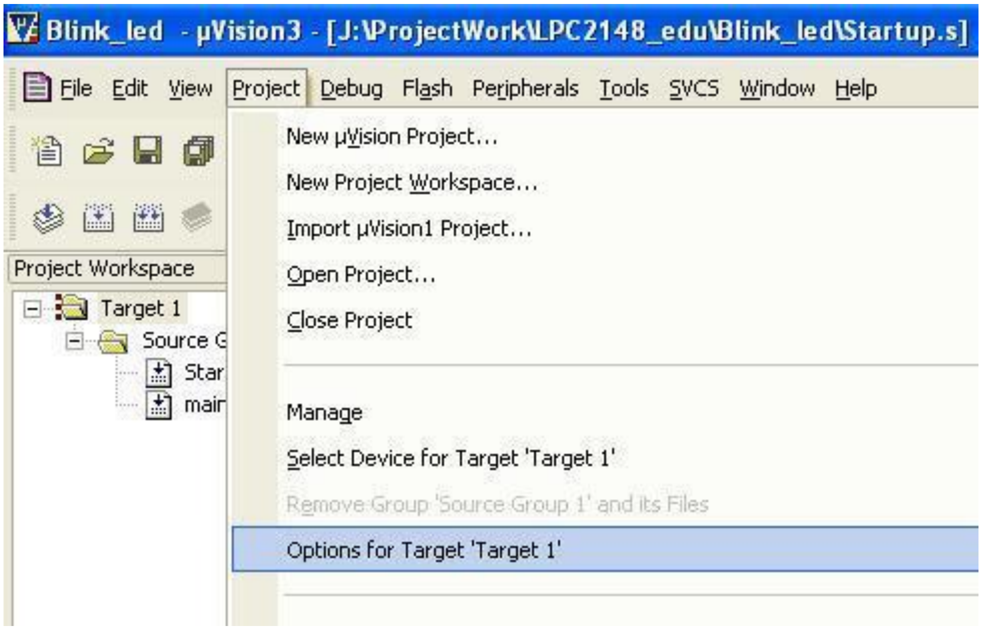
4. Let's give the project a name and save it as shown below





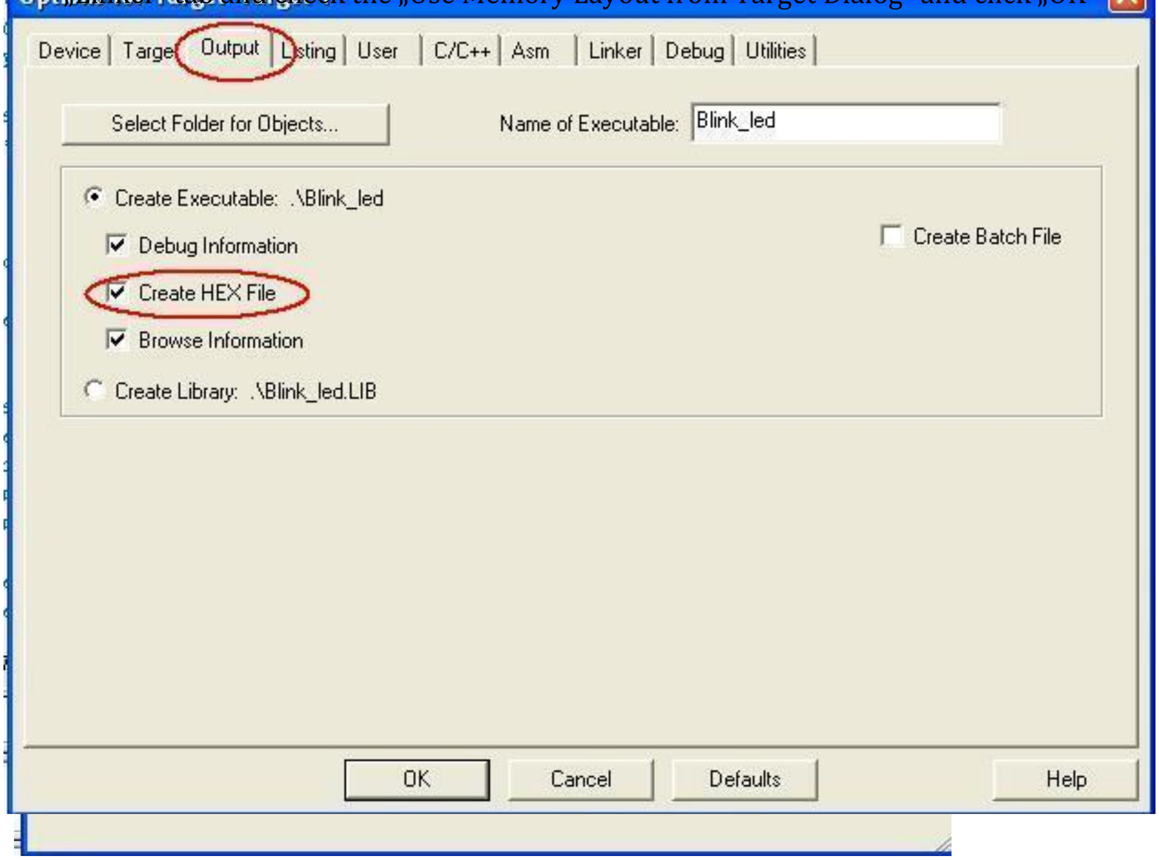
5. Copy the main.c file under the folder where we have the project file saved
6. Include the main.c file into the project as shown below

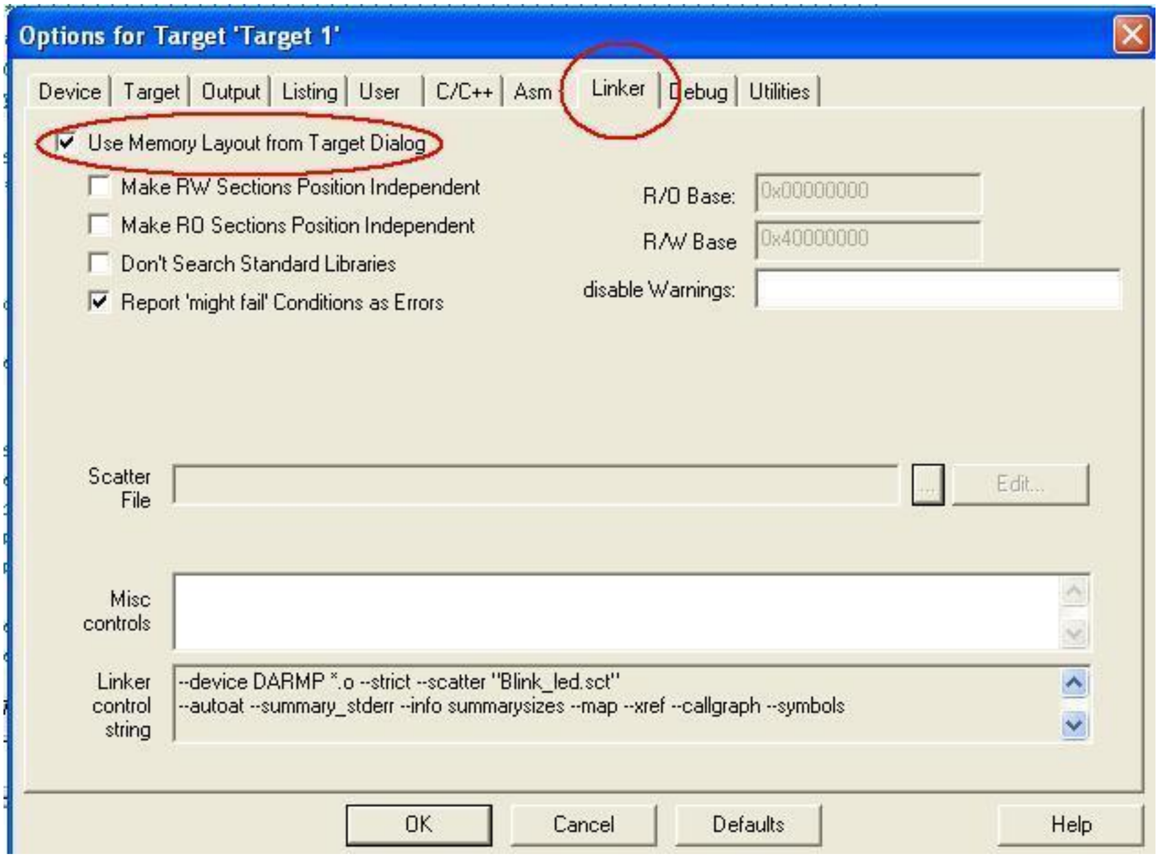




7. Perform the following settings: „Click on Target1" in the left pain, and then go to

„Options for target1" as shown below and in the next pop-up window select the „Output" tab and check the option „Create HEX file" as shown below. Next go to „Linker" tab and check the „Use Memory Layout from Target Dialog" and click „OK"





8. To compile the project click on the icon on top of the left pane, as highlighted below. If the project is successfully compiled a HEX file would be generated in the same path where we have stored the project file. Flash the HEX file using HJTAG Tool.

Experiment No. 1 **Date:**

AIM:

- To Practice Using Keil IDE and Data Processing Instructions of ARM Instruction Set.

i) Swap two numbers without using an intermediate register.
ii) To find the factorial of a given number.
iii) Convert word of little endian format to big endian format.

Circuit /Logic / Block Diagram:

i) Swap two numbers without using an intermediate register.

The numbers to be swapped are placed in 2 registers and Ex-OR operation between the registers is carried out in order specified below.
R0^R1
R1^R0
R0^R1
The results are verified in the registers R0 & R1.

ii) To find the factorial of a given number

This program takes an integer and placed in the register R6. The factorial is found using n! = n*(n-1)*(n-2)*(n-3)...3*2*1. The result is verified in the register R7.

iii) Convert word of little endian format to big endian format

In little endian, you store the least significant byte in the smallest address. In big endian, you store the most significant byte in the smallest address. The number in little endian format is placed in register R0. Using R2 and R3 register contents and by using appropriate logical and shift operations are carried out to obtain R1 & R0 values as specified below.

R2 = 0X00FF00FF
R3 = 0XFF00FF00
R0 = 0X0A0B0C0D
R1 = 0X000C000A
R0 = 0X0D000B00
R0 = 0X0D0C0B0A
The result is verified in the register R0.

Equipment's / Components / Apparatus Required:

Keil IDE simulator, Window's Calculator

Design / Program:

I. SWAPPING OF 2 NUMBERS WITHOUT USING INTERMEDIATE REGISTER

```
        AREA Rotate, CODE, READONLY
        ENTRY

                LDR     R0,=0XF631024C
                LDR     R1,=0X17539ABD
                EOR     R0,R0,R1                        ;R0^R1
                EOR     R1,R0,R1                        ;R1^R0
                EOR     R0,R0,R1                        ;R0^R1

        stop    B       stop                            ;stop program
                END
```

II. FIND THE FACTORIAL OF A GIVEN NUMBER

```
                AREA Factorial, CODE, READONLY
                ENTRY

                MOV             R6,#03
                MOV             R4,R6
LOOP            SUBS            R4,R4,#1
                MULNE           R7,R6,R4
                MOV             R6,R7
                BNE             LOOP

        stop    B       stop
                END
```

III. CONVERT WORD OF LITTLE ENDIAN TO BIG ENDIAN FORMAT

```
                AREA Rotate, CODE, READONLY
                ENTRY

                MOV  R2,#0XFF                           ;R2=0XFF
                ORR  R2,R2,#0XFF0000                    ;R2=0X00FF00FF
                MOV  R3,R2,LSL #8               ;R3=0XFF00FF00
                                                ;R0=A B C D
                AND     R1,R2,R0,ROR #24        ;R1=0 C 0 A
                AND     R0,R3,R0,ROR #8         ;R0=D 0 B 0
                ORR     R0,R0,R1                ;R0 = D C B A

        stop    B       stop                                    ;stop program
                END
```

<u>Procedure for Conduction</u>

- Create a project and choose a simple microcontroller such as NXP's LPC 2148 as your target.
- Create a new assembly file and attach the source file to the project.
- Start the debugger and single step through the code or perform run the entire program at once. Breakpoints can be used for debugging purpose.
- Examine the changes that occur to registers and memory contents

Calculations

Verify your results for finding factorial for given number using Windows Calculator. The calculator is in the Accessories folder of Windows. Select the scientific calculator.

Observations

Fill the table of registers with its contents after execution of specific instructions.

Conclusion from the experiment

Write conclusions about utility of simulator for debugging programs.
Write conclusions based on the observations made while debugging the programs.

Follow Up Questions

A. Use MOV instruction in place of pseudo instruction LDR for moving 32 bit data in to register and observe results.
B. With respect to instruction set justify how ARM design philosophy differs from RISC design philosophy.

Experiment No. 2 **Date:**

AIM:

- To Practice Using Data Processing Instructions of ARM Instruction Set.

Generate 12 bit Hamming code from a given 8 bit code.

Circuit /Logic / Block Diagram:

Generate 12 bit Hamming code from a given 8 bit code

Original 8 – bit value

d7	d6	d5	d4	d3	d2	d1	d0

Modified 8 – bit value

d7	d6	d5	d4	c3	d3	d2	d1	c2	d0	c1	c0

The checksum bits are computed as follows:

- Checksum bit c0 should produce even parity for bits 0, 2, 4, 6, 8 and 10. In other words, we're checking a bit, checking a bit, etc.
- Checksum bit c1 should produce even parity for bits 1, 2, 5, 6, 9, and 10. In other words, we're checking two bits, skipping two bits, checking two bits, etc.
- Checksum bit c2 should produce even parity for bits 3, 4, 5, 6, and 11. Now we're checking four bits, skipping four bits etc.
- Checksum bit c3 should produce even parity for bits 7, 8, 9, 10, and 11.

Equipment's / Components / Apparatus Required:

Keil IDE simulator.

Design / Program:

GENERATE 12 BIT HAMMING CODE FROM A GIVEN 8 BIT CODE

```
        AREA ROTT, CODE, READONLY
        ENTRY
MAIN
        MOV     R2,#0
        ADR     R1,ARRAYA
        LDRB    R0,[R1]

        ;CAL C0

        MOV     R4,R0
        EOR     R4,R4,R0,ROR #1
        EOR     R4,R4,R0,ROR #3
        EOR     R4,R4,R0,ROR #4
        EOR     R4,R4,R0,ROR #6
        AND     R2,R4,#1

        ;CAL C1
```

```
        MOV     R4,R0
        EOR     R4,R4,R0,ROR #2
        EOR     R4,R4,R0,ROR #3
        EOR     R4,R4,R0,ROR #5
        EOR     R4,R4,R0,ROR #6
        AND     R4,R4,#1
        ORR     R2,R2,R4,LSL #1

        ;CAL C2

        MOV     R4,R0,ROR #1
        EOR     R4,R4,R0,ROR #2
        EOR     R4,R4,R0,ROR #3
        EOR     R4,R4,R0,ROR #7
        AND     R4,R4,#1
        ORR     R2,R2,R4,ROR #29

        ;CAL C3

        MOV     R4,R0,ROR #4
        EOR     R4,R4,R0,ROR #5
        EOR     R4,R4,R0,ROR #6
        EOR     R4,R4,R0,ROR #7
        AND     R4,R4,#1

        ;CAL FINAL 12 BIT

        ORR     R2,R2,R4,ROR #25        ;ROTATE LEFT 7 BITS
        AND     R4,R0,#1                ;GET BIT 0 FROM ORIGINAL
        ORR     R2,R2,R4,LSL #2         ;ADD BIT 0 INTO FINAL
        BIC     R4,R0,#0XF1             ;GET BITS 3,2,1
        ORR     R2,R2,R4,LSL #3         ;ADD BITS 3,2,1 TO FINAL
        BIC     R4,R0,#0X0F              ;GET UPPER NIBBLE
        ORR     R2,R2,R4,LSL #4         ;R2 NOW CONTAINS 12 BITS WITH
                                        ;CHECKSUM

STOP    B       STOP
;       ALIGN
ARRAYA
        DCB     0XB5
        DCB     0XAA
        DCB     0X55
        DCB     0XAA
        END
```

<u>Procedure for Conduction</u>

- Create a project and choose a simple microcontroller such as NXP's LPC 2148 as your target.
- Create a new assembly file and attach the source file to the project.

- Start the debugger and single step through the code or perform run the entire program at once. Breakpoints can be used for debugging purpose.
- Examine the changes that occur to registers and memory contents

Calculations

Verify your results using manual calculation.

Observations

Fill the table of registers with its contents after execution of specific instructions.

Conclusion from the experiment

Write conclusions based on the observations made while debugging the programs.

Follow Up Questions

A. Explain pseudo instructions LDR and ADR
B. Translate the following conditions into a single ARM instruction
    a. Add registers R3, R6 only if N is clear. Store the result in R7
    b. Multiply registers R7 and R12, putting the results in register R3 only if C is set and Z is clear.
    c. Compare registers R6 and R8 only if Z is clear.

Experiment No. 3 **Date:**

AIM:

- To Practice Using Data Processing, Branch and Load - Store Instructions of ARM Instruction Set.

i) Move a string from given memory location to another location
ii) To Add N numbers of data stored consecutively in memory location
iii) Translate the given C code to assembly.
for (i=0;i<8;i++){a[i]=b[7-i];}

Circuit /Logic / Block Diagram:

i) Move a string from given memory location to another location
- Source string is stored in memory just after the program code, starting address of string is loaded into register R1.
- The address of the destination is loaded into the register R0.
- The character/data is loaded from memory into register R2 and then stored into destination location.
- All the characters of the string are moved to the destination location. Comparison is made to check whether null character is reached.

ii) To Add N numbers of data stored consecutively in memory location
- Clear register R0 to have accumulated sum.
- Start of the array in memory is loaded into register R2
- Register R3 is loaded with one word of data and the value is then added into the accumulated sum.
- The counter of the loop is decremented and the loop terminates once the counter becomes negative.

iii) Translate the given C code to assembly.
for (i=0;i<8;i++){a[i]=b[7-i];}
Arrays a & b are assumed to contain byte wide data.
- Need to have the array a be located in writable memory and start address is moved to R2.
- The address of array b is loaded into register R1.
- The reverse subtract operation calculates difference between 7 and i to use as pointer into memory.
- The data is loaded from memory into register R5 and then stored into array.
- Counter is incremented and tested against 8 to see is its even necessary to move any data.

Equipment's / Components / Apparatus Required:
Keil IDE simulator, Window calculator.

Design / Program:

i) MOVE A STRING FROM GIVEN MEMORY LOCATION TO ANOTHER LOCATION

```
        SRAM_BASE EQU 0X40000000
        AREA Rotate, CODE, READONLY
        ENTRY
MAIN
        ADR     R1,SRCSTR
        LDR     R0,=SRAM_BASE
```

```
STRCPY
            LDRB  R2,[R1],#1
            STRB  R2,[R0],#1
            CMP   R2,#0
            BNE   STRCPY
stop        B     stop                          ;stop program

SRCSTR      DCB   "HELLO",0
            END
```

ii) TO ADD N NUMBERS OF DATA STORED CONSECUTIVELY IN MEMORY LOCATION

```
            AREA Rotate, CODE, READONLY
            ENTRY
            MOV         R0,#0
            MOV         R1,#5
            ADR         R2,ARRAYA

LOOP        LDR         R3,[R2,R1,LSL #2]
            ADD         R0,R3,R0
            SUBS        R1,R1,#1

            BGE         LOOP
stop        B           stop            ;stop program

            ALIGN
ARRAYA      DCD         1, 2, 3, 4, 5, 6
            END
```

iii) TRANSLATE THE FOLLOWING C CODE TO ASSEMBLY
for(i=0;i<8;i++)
{a[i]=b[7-i];}

```
            AREA Rotate, CODE, READONLY
            SRAM_BASE   EQU         0X40000000
            ENTRY
            MOV         R0,#0
            ADR         R1,ARRAYB
            MOV         R2,#SRAM_BASE

LOOP        CMP         R0,#8
            BGE         DONE
            RSB         R3,R0,#7
            LDRB        R5,[R1,R3]
            STRB        R5,[R2,R0]
            ADD         R0,R0,#1
            B           LOOP

DONE        B           DONE
```

```
            ALIGN
ARRAYB      DCB         0XA,0X9,0X8,0X7,0X6,0X5,0X4,0X3
            END
```

Procedure for Conduction

- Create a project and choose a simple microcontroller such as NXP's LPC 2148 as your target.
- Create a new assembly file and attach the source file to the project.
- Start the debugger and single step through the code or perform run the entire program at once. Breakpoints can be used for debugging purpose.
- Examine the changes that occur to registers and memory contents

Calculations

Verify your results for adding N numbers using Windows Calculator. The calculator is in the Accessories folder of Windows. Select the scientific calculator.

Observations

Fill the table of registers with its contents after execution of specific instructions.
Fill the table of Memory with the address and data of the numbers on which operation is performed and final results obtained.

Conclusion from the experiment

Write conclusions based on the observations made while debugging the programs.

Follow Up Questions

A. Write a routine that reverses the bits in a register, So that a register containing $d_{31}d_{30}.....d_1d_0$ now contains $d_0d_1.....d_{29}d_{30}d_{31}$
B. Give the bit pattern that Keil assembler produces for the instruction below. Explain why.

```
BIC     R6, R6, #0xFFFFFFFF
```

Experiment No. 4 **Date:**

AIM:

- To Practice Using Data Processing, Branch and Load - Store Instructions of ARM Instruction Set.

i) Move a block of data from memory location to another location using LOAD multiple and STORE multiple instructions.
ii) Exchange a block of data between memory locations.

Circuit /Logic / Block Diagram:

i) Move a block of data from memory location to another location using load multiple and store multiple instructions.
   - Register R9 is the pointer to source block. Register R10 is the pointer to destination block.
   - Register R0 holds the number of words to be copied.
   - Four words are copied at a time.
   - The counter of the loop is decremented and the loop terminates once the counter becomes negative.

ii) Exchange a block of data between memory locations
   - Register R9 is the pointer to source1 memory block. Register R10 is the pointer to source2 memory block.
   - The data is loaded from source1 address to register R1 and from source2 address to register R3
   - The data in register R1 is then stored into source2 memory.
   - The data in register R3 is then stored into source1 memory.
   - The pointer is incremented to hold the next memory addresses
   - The counter of the loop is decremented and the loop terminates once the counter becomes negative.

iii) Arrange a given set of data in ascending /descending order
   - The address of array to be sorted is loaded into register R1.
   - The first element is compared with adjacent elements of the array and swaps them if they are in wrong order.
   - Smaller elements bubbles up to the top of the list.
   - The counter of the loop is decremented and the loop terminates once the counter becomes negative.

Equipment's / Components / Apparatus Required:

Keil IDE simulator, Window calculator.

Design / Program:

i) MOVE A BLOCK OF DATA FROM MEMORY LOCATION TO ANOTHER LOCATION USING LOAD MULTIPLE AND STORE MULTIPLE INSTRUCTIONS.

```
AREA Rotate, CODE, READONLY
ENTRY
MOV         R0,#07
```

```
                LDR         R9,=0X40000000
                LDR         R10,=0X400000C0

LOOP            LDMIA  R9!,{R1-R4}
                STMIA  R10!,{R1-R4}

                SUBS   R0,#04
                BGT    LOOP
stop    B       stop                                    ;stop program


                END
```

ii) EXCHANGE A BLOCK OF DATA BETWEEN MEMORY LOCATIONS.

```
                AREA Rotate, CODE, READONLY
                ENTRY
                MOV         R0,#03
                LDR         R9,=0X40000000
                LDR         R10,=0X400000C0


LOOP1           LDR         R1,[R9]
                SWP         R3,R1,[R10]
                STR         R3,[R9],#04
                ADD         R10,#04


                SUBS   R0,#01
                BGT         LOOP1
stop    B              stop
```

iii) ARRANGE A GIVEN SET OF DATA IN ASCENDING / DESCENDING ORDER

```
                AREA Rotate, CODE, READONLY
                ENTRY
                MOV         R0,#4
                SUB         R0,R0,#1
                MOV         R5,R0

                LDR         R1,=0X40000000

LOOP2           MOV         R0,R5
                MOV         R2,R1
                ADD         R2,#4

L1              LDR         R3,[R1]
                LDR         R4,[R2]
                CMP         R3,R4
```

```
                BLE         LOOP1

                SWP         R3,R4,[R1]
                STR         R3,[R2]

        LOOP1   ADD         R2,#4
                SUBS        R0,R0,#1
                BGT         L1
                ADD         R1,#4
                SUBS        R5,R5,#1
                BGT         LOOP2

stop    B           stop                        ;stop program


        END
```

Procedure for Conduction

- Create a project and choose a simple microcontroller such as NXP's LPC 2148 as your target.
- Create a new assembly file and attach the source file to the project.
- Start the debugger and single step through the code or perform run the entire program at once. Breakpoints can be used for debugging purpose.
- Examine the changes that occur to registers and memory contents

Calculations

Observations

Fill the table of registers with its contents after execution of specific instructions.
Fill the table of Memory with the address and data of the numbers on which operation is performed and final results obtained.

Conclusion from the experiment

Write conclusions based on the observations made while debugging the programs.

Follow Up Questions

A. Write the assembly code to compute a dot product for 20 samples given as $a = \sum_{m=0}^{n-1} c_m x_m$. Leave the dot product in a register. The coefficients c_m and the samples x_m and the input samples are stored as arrays in memory.
B. Write a program that counts the number of ones in a 32 – bit value. Save the result in register R3.

Experiment No. 5 **Date:**

AIM:

- To Practice Using Data Processing, Branch and Load - Store Instructions of ARM Instruction Set.

i) Arrange a given set of data in ascending
ii) Arrange a given set of data in descending order

Circuit /Logic / Block Diagram:

i) Arrange a given set of data in ascending /descending order
   - The address of array to be sorted is loaded into register R1.
   - The first element is compared with adjacent elements of the array and swaps them if they are in wrong order.
   - Larger elements comes up to the top of the list.
   - The counter of the loop is decremented and the loop terminates once the counter becomes negative.

Equipment's / Components / Apparatus Required:
Keil IDE simulator, Window calculator.

Design / Program:

i) ARRANGE A GIVEN SET OF DATA IN ASCENDING / DESCENDING ORDER

```
        AREA Rotate, CODE, READONLY
        ENTRY
        MOV     R0,#4
        SUB     R0,R0,#1
        MOV     R5,R0

        LDR     R1,=0X40000000

LOOP2   MOV     R0,R5
        MOV     R2,R1
        ADD     R2,#4

L1      LDR     R3,[R1]
        LDR     R4,[R2]
        CMP     R3,R4
        BLE     LOOP1

        SWP     R3,R4,[R1]
        STR     R3,[R2]

LOOP1   ADD     R2,#4
        SUBS    R0,R0,#1
        BGT     L1
        ADD     R1,#4
```

```
                SUBS        R5,R5,#1
                BGT         LOOP2

        stop    B           stop                        ;stop program


                END
```

Procedure for Conduction

- Create a project and choose a simple microcontroller such as NXP's LPC 2148 as your target.
- Create a new assembly file and attach the source file to the project.
- Start the debugger and single step through the code or perform run the entire program at once. Breakpoints can be used for debugging purpose.
- Examine the changes that occur to registers and memory contents

Calculations

Observations

Fill the table of registers with its contents after execution of specific instructions.
Fill the table of Memory with the address and data of the numbers on which operation is performed and final results obtained.

Conclusion from the experiment

Write conclusions based on the observations made while debugging the programs.

Follow Up Questions

A. Write the assembly code to arrange a given set of data in descending order

Experiment No. 6 **Date:**

AIM:

- To Practice Using Stack, SWI Instructions of ARM Instruction Set and to work on state change from ARM to THUMB..

  i) Implement subroutine nesting using stack
  ii) To implement ARM –THUMB interworking to find the smallest.
  iii) To handle SWI instruction in the program

Circuit /Logic / Block Diagram:

i) Implement subroutine nesting using stack
- Passing parameters to subroutine is through stack.
- Data is pushed onto stack before the subroutine call.
- The subroutine grabs the data off the stack to be used.
- Results are then stored back onto the stack to be retrieved by the calling routine.

ii) To implement ARM –THUMB interworking to find the smallest.
- BX and BLX instruction is used to jump to an address called thumbcode.
- The short section of code begins with the directive CODE16, indicating the following instructions are THUMB instructions.
- The THUMB code that follows finds the smallest in an array by comparing the first element in the array with the other items in the array.
- The first element in the array stores the smallest.

iii) To handle SWI instruction in the program
- Demonstrates how processor behaves during SWI instruction execution.
- In SWI handler, the entire instruction from code memory is taken and SWI number is extracted.

Equipment's / Components / Apparatus Required:

Keil IDE simulator, Window calculator.

Design / Program:

i) ALP TO IMPLEMENT SUBROUTINE NESTING USING STACK

```
                AREA ROT,CODE,READONLY
SRAM_BASE       EQU  0X40000000
                ENTRY
MAIN
                LDR SP,=SRAM_BASE
                MOV R5,#0X05
                MOV R6,#0X06

                BL  FOURADD
STOP            B  STOP

FOURADD
```

```
                STMEA  SP!,{R5,R6,LR}
                MOV  R5,R5,LSL #2
                MOV  R6,R6,LSL #2
                ADD R0,R5,R6
                BL FOURSUB
                MOV PC,R7
FOURSUB
                LDMEA SP!,{R5,R6,R7}
                MOV R5,R5,LSL #2
                MOV R6,R6,LSL #2
                SUB R1,R5,R6
                MOV PC,LR
                END
```

ii) WRITE AN ALP TO IMPLEMENT ARM-THUMB INTERWORKING TO FIND SMALLEST

```
    AREA Rotate, CODE, READONLY

            ENTRY

            ; ARM code
            CODE32          ; word aligned

            LDR     r0, =thumbCode+1        ; +1 to enter Thumb state
            MOV   lr, pc

            BX r0

    STOP    B       STOP
            ; Thumb code
            CODE16  ; halfword aligned

    thumbCode
            LDR     R0,=0X40000000
            MOV     R1,R0
            MOV     R5,#3

    L2      ADD     R1,#4
            LDR     R2,[R0]
            LDR     R3,[R1]

            CMP     R2,R3
            BLE     L1

            STR             R3,[R0]
            STR             R2,[R1]

      L1    SUB     R5,#1
            BNE             L2
            BX      lr   ; return to ARM code & state
```

```
END
```

iii) TO HANDLE SWI INSTRUCTION IN THE PROGRAM

```
                AREA SWI_TEST, CODE, READONLY
                LDR PC,Reset_Addr
                LDR PC,Undef_Addr
                LDR PC,SWI_Addr
                LDR PC,PAbt_Addr
                LDR PC,DAbt_Addr
                NOP
                LDR PC,IRQ_Addr
                LDR PC,FIQ_Addr

Reset_Addr      DCD Reset_Handler
Undef_Addr      DCD Undef_Handler
SWI_Addr        DCD SWI_Handler
PAbt_Addr       DCD PAbt_Handler
DAbt_Addr       DCD DAbt_Handler
                DCD 0
IRQ_Addr        DCD IRQ_Handler
FIQ_Addr        DCD FIQ_Handler
Undef_Handler   B Undef_Handler
SWI_Handler     B SWI_Handler1
PAbt_Handler    B PAbt_Handler
DAbt_Handler    B DAbt_Handler
IRQ_Handler     B IRQ_Handler
FIQ_Handler     B FIQ_Handler

Reset_Handler
                MOV R3,#25
                MOV R7,#207
                ADD R2,R3,R7
                SWI 0X12
STOP            BAL     STOP

SWI_Handler1
                SUB R0,LR,#4
                LDR R1,[R0]
                BIC R1,#0XFF000000
                MOV PC,LR
                END
```

Procedure for Conduction

- Create a project and choose a simple microcontroller such as NXP's LPC 2148 as your target.
- Create a new assembly file and attach the source file to the project.
- Start the debugger and single step through the code or perform run the entire program at once. Breakpoints can be used for debugging purpose.
- Examine the changes that occur to registers and memory contents

Observations

Fill the table of registers with its contents after execution of specific instructions.
Fill the table of Memory with the address and data of the numbers on which operation is performed and final results obtained.

Conclusion from the experiment

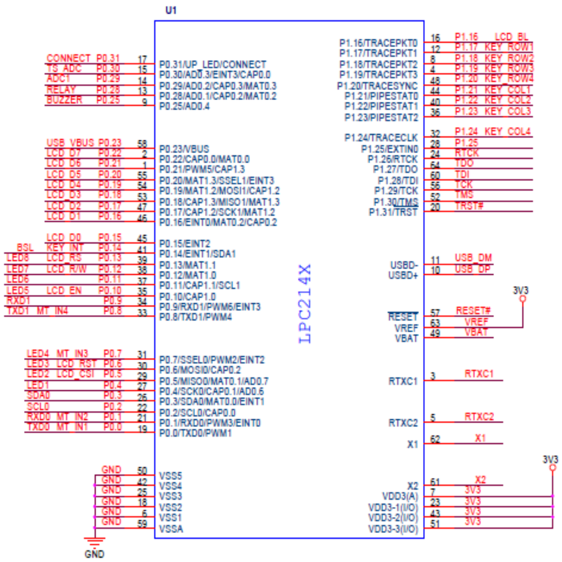
Write conclusions based on the observations made while debugging the programs.

Follow Up Questions

A. Create a queue of 32 – bit data values in memory. Write a function to remove the first item in the queue.
B. Write a parity checker routine that examines a byte in memory for correct parity. For even parity, the number of ones in a byte should be an even number. For odd parity, the number of ones should be an odd number. Create two small blocks of data, one assumed to have even parity and the other assumed to have odd parity. Introduce errors in both sets of data, writing the value OXDEADDEAD into register R0 when an error occurs.

Part B: C programs

Pin Diagram:



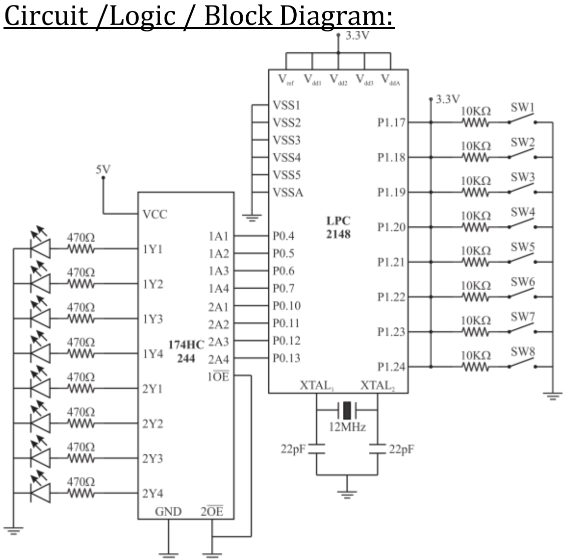
Experiment No. 7 **Date:**

AIM:

- To understand the input and output ports of LPC 2148

Interface LED's to LPC 2148. Write a C program to control the ON – OFF of the LED through switches.

Circuit /Logic / Block Diagram:



Equipment's / Components / Apparatus Required:

Keil IDE simulator, Flash Magic or HJTAG software's installed, LPC 2148 datasheet, LPC 2148 kit with power adapter, serial and parallel cable.

Design / Program:

```
#include <LPC214x.H>                    /* LPC21xx definitions */

void wait (void)  {                    /* wait function */
  int  d;

  for (d = 0; d < 2000000; d++);        /* only to delay for LED flashes */
}

delay()
{
int j,k,i;
for(i=0;i<30;i++)
for(j=0;j<300;j++)
for(k=0;k<300;k++);
}

#define SWITCH1 (1 << 17)              //P1.17 t0 P1.24 I/p DIP keys, SWD1 to SWD_8
#define SWITCH2 (1 << 18)
#define SWITCH3 (1 << 19)
#define SWITCH4 (1 << 20)
```

```
#define SWITCH5 (1 << 21)
#define SWITCH6 (1 << 22)
#define SWITCH7 (1 << 23)
#define SWITCH8 (1 << 24)

#define LED1 (1 << 4)                    //P0.4 to P0.7 & P0.10 to P0.13 for LEDS
#define LED2 (1 << 5)
#define LED3 (1 << 6)
#define LED4 (1 << 7)
#define LED5 (1 << 10)
#define LED6 (1 << 11)
#define LED7 (1 << 12)
#define LED8 (1 << 13)


void main()
{
int i;
IO0DIR= 0XFFFFFFFF;  // MAKING ALL O/PS
IO1DIR=0X00000000;  //MAKING AS I/PS FOR DIP SWITCH

delay();


while(1)
        {
         if(IO1PIN&SWITCH1)
           IO0CLR|=LED1;
          else
                IO0SET|=LED1;
wait();

        if(IO1PIN&SWITCH2)
           IO0CLR|=LED2;
          else
                IO0SET|=LED2;

wait();

if(IO1PIN&SWITCH3)
           IO0CLR|=LED3;
          else
                IO0SET|=LED3;
wait();

        if(IO1PIN&SWITCH4)
           IO0CLR|=LED4;
          else
                IO0SET|=LED4;
```

```
wait(); if(IO1PIN&SWITCH5)
          IO0CLR|=LED5;
         else
              IO0SET|=LED5;
wait();

        if(IO1PIN&SWITCH6)
          IO0CLR|=LED6;
         else
              IO0SET|=LED6;

wait();
        if(IO1PIN&SWITCH7)
          IO0CLR|=LED7;
         else
              IO0SET|=LED7;
wait();

        if(IO1PIN&SWITCH8)
          IO0CLR|=LED8;
         else
              IO0SET|=LED8;

wait();
        }

}
```

Procedure for Conduction

- Create a project and choose a simple microcontroller such as NXP's LPC 2148 as your target.
- Create a new C file and attach the source file to the project.
- Create HEX file and flash the HEX file using HJTAG Tool.
- Start the debugger and single step through the code or perform run the entire program at once. Breakpoints can be used for debugging purpose.
- Examine the output.

Observations

Observe the LEDs connected to ports is activated or deactivated by pressing / depressing the corresponding switches.

Conclusion from the experiment

Write conclusions based on the observations made while debugging the programs.

Follow Up Questions

A. Write a C program to flash the LEDs serially that are connected to port pins of LPC 2148.
B. Write a C program to flash a pattern on LEDs by sending bit patterns 0x81, 0x42, 0x24, 0x18 to port pins of LPC 2148 continuously.

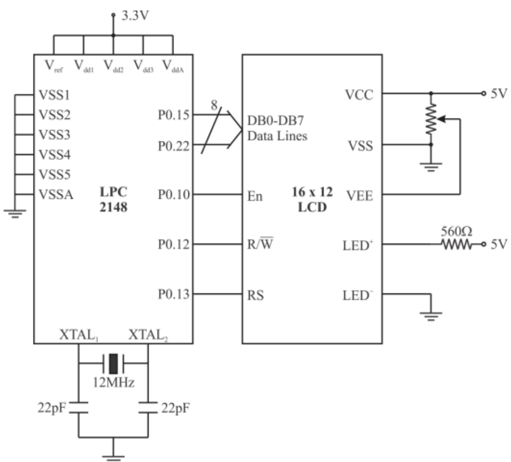
Experiment No. 8 **Date:**

AIM:

- To understand the connections of LCD module and the commands to control the LCD module.

Interface LCD module to LPC 2148. Write a C program to display a string on LCD module.

Circuit /Logic / Block Diagram:



Equipment's / Components / Apparatus Required:

Keil IDE simulator, Flash Magic or HJTAG software's installed, LPC 2148 datasheet, LPC 2148 kit with power adapter, serial and parallel cable.

Design / Program:

```
#include <LPC214X.H>



#define LCDRS (1 << 13)          //0 for cmd n 1 for data
#define LCDRW       (1 << 12)    //0 for write n 1 for read
#define LCDEN(1 << 10)           //0 for disable, 1 for normal oprn n 1 to 0 for tx data/cmd

#define LCD_D0 (1 << 15)         //port 0
#define LCD_D1 (1 << 16)
#define LCD_D2 (1 << 17)
#define LCD_D3 (1 << 18)
#define LCD_D4 (1 << 19)
#define LCD_D5 (1 << 20)
#define LCD_D6 (1 << 21)
#define LCD_D7 (1 << 22)

unsigned char _data[7]=" MSRIT ";
unsigned int _code[3]={0x00070000,0x00008000,0x00400000};

void display_data(unsigned char);
void display_code(unsigned int ) ;
```

```
void wait (void)  {                    /* wait function */
  int  d;

  for (d = 0; d < 20000; d++);
}

delay(int sec)
{
int i,j;
for(i=0;i<(sec*10);i++)
for(j=0;j<10000;j++);
}

void display_code(unsigned int code)
{
IO0CLR=0xffffffff&0xffffffff;
delay(1);
IO0SET|=code;//0x00070000;

delay(3);
IO0CLR|=LCDRS;
IO0CLR|=LCDRW ;
IO0SET|=LCDEN;
delay(5);
IO0CLR|=LCDEN;
delay(3);
}

void display_data(unsigned char data)
        {
        unsigned int m=0;
        IO0CLR=0xffffffff&0xffffffff;//for R  0x001f8000&0xffffffff;
        delay(5);
        m = data<<15;
        IO0SET|=m&0xffffffff;
        delay(5);
        IO0SET|=LCDRS;
        IO0CLR|=LCDRW ;
        IO0SET|=LCDEN;
        delay(5);
        IO0CLR|=LCDEN;
        delay(5);
        }

void main()
{
int i=0;
IO0DIR=0xffffffff&0xffffffff;
for(i=0;i<3;i++)
{
```

```
display_code(_code[i]);
}

for(i=0;i<7;i++)
{
display_data(_data[i]);
}
}
```

Procedure for Conduction

- Create a project and choose a simple microcontroller such as NXP's LPC 2148 as your target.
- Create a new C file and attach the source file to the project.
- Create HEX file and flash the HEX file using HJTAG Tool.
- Start the debugger and single step through the code or perform run the entire program at once. Breakpoints can be used for debugging purpose.
- Examine the output.

Observations

Observe the display of string on the LCD module connected to LPC 2148.

Conclusion from the experiment

Write conclusions based on the observations made while debugging the programs.

Follow Up Questions

A. Modify the program to display the string in the second row and in the middle of the LCD Module.
B. Modify the program to display a moving string. String should move in the first row and later switch to second row and starts to move.

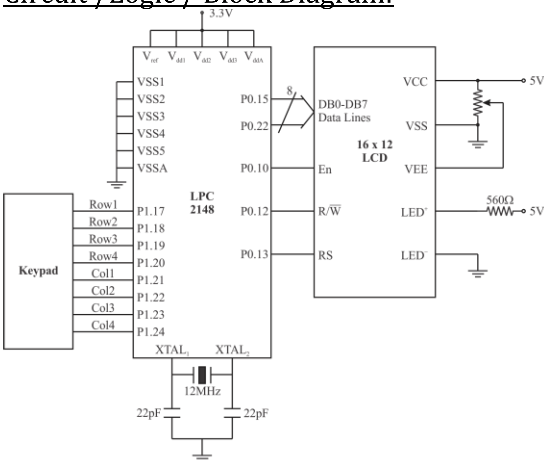
Experiment No. 9 **Date:**

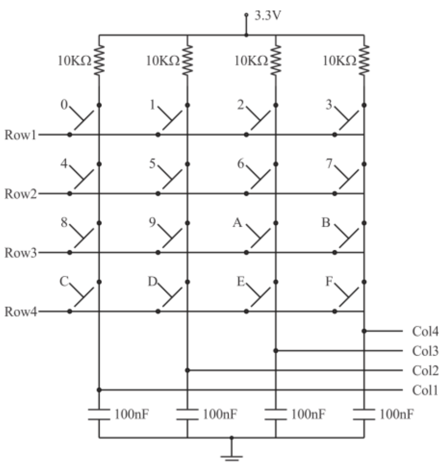
AIM:

- To understand the programming for sensing a key of a matrix keyboard.

Interface key pad and LCD module to LPC 2148. Write a C program to recognize the key press and display it on LCD module.

Circuit /Logic / Block Diagram:





Equipment's / Components / Apparatus Required:

Keil IDE simulator, Flash Magic or HJTAG software's installed, LPC 2148 datasheet, LPC 2148 kit with power adapter, serial and parallel cable.

Design / Program:

```
#include <LPC214X.H>

#define LCDRS       (1 << 13)     //0 for cmd n 1 for data
#define LCDRW       (1 << 12)     //0 for write n 1 for read
#define LCDEN       (1 << 10)     //0 for disable, 1 for normal oprn n 1 to 0 for tx data/cmd

#define LCD_D0 (1 << 15)          //port 0
```

```
#define LCD_D1 (1 << 16)
#define LCD_D2 (1 << 17)
#define LCD_D3 (1 << 18)
#define LCD_D4 (1 << 19)
#define LCD_D5 (1 << 20)
#define LCD_D6 (1 << 21)
#define LCD_D7 (1 << 22)


#define row1 (1 << 17)                                    //P1.17 t0 P1.24 for keypad
#define row2 (1 << 18)
#define row3 (1 << 19)
#define row4 (1 << 20)
#define col1 (1 << 21)
#define col2 (1 << 22)
#define col3 (1 << 23)
#define col4 (1 << 24)

unsigned char _data[7]=" MSRIT ";
unsigned int _code[6]={0x00070000,0x00008000,0x00400000};

void display_data(unsigned char);
void display_code(unsigned int ) ;

void wait (void)  {                    /* wait function */
  int  d;

  for (d = 0; d < 20000; d++);          /* only to delay for LED flashes */
}

delay(int sec)
{
int i,j;
for(i=0;i<(sec*10);i++)
for(j=0;j<10000;j++);
}

void keypad(int row)
{

unsigned char digit1[16]="0123456789ABCDEF";

wait();

if(!(IO1PIN&col1))
   display_data(digit1[((4*(row)))]);

wait();

if(!(IO1PIN&col2))
```

```
    display_data(digit1[((4*(row))+1)]);

wait();

if(!(IO1PIN&col3))
    display_data(digit1[((4*(row))+2)]);

wait();

if(!(IO1PIN&col4))
    display_data(digit1[((4*(row))+3)]);

wait();

}

void display_code(unsigned int code)
{
IO0CLR=0xffffffff&0xffffffff;
delay(1);
IO0SET|=code;//0x00070000;

delay(3);
IO0CLR|=LCDRS;
IO0CLR|=LCDRW ;
IO0SET|=LCDEN;
delay(5);
IO0CLR|=LCDEN;
delay(3);
}

void display_data(unsigned char data)
        {
        unsigned int m=0;
        IO0CLR=0xffffffff&0xffffffff;//for R  0x001f8000&0xffffffff;
        delay(5);
        m = data<<15;
        IO0SET|=m&0xffffffff;
        delay(5);
        IO0SET|=LCDRS;
        IO0CLR|=LCDRW ;
        IO0SET|=LCDEN;
        delay(5);
        IO0CLR|=LCDEN;
        delay(5);
        }
void main()
{
int i=0,n=0;
```

```
IO0DIR=0xffffffff&0xffffffff;
for(i=0;i<3;i++)
{
display_code(_code[i]);
}
for(i=0;i<6;i++)
{
display_data(_data[i]);
}
 for(n=0;n<3;n++)
{
display_code(_code[n]);
}

while(1)
{
        for(i=17;i<21;i++)
        {
        IO1DIR=0x00000000&0xffffffff;
        IO1DIR|=(1<<i);
        keypad((i-17));
        }

}
}
```

Procedure for Conduction

- Create a project and choose a simple microcontroller such as NXP's LPC 2148 as your target.
- Create a new C file and attach the source file to the project.
- Create HEX file and flash the HEX file using HJTAG Tool.
- Start the debugger and single step through the code or perform run the entire program at once. Breakpoints can be used for debugging purpose.
- Examine the output.

Observations

Observe the display of sensed key on the LCD module.

Conclusion from the experiment

Write conclusions based on the observations made while debugging the programs.

Follow Up Questions

A. Modify the program so as to assign different key code to each key starting from ASCII 'A' to ASCII 'P'
B. Modify the program so as to display a string for each key press.

Experiment No. 10 & 11 **Date:**

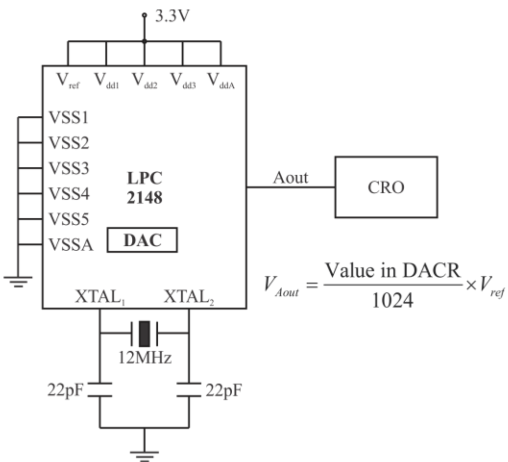
AIM:

- To understand the DAC and ADC peripheral of LPC 2148.

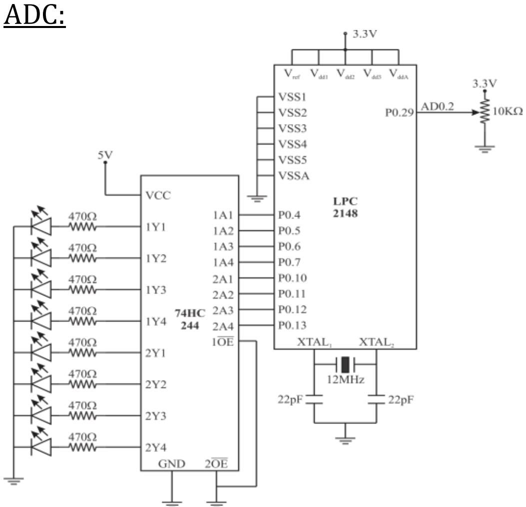
i) Waveform generation using the internal DAC of LPC 2148.
ii) To convert a given analog voltage to digital using ADC of LPC 2148

Circuit /Logic / Block Diagram:

DAC:



ADC:



Equipment's / Components / Apparatus Required:

Keil IDE simulator, Flash Magic or HJTAG software's installed, LPC 2148 datasheet, LPC 2148 kit with power adapter, serial and parallel cable.

Design / Program:

i) WAVEFORM GENERATION USING THE INTERNAL DAC OF LPC 2148.

```
#include <LPC214X.H>

delay()
```

```
{int i=0;for(i=0;i<400;i++);}

void sawtooth()
{
unsigned int i;
for(i=0;i<0x3ff;i++)
{
        DACR=0x00000000|(i<<6);
        delay();
}
}

void triangle()
{
unsigned int i;
for(i=0;i<0x3ff;i++)
{
        DACR=0x00000000|(i<<6);
        delay();
}

for(i=0x3ff;i>0;i--)
{
        DACR=0x00000000|(i<<6);
        delay();
}
}

square()
{
        unsigned int i=0X00000000;
        DACR=0x00000000|(i<<6);
        delay();
        i=0XFFFFFFFF;
        DACR=0x00000000|(i<<6);
        delay();

}
void main(void)
{

PINSEL1|=0x00080000;

while(1)
{
triangle();
}
}
```

ii) TO CONVERT A GIVEN ANALOG VOLTAGE TO DIGITAL USING ADC OF LPC 2148.

```
#include <LPC214X.H>

delay(int sec)
{int i,j;
for(i=0;i<sec;i++)
for(j=0;j<90000;j++);
}

void main()
{                                                   // 4 to 7 and 10 to 13
unsigned int result,val;
IO0DIR=0x0000fcf0;
PINSEL1=0x04000000&0xffffffff;//for A/D0.2;
AD0CR=0x00210504;  // for 10 bit n A/D0.2;
while(1)
{
val=AD0GDR;
result=((val>>8)&0x3ff);
IO0SET=(result<<4)&0x000000f0;
IO0SET=(result<<6)&0x0000fc00;
delay(1);
IO0CLR=0xffffffff&0xffffffff;
delay(1);
}
}
```

Procedure for Conduction

- Create a project and choose a simple microcontroller such as NXP's LPC 2148 as your target.
- Create a new C file and attach the source file to the project.
- Create HEX file and flash the HEX file using HJTAG Tool.
- Start the debugger and single step through the code or perform run the entire program at once. Breakpoints can be used for debugging purpose.
- Examine the output.

Observations

Observe waveforms in the simulator corresponding to the DAC output.
Observe the digital value on the LCD module for the given analog input.

Conclusion from the experiment

Write conclusions based on the observations made while debugging the programs.

Follow Up Questions

A. Modify the program to generate sine waveform using DAC peripheral of LPC 2148.
B. Modify the program to display the digital value from ADC peripheral onto the LCD module.

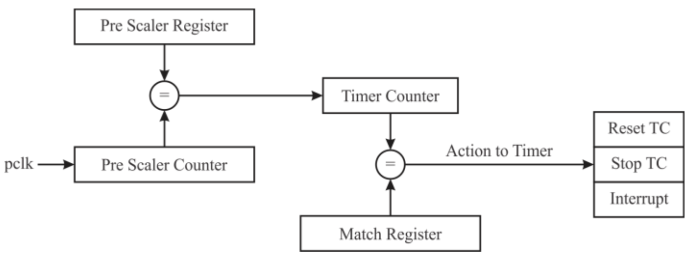
Experiment No. 12 **Date:**

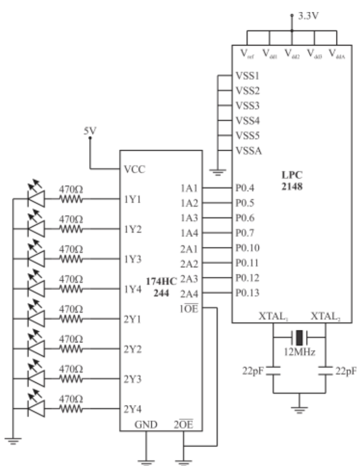
AIM:

- To understand the timer peripheral of LPC 2148 and enabling of timer interrupt.

i) Write a C program to flash a LED for every 1 sec. The delay must be generated using timer peripheral of LPC 2148.

Circuit /Logic / Block Diagram:





Equipment's / Components / Apparatus Required:
Keil IDE simulator, Flash Magic or HJTAG software's installed, LPC 2148 datasheet, LPC 2148 kit with power adapter, serial and parallel cable.

Design / Program:

```
#include <LPC214X.H>

#define LED1 (1 << 4)             //P0.4 for LED
#define LED2 (1 << 5)             //P0.5 for LED

void T0isr(void) __irq;

void delay()
{int i,j;
for(i=0;i<2000;i++)
for(j=0;j<1000;j++);
}
```

```
int main(void)
{
VPBDIV=0x00;

/*************interrupt enable*********/
VICIntSelect     = 0<<4;
VICVectAddr0  = (unsigned long)T0isr;
VICVectCntl0    = 0x020 | 4 ;
VICIntEnable    = 1<<4;
/************************************/


T0PR=0x00003A98&0xffffffff;
T0MR0=0x3E8&0xffff;
T0CTCR=0x00&0xff;
T0MCR=0x0003;
T0TCR=0x02;
T0TCR=0x01;

IO0DIR=0xffffffff;

while(1)
{
IO0SET|=LED2;
}
}

void T0isr(void) __irq
{
static int blink=0;
T0IR=0xff;
T0TCR=0x02;
if(blink==0)
        {
        IO0SET|=LED1;
        ++blink;
        }
        else if(blink==1)
        {
        IO0CLR|=LED1;
        --blink;
        }
        //delay();
T0TCR=0x01;
VICVectAddr    =           0x00;
}
```

<u>Procedure for Conduction</u>

- Create a project and choose a simple microcontroller such as NXP's LPC 2148 as your target.
- Create a new C file and attach the source file to the project.

- Create HEX file and flash the HEX file using HJTAG Tool.
- Start the debugger and single step through the code or perform run the entire program at once. Breakpoints can be used for debugging purpose.
- Examine the output.

Observations
Observe the flashing of LED continuously with the time interval given by the user.

Conclusion from the experiment
Write conclusions based on the observations made while debugging the programs.

Follow Up Questions

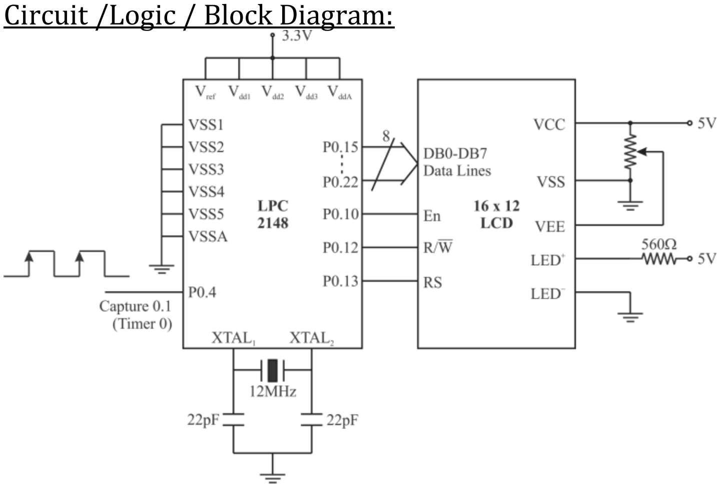
A. Write a C program to display ‘WELCOME’ & “MSRIT” alternatively for every 1 sec. The delay must be generated using timer peripheral of LPC 2148.
B. Modify the program to work for various delay values provided by the user.

Experiment No. 13 **Date:**

AIM:
- To understand the timer/counter/capture module peripheral of LPC 2148.

i) Write a program to count the number of pulses using timer/counter/capture module of LPC 2148 and display the counted value on LCD.

Circuit /Logic / Block Diagram:



Equipment's / Components / Apparatus Required:
Keil IDE simulator, Flash Magic or HJTAG software's installed, LPC 2148 datasheet, LPC 2148 kit with power adapter, serial and parallel cable.

Design / Program:
```
#include <LPC214X.H>

#define sw (1 << 4)                  //P0.4 for LED

#define LCDRS       (1 << 13)        //0 for cmd n 1 for data
#define LCDRW       (1 << 12)        //0 for write n 1 for read
#define LCDEN       (1 << 10)        //0 for disable, 1 for normal oprn n 1 to 0 for tx data/cmd

#define LCD_D0 (1 << 15)             //port 0
#define LCD_D1 (1 << 16)
#define LCD_D2 (1 << 17)
#define LCD_D3 (1 << 18)
#define LCD_D4 (1 << 19)
#define LCD_D5 (1 << 20)
#define LCD_D6 (1 << 21)
#define LCD_D7 (1 << 22)

unsigned char digit1[1]="0";
unsigned int _code[3]={0x00070000,0x00008000,0x00400000};
unsigned char _data[8]=" MSRIT ";
void display_data(unsigned char);
void display_code(unsigned int ) ;
```

```
void delay(int sec)
{int i,j;
for(i=0;i<sec;i++)
for(j=0;j<10000;j++);
}
int main(void)
{
unsigned int i=0,xx=0,blink=0x00000001;
IO0DIR|=0xffffffEf&0xffffffff;              // set as o/ps;

PINSEL0=0x00000200&0xffffffff;          //c0nfgure P0.4 as cap0.1 i/p pin f0r c0unter 0
T0PR=0x00000001&0xffffffff;             //0x00003A98&0xffffffff;        //div pclk by 15000 in dec;

T0CTCR=0x05&0xff;                           // for setting counter operation;

T0TCR=0x02;                                 //reset the timer untill TCR[1] bit is 0;
T0TCR=0x01;

for(i=0;i<3;i++)
{
display_code(_code[i]);
}
for(i=0;i<8;i++)
{
display_data(_data[i]);
 }

for(i=0;i<3;i++)
{
display_code(_code[i]);
}
display_data(digit1[0]);
while(1)
{
delay(3);
for(i=0;i<3;i++)
{
display_code(_code[i]);
}
if(T0TC>9)
{
display_data(T0TC+0x37);
}
else
{
display_data(T0TC+0x30);
}
```

```
}
}
  void display_code(unsigned int code)
{
IO0CLR=0xffffffff&0xffffffff;
delay(1);
IO0SET|=code;//0x00070000;

delay(3);
IO0CLR|=LCDRS;
IO0CLR|=LCDRW ;
IO0SET|=LCDEN;
delay(5);
IO0CLR|=LCDEN;
delay(3);
}
void display_data(unsigned char data)
        {
        unsigned int m=0;
        IO0CLR=0xffffffff&0xffffffff;//for R  0x001f8000&0xffffffff;
        delay(5);
        m = data<<15;
        IO0SET|=m&0xffffffff;
        delay(5);
        IO0SET|=LCDRS;
        IO0CLR|=LCDRW ;
        IO0SET|=LCDEN;
        delay(5);
        IO0CLR|=LCDEN;
        delay(5);
        }
```

Procedure for Conduction

- Create a project and choose a simple microcontroller such as NXP's LPC 2148 as your target.
- Create a new C file and attach the source file to the project.
- Create HEX file and flash the HEX file using HJTAG Tool.
- Start the debugger and single step through the code or perform run the entire program at once. Breakpoints can be used for debugging purpose.
- Examine the output.

Observations

Observe the count on the LCD module.

Conclusion from the experiment

Write conclusions based on the observations made while debugging the programs.

Follow Up Questions

The limitation of the program given is that it gives count in hexadecimal. Modify the program to have the decimal count value displayed onto the LCD.

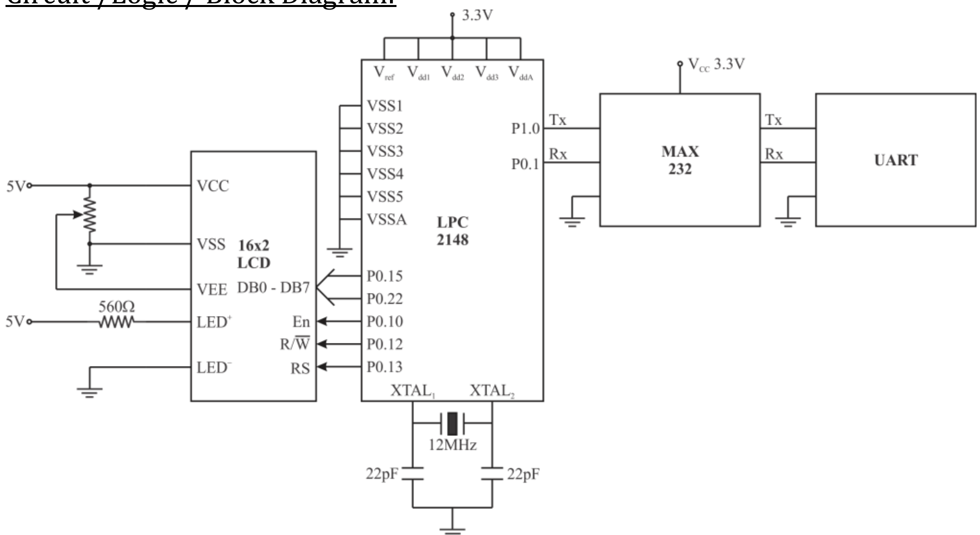
Experiment No. 14 **Date:**

AIM:

- To understand the UART peripheral of LPC 2148.

i) WAP to transmit the given string "MSRIT" using UART functionality of LPC 2148.
ii) WAP to receive a character and display the same on the LCD using UART functionality of LPC 2148.

Circuit /Logic / Block Diagram:



Equipment's / Components / Apparatus Required:

Keil IDE simulator, Flash Magic or HJTAG software's installed, LPC 2148 datasheet, LPC 2148 kit with power adapter, serial and parallel cable.

Design / Program:

i) WAP TO TRANSMIT THE GIVEN STRING "MSRIT" USING UART FUNCTIONALITY OF LPC 2148.

```
#include <LPC214X.H>
#include <stdio.h>

void delay(int count)
{
 int j=0,i=0;

 for(j=0;j<count;j++)
 {
  for(i=0;i<35;i++);
 }
}

void uart0_send(char *data)
{
```

```
        char msg;

while(*data)
        {       while(!(U0LSR & 0x20));
                msg=*data++;
                U0THR = msg; // Send character
        }
}


void main()
{
short x=10;
PINSEL0=0x00000001;//&0xffffffff;

U0LCR=0x83;                                     // 8 bits, no Parity, 1 Stop bit
U0DLL = 97;             //9600 Baud Rate @ 15MHz  PClock
U0LCR = 0x03;                                //DLAB=0

while(x--)
{
uart0_send("MSRIT\n");
delay(100000);
 }
}
```

ii) WAP TO RECEIVE A CHARACTER AND DISPLAY THE SAME ON THE LCD USING UART FUNCTIONALITY OF LPC 2148.

```
#include <LPC214X.H>


#define LCDRS          (1 << 13)        //0 for cmd n 1 for data
#define LCDRW          (1 << 12)        //0 for write n 1 for read
#define LCDEN          (1 << 10)        //0 for disable, 1 for normal oprn n 1 to 0 for tx data/cmd

static lcd_delay(int sec)
{
int i,j;
for(i=0;i<(sec*10);i++)
for(j=0;j<10000;j++);
}

void lcd_display_code(unsigned int code)
        {
        IO0DIR|=0x007FB400;
        IO0CLR=0xffffffff&0xffffffff;
        lcd_delay(1);
        code<<=15;
        IO0SET|=code;//0x00070000;
```

```
        lcd_delay(2);
        IO0CLR|=LCDRS;
        IO0CLR|=LCDRW ;
        IO0SET|=LCDEN;
        lcd_delay(3);
        IO0CLR|=LCDEN;
        lcd_delay(2);
        }


void lcd_display_data(unsigned char data)
        {
        unsigned int m=0;
        IO0DIR|=0x007FB400;// rs,rw &en pins as o/p, 10,12,13 , data pins 15-22
        IO0CLR=0xffffffff&0xffffffff;//for R  0x001f8000&0xffffffff;
        lcd_delay(1);
        m = data<<15;
        IO0SET|=m&0xffffffff;
        lcd_delay(1);
        IO0SET|=LCDRS;
        IO0CLR|=LCDRW ;
        IO0SET|=LCDEN;
        lcd_delay(1);
        IO0CLR|=LCDEN;
        lcd_delay(1);
        }

void uart0_rec() __irq
{
char data;
if(U0LSR & 0x01)
{data=U0RBR ;
 lcd_display_data(data);}
 VICVectAddr   =          0x00;
 }

void main()
{
char data[16]={'E','N','T','E','R',' ','D','A','T','A',' ','I','N',' ','P','C'};
int i,_code[6]={0x0E,0x01,0x80};
PINSEL0=0x00000005&0xffffffff;
U0LCR=0x83;
U0DLL = 97;
U0LCR = 0x03;
U0IER=0x01;
/*************interrupt enable********/
VICIntSelect     = 0<<6;
VICVectAddr0  = (unsigned long)uart0_rec;
VICVectCntl0    = 0x020 | 6 ;
VICIntEnable    = 1<<6;
```

```
/************************************/

for(i=0;i<3;i++)
lcd_display_code(_code[i]);

for(i=0;i<16;i++)
lcd_display_data(data[i])            ;

for(i=0;i<3;i++)
lcd_display_code(_code[i]);

while(1)
 {}
}
```

<u>Procedure for Conduction</u>

- Create a project and choose a simple microcontroller such as NXP's LPC 2148 as your target.
- Create a new C file and attach the source file to the project.
- Create HEX file and flash the HEX file using HJTAG Tool.
- Connect the serial port of the system to UART0 of the kit using serial cable.
- Start the debugger and single step through the code or perform run the entire program at once. Breakpoints can be used for debugging purpose.
- Examine the output in the HyperTerminal of the system.

<u>Observations</u>

Observe the output in the HyperTerminal of the system.

<u>Conclusion from the experiment</u>

Write conclusions based on the observations made while debugging the programs.