COURSE OBJECTIVES

CO1: Write ARM7 assembly language program and simulate using Keil IDE
CO2: Demonstrate interfacing ARM7 with peripheral devices
CO3: Write ARM7 programs in embedded C
CO4: Develop and demonstrate project on Microcontrollers

INSTRUCTIONS TO THE STUDENTS

1. Students should carry the Lab Manual and Observation Book to every lab session.
2. Be on time and follow the institution dress code.
3. You should try to analyze and understand the solved problems and then try to solve all the exercise problems of the experiment in the lab.
4. Maintaining an observation copy is compulsory for all, where in the results of all the problems solved in the lab should be properly noted down.
5. You have to get your results verified and observation copies checked by the instructor before leaving the lab for the day.
6. You should maintain a folder of all the programs you do in the lab in the computer you use by your registration number. You are also advised to keep a back-up of it.
7. Use of external storage media during lab is not allowed.
8. Maintain the timings and the discipline of the lab.

EVALUATION PLAN

- Internal Assessment Marks : 60% (60 marks)

✓ Continuous evaluation component (for each experiment):10 marks
✓ Assessment is based on, preparation, conduction of each experiment, exercise problems, maintaining the observation note and answering the questions related to the experiment.
✓ Total marks of the10 experiments scaled to 60 marks

Note: Follow code of conduct (punctual, discipline, and sincere)

- End semester assessment: 40 %(40 marks)

✓ Write up: 12 marks
✓ Conduction: 12 marks
✓ Results: 8 marks
✓ Viva –Voce: 8 marks

- Mini Project assessment: 100% (100 marks)

✓ Presentation of project idea: 20 marks
✓ Progress presentation: 30 marks
✓ Report: 20 marks
✓ Working project demonstration: 30 marks

Final grading will be the average of Mini-project (100 marks) and Regular labs (100 marks)

Note: (i) Best projects will be selected from each batch for project exhibition
(ii) Three best projects will be awarded.

Expt. No. 1
INTRODUCTION TO ARM7 ASSEMBLY LANGUAGE PROGRAMMING (ADDRESSING MODES)

Aim: To understand basics of ARM7 programming and simulation

1.To get familiar with ARM7 programming

```
01    AREA PROGRAM1, CODE, READONLY; AREA CAN NOT START FROM FIRST COLUMN
02  ENTRY
03      MOV R1,#0XFFFFFFFF  ; MOV IMMEDIATE DATA TO R0
04      MOV R0,#8_257    ;MOV IMMEDIATE DATA TO R1
05      MOV R2,#82       ;
06      MOV R3,#2_1011101
07      MOV R4,R1
08      MOV R5,R2
09      END
```

Note: For the above program try following with immediate data:

a) line3: 0Xffff, line4: 8_25765, line5:825 and conclude
b) line3: 0x3E8, line4: 8_275, line5:16000 and conclude
c) line3: 0xC000003F, line4: 8_275, line5:16000 and conclude
d) Replace MVN instruction with MOV.

2. Program to use EQU, and Access address/data from data / code Memory

```
01  AD1   EQU 0xAB
02  NUM2  EQU 247
03  NUM3  EQU AD1+1
04   AREA PROGRAM1, CODE, READONLY
05  ENTRY
06      MOV R0,#AD1   ;?
07      MOV R1,#NUM2  ;?
08      MOV R2,#NUM3  ;?
09      LDR R0,Q      ;?
10      LDR R1,=Q     ;
11      ADR R3,P      ;?
12      LDR R5,=R     ;?
13      ;LDR R6,R
14  s   B    s
15  Q   DCD 0xAABBCCDD
16  P   DCD 0xABCDEF78
17
18    AREA    MEMORY, DATA, READWRITE ;?
19  VALU DCD 0X11223344
20  R   DCD 0XABBCCDDE
21  T   DCD 0X11223344
22
23  RES SPACE 10 ; RESERVE 10 LOCATIONS
24      end
```

Note: For the above program try following:

a. Remove semicolon in line13, and check.
b. Remove semicolon in line13 and use ADR instead of LDR, and check
c. Use ADR in line 12 instead of LDR, and check

3.Program to read from memory and write to memory

```
01   AREA PROGRAM1, CODE, READONLY
02  ENTRY
03      ADR R0,Q     ; ?
04      LDR R1,[R0] ; ?
05      ldrb r2,[r0]
06      ldrh r3,[r0]
07      LDR R4,=RES ;?
08      STR R1,[r4]
09      strb r2,[r4,#4]
10      strh r3,[r4,#8]
11      LDR R6,=VALU ; Specify desired value at 0x40000000
12                   ; in memory window during runtime
13      LDR R7,[R6]
14      mov r8,#4;
15      ldr r9,[r0,r8]
16  s   B   s
17  Q   DCD 0xAABBCCDD
18  P   DCD 0xABCDEF78
19
20    AREA   MEMORY, DATA, READWRITE ;?
21  VALU DCD 0X11223344 ; these values never get
22  R   DCD 0XABBCCDDE  ; initialized in data memory
23  T   DCD 0X11223344
24
25  RES SPACE 10 ; RESERVE 10 LOCATIONS
26      end
```

Note: For the above program try following:

a) Line11: write 8 instead of 5 and check

4.Program on pre and post increments (Use memory window with address 0 for during LDR and use address 0x40000000 during STR)

```
01   AREA PROGRAM1, CODE, READONLY
02  ENTRY
03      ADR R0,Q          ; ?
04      LDR R1,[R0],#4    ; ?
05      ldrb r2,[r0]      ;?
06      LDR R3,[R0,#4]!   ;?
07      ldrh r4,[r0]      ;?
08      LDR R5,=R         ;?
09      STR R0,[r5],#4    ;?
10      strb r1,[r5]     ; ?
11      strh r2,[r5,#4]!;?
12      MOV R6,#8         ;?
13      STRB R1,[R5], R6;?
14      STRH R2,[R5,R6]!;?
15      STR R3,[R5]       ;?
16  s   B   s
17  Q   DCB "MANIPAL INSTITUTE OF TECHNOLOGY, MANIPAL"
18  P   DCD 0xABCDEF78
19
20    AREA   MEMORY, DATA, READWRITE ;?
21  VALU DCD 0X11223344 ; these values never get
22  R   DCD 0XABBCCDDE  ; initialized in data memory
23  T   DCD 0X11223344
24  RES SPACE 10 ; RESERVE 10 LOCATIONS
25      end
```

Note: Use **LTORG** between line8 and 9 and check the difference.

Exercise:

Write a program to read one byte, one half-word and on word at data memory locations starting from 0x40000000 (enter during runtime) and store these data back at data memory locations starting from 0x4000001C.

5.Program on data transfer with shift/rotate

```
 AREA PROGRAM1, CODE, READONLY
ENTRY
    ADR R0,Q          ; ?
    LDR R1,=VAL
    LDRB R3,[R0]
    MOV R4, R3, LSL#2
    MOV R4, R4, LSR#1
    LDR R5,[R0],R4,LSL#2
    STRB R5,[R1]
S   B S
Q   DCB 1
  AREA   MEMORY, DATA, READWRITE ;?
VAL DCD 0X11223344
    End
```

Note:

a. Use ROR, RRX, and ASR.
b. Can rotate left instruction be used?

6.Program to understand SWP instruction

```
 AREA PROGRAM1, CODE, READONLY
ENTRY
    ADR R0,Q          ; ?
    LDR R1,=VAL
UP  LDRB R3,[R0],#1
    CMP R3,#0
    BEQ DN
    ADDS R2,#1
    STRB R3,[R1],#1
    B UP
DN  LDR R0,=VAL
UP1 LDRB R3,[R0]
    SWPB R3,R3,[R1]
    SWPB R3,R3,[R0]
    ADD R1,#1
    ADD R0,#1
    SUBS R2,#1
    BNE UP1
S  B S
Q   DCB "ABCDEFGH",0

  AREA   MEMORY, DATA, READWRITE ;?
VAL DCD 0X11223344
    END
```

7. Program for Multi-word data transfer

```
 AREA  ASCENDING , CODE, READONLY
ENTRY
    LDR R12,=SOURCE
    LDR R13,=DEST
    MOV R14,#12
    LDMIA R12!,{R0-R11}
    STMIA R13!,{R0-R11}
    LDMIA R12!,{R0-R2}
    STMIA R13!,{R0-R2}
;LOAD/STORE 12 WORDS TO/FROM REGISTERS FROM/TO MEMORY
; CAN BE MEADE TO LOAD/STORE MULTIPLE OF 12 WORDS
S   B   S
SOURCE DCD  1,2,3,4,5,6,7,8,9,0XA,0XB,0XC,0XD,0XE,0XF

 AREA DATA1,DATA,READWRITE
DEST    DCW 0X0045
    END
```

8. Program to find square of a decimal number (0 to 10) using lookup table

```
 AREA  SQUARE , CODE, READONLY
ENTRY               ;Mark first instruction to execute
START
    LDR R0,=TABLE1  ; Load start address of Lookup table
    LDR R1,=INP     ; Initialize value at 0x40000000
    MOV R2,R1
    LDR R1,[R1]     ; Load no whose square is to be find
    MOV R1,R1,LSL#0x2; R1=Value at INPx4. to point at corresponding square
    ADD R0,R0,R1    ; Load address of element in Lookup table
    LDR R3, [R0]    ; Get square of given no in R3
    STR R3,[R2,#4]!
S   B   S
;Lookup table contains Squares of nos from 0 to 10
TABLE1  DCD 0,1,4,9,0x16,0x25,0x36,0x49,0x64,0x81,0x100;
 AREA DATA1,DATA,READWRITE
INP DCD 0X0045
    END             ; Mark end of file
```

Expt. No. 2
INTRODUCTION TO ARM7 ASSEMBLY LANGUAGE PROGRAMMING (DATA PROCESSING-I)

Aim: To understand data processing

1.Program to check whether the number is even or odd.

```
 AREA PROGRAM1, CODE, READONLY
ENTRY
    MOV R0,#NUMBER ; READ NUMBER FROM MEMORY
    MOVS R1,R0,LSR #1  ;SHIFT LSB TO CARRY
    MOVCS R2,#0XFFFFFFFF; IF EVEN
    LDRCC R2,=0X55555555; ELSE ODD
S   B S
    END
```

Note: Try with LSL, ASR ROR and RRX

2.Program to logical operations

```
 AREA PROGRAM1, CODE, READONLY
ENTRY
    LDR R0,NUMBER1 ; READ NUMBER FROM MEMORY
    LDR R1,NUMBER2; READ NUMBER FROM MEMORY
    AND R2,R0,R1
    EOR R3,R0,R1
    ORR R4,R0,R1
S   B   S
```

Exercise:

Write a program to perform logical operations on two 48 bit numbers

3.Program to sort in ascending order

```
 AREA  ASCENDING , CODE, READONLY
ENTRY
        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 TO 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
S       B   S
```

```
28  CVALUE   DCD 0X44444444       ;DATA
29           DCD 0X11111111
30           DCD 0X33333333
31           DCD 0X22222222
32       AREA DATA1,DATA,READWRITE
33  DVALUE   DCD 0X00000000       ;STORE RESULT
34       END
```

4.Program to reverse half-word

```
01   AREA  ASCENDING , CODE, READONLY
02  ENTRY
03      LDR R0,=NUMBER
04      LDR R1,[R0]
05      AND R2,R1,#0XF
06      MOV R2,R2,LSL#12
07      AND R3,R1,#0XF0;
08      MOV R3,R3,LSL#4
09      orr r2,r3
10      AND R4,R1,#0XF00
11      MOV R4,R4,LSR#4
12      orr r2,r4
13      AND R5,R1,#0XF000;
14      MOV R5,R5,LSR#12
15      orr r2,r5
16  s   B   s
17
18   AREA DATA1,DATA,READWRITE
19  NUMBER   DCW 0X0045
20       END
```

Exercise:

1. Write a program to check whether or not read number is positive.
2. Write a program to separate an array ten numbers into even array and odd array.
3. Write a program to count number of 1's present in a word.
4. Write a program to arrange N half-words in descending order.

5. Write a program to verify that word read from memory is nibble-wise palindrome (ex: 0x1221 is nibble-wise palindrome)
6. Write a program to verify that byte read from memory is bit-wise palindrome (ex: b_10111101 is bit-wise palindrome)

Expt. No. 3
INTRODUCTION TO ARM7 ASSEMBLY LANGUAGE PROGRAMMING (DATA PROCESSING-II)

Aim: To understand data processing (Arithmetic operations)

1.Program with different arithmetic instructions.

```
  AREA PROGRAM1, CODE, READONLY;
ENTRY
        LDR R1,NUM1
        LDR R2,NUM2
        LDR R0,=RES
        ADD R3,R1,R2     ;R3=R1+R2
        STR R3,[R0],#4
        ADC R3,R1,R2     ;R3=R1+R2+C
        STR R3,[R0],#4
        SUB R3,R1,R2   ;R3=R1-R2
        STR R3,[R0],#4
        SBC R3,R1,R2     ;R3=R1-R2+C
        STR R3,[R0],#4
        RSB R3,R1,R2   ;R3=R2-R1
        STR R3,[R0],#4
        RSC R3,R1,R2   ;R3=R2-R1+C-1
        STR R3,[R0],#4
        BIC R3,R1,R2 ;R3=R1 AND (~R2)
        STR R3,[R0]
S       B S
NUM1 DCD 0XF7654321
NUM2 DCD 0XF2345678
    AREA   MEMORY, DATA, READWRITE ;?
RES DCD 0        ; starting address is 0x40000000
    END
```

Note: Include S at the end in ADD, ADC, SUB, SBC,RSB, RSC and BIC of the above program and observe the CPSR register and data memory.

2.Program to add two 16-bit numbers

```
;/* PROGRAM TO ADD two 16-BIT NUMBERS & STORE IN INTERNAL RAM       */
;/* THE RESULT CAN BE VIEWED IN LOCATION 0X40000000 & ALSO IN R0    */

    AREA  ADDITION , CODE, READONLY
ENTRY               ;Mark first instruction to execute
START
    MOV R0,#0       ; INTIALISE SUM TO ZERO
    LDR R1, VALUE1      ; LOADS THE FIRST VALUE
    LDR R3,MASK     ; MASK TO GET 16 BIT
    AND R1,R1,R3        ; MASK MSB
    ADD R0,R0,R1        ; ADD THE ELEMENTS
    LDR R2, VALUE2
    AND R2,R2,R3
    ADD R0,R0,R2
    LDR R4,=RESULT      ; LOADS THE ADDRESS OF RESULT
    STR R0,[R4]     ; STORES THE RESULT IN R1
here b here
MASK DCD 0X0000FFFF     ;  MASK MSB
VALUE1 DCW  0XAAAA
 ALIGN
VALUE2 DCW 0X2222

 AREA DATA2,DATA,READWRITE      ; TO STORE RESULT IN GIVEN ADDRESS
RESULT DCD 0X0, 0x0

    END         ; Mark end of file
```

3. Program to add two 64 bit numbers

```
;/* PROGRAM TO ADD two 64-BIT NUMBERS & STORE IN INTERNAL RAM       */
;/* THE RESULT CAN BE VIEWED IN LOCATION 0X40000000 onward      */
    AREA  ADDITION , CODE, READONLY
ENTRY               ;Mark first instruction to execute
START
    LDR R0, VALUE1+4     ; LOADS THE LEAST SIGNIFICANT PART OF FIRST VALUE
    LDR R2, VALUE2+4     ; LOADS THE LEAST SIGNIFICANT PART OF SECOND VALUE
    LDR R7, =RESULT      ; LOADS THE ADDRESS OF RESULT STORAGE LOCATION
    ADD R7, R7, #8       ; ADJUST RESULT ADDRESS FOR LITTLE ENDIAN FORMAT
    ADDS R4, R0, R2      ; ADD THE LSB PART OF THE NUMBERS
    STR R4, [R7]         ; STORES THE RESULT IN MEMORY
    SUB R7, R7, #4       ; ADJUST RESULT ADDRESS FOR LITTLE ENDIAN FORMAT
    LDR R1, VALUE1       ; LOADS THE MOST SIGNIFICANT PART OF FIRST VALUE
    LDR R3, VALUE2       ; LOADS THE MOST SIGNIFICANT PART OF SECOND VALUE
    ADCS R5, R1, R3      ; ADD THE MOST PART OF THE NUMBERS
    STR R5, [R7]         ; STORES THE RESULT IN MEMORY
    SUB R7, R7, #4
    MOV R6, #0       ; STORING THE END AROUND CARRY, IF ANY
    ADC r6, r6, #0
    STR R6, [R7]
here b here
VALUE1 DCD  0XFFA2E640, 0xF2100123    ; NUMBER IS VALUE1 = 0XFFA2E640F2100123
VALUE2 DCD 0XAA1019BF, 0x40023F51     ; NUMBER IS VALUE2 = 0XFFA2E640F2100123
    AREA DATA2,DATA,READWRITE     ; TO STORE RESULT IN GIVEN ADDRESS
RESULT DCD 0X0, 0x0, 0x0

    END              ; Mark end of file
```

4. Program to add array 16 bit unsigned array

```
; PROGRAM TO ADD an array of unsigned 16-BIT NUMBERS & STORE IN INTERNAL RAM
; THE RESULT CAN BE VIEWED IN LOCATION 0X40000000 & ALSO IN R0
    AREA  ADDITION , CODE, READONLY
ENTRY               ;Mark first instruction to execute
START
    MOV R0, #0           ; INTIALISE SUM TO ZERO
    mov r5, #10          ; number of half words to add
    LDR R6, =num_array          ; points to the array
next_element
    LDRH R1, [r6]            ; read the number from array into r1

    ADD R0, R0, R1           ; ADD THE ELEMENTS
    add r6, r6, #2           ; update pointer for next element
    subs r5, r5, #1          ; reduce the count of number of elements in the array
    bne next_element

    LDR R4, =RESULT          ; LOADS THE ADDRESS OF RESULT
    STR R0, [R4]             ; STORES THE RESULT IN R1

here b here

num_array DCW   0X1111, 0X2222, 0x3333, 0x4444, 0x5555, 0x6666, 0XAAAA, 0XBBBB, 0xCCCC,

    AREA DATA2, DATA, READWRITE      ; TO STORE RESULT IN GIVEN ADDRESS
RESULT DCD 0X0, 0x0
    END              ; Mark end of file
```

Exercise:

a) Write a program add two four digit Binary Coded Decimal numbers.

5. Program to simulate subtraction

```
; trial of SUB, SBC, RSB instruction
 area Program, CODE, READONLY
ENTRY
Main
    LDRB  R1, x      ; load a byte into r1
    LDRH  r2, y      ; load a word into r2
    LDR r4, =Result
    sub r3, r1, r2   ; normal subtraction, r3 = r1 = r2
    strh r3, [r4]    ; store half word result at location pointed by r4
    add r4, r4, #4
    rsb  R3, R1, r2  ; reverse subtract- r2 - r1 = r3
    strh r3, [r4]    ; store half word result at location pointed by r4
    add r4, r4, #4
    sbc r3, r1, r2   ; sbc r3 = r1-r2-!Carry
    strh r3, [r4]    ; store half word result at location pointed by r4
    add r4, r4, #4
stop b stop
x   DCW     -0X0001  ; Value to be tried
y   DCW     -0X0001  ; Value to be tried
;   try with same values of x and y and analyse results

 AREA Example1_data, data, READWRITE
ALIGN            ;Need to do this because working with a series of 16-bit data
Result  DCW     0  ;Storage space
 END
```

b) Write a program to demonstrate reverse subtraction of two 16 bit numbers with carry

6.Program to multiply two 32 bit numbers

```
;/* THE RESULT CAN BE VIEWED IN LOCATION 0X40000000 & ALSO IN R0
    AREA  multiply , CODE, READONLY
ENTRY                ;Mark first instruction to execute
START
    LDR R1, x        ; LOADS THE FIRST VALUE
    LDR R2, y        ; LOADS THE FIRST VALUE
    LDR R4,=RESULT       ; LOADS THE ADDRESS OF RESULT
     mul    R0, R1, R2   ; Multiply the numbers
    STR R0,[R4]      ; STORES THE RESULT IN R0
    umull R0, R3, R1, R2     ; Multiply THE numbers
    add R4, R4, #4       ; LOADS THE ADDRESS OF RESULT
    STR R0,[R4]      ; STORES THE RESULT IN R0
    add R4, R4, #4       ; LOADS THE ADDRESS OF RESULT
    STR R3,[R4]      ; STORES THE RESULT IN R3
    smull R0, R3, R1, R2     ; Multiply THE numbers
    add R4, R4, #4       ; LOADS THE ADDRESS OF RESULT
    STR R0,[R4]      ; STORES THE RESULT IN R0
    add R4, R4, #4       ; LOADS THE ADDRESS OF RESULT
    STR R3,[R4]      ; STORES THE RESULT IN R3
here b here
x DCD -0x1
y DCD  0X2
; give different set of values for x and y and analyse the results
 AREA DATA2, DATA, READWRITE  ; TO STORE RESULT IN GIVEN ADDRESS
RESULT DCD 0X0
    END
```

c) Write a program to multiply two 16 bit signed numbers.

d) Write a program to multiply two 64 bit numbers

Expt. No. 4

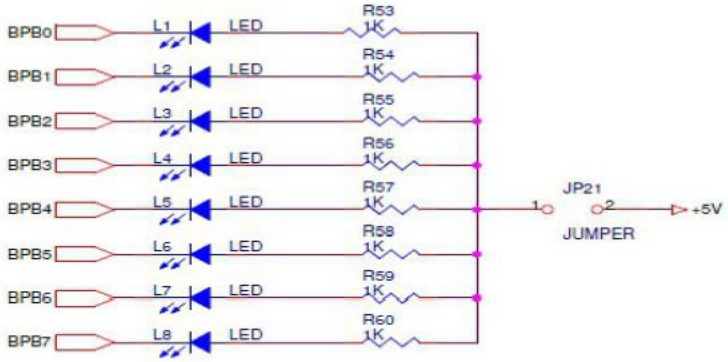
ASSEMBLY LANGUAGE PROGRAMMING GPIOs TO INTERFACE LEDS, SWITCHES BUZZER AND RELAY

Aim: To program general purpose input output pins of Arm7 to interface switches, LEDs and Buzzer

1.**Program to display ring count on (P0.16-P0.23 pins) LEDs (Active low)**

```
IO0DIR EQU 0xE0028008 ;Address for configuring Port0 as I/O
IO0SET EQU 0xE0028004 ;Register address for setting Port0 pins
IO0CLR EQU 0xE002800C ;Register address for clearing Port0 pins
    AREA CHANG, CODE,READONLY
    EXPORT __main  ;refer the line no 416 of startup.s file
__main
    ENTRY
        LDR r1,=IO0DIR ; load the address of the IODIR reg to R1
        LDR r0,=0x00FF0000; To set pins P0.16 to P0.23 as output pins
        STR r0,[r1]       ;This configure P0.16 to P0.23 as output pins
        LDR r2,=IO0CLR ; load the address of the IOCLR reg to R2
        LDR r3,=IO0SET ; load the address of the IOSET reg to R3
repeat  MOV r4, #0x00010000 ; write a control word to set one bit
next    STR r4, [r2]
        LDR r5, =0x2FFFFF ; Delay program to retain the bit for some time.
delay   SUBS r5, r5, #1
        BNE delay
        STR r0, [r3] ; set all pins
        MOV r4, r4, LSL #1
        CMP r4, #0x1000000
        BNE next
        B repeat
        END
```

Light Emitting Diodes (LED's) are components most commonly used for displaying the port line status. There are 8 LEDs on the board; these lines are connected to the Port lines P0.16 (PB0) to P0.23 (PB7) through buffer.



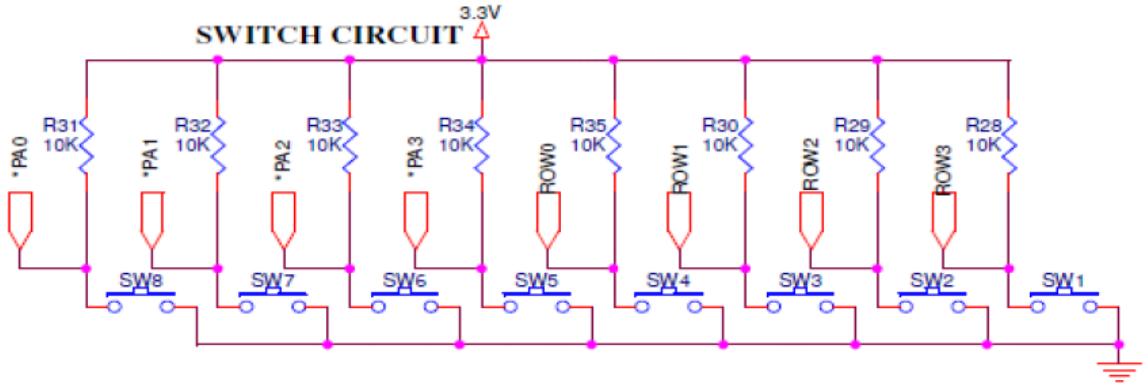
Exercise:

a) Write an Assembly Language Program (ALP) to display 8-bit Johnson's count.
b) Write an ALP to display Mod-16 up count on LED circuit.
c) Write an ALP to display 8-bit BCD up count on LED circuit.

2.**Program to interface switch circuit and LEDs**

```
;PROGRAM TO READ FROM PORT1(P1.16-1.23) AND DISPLAY THE SAME ON PORT( P0.16-0.23
IO0DIR EQU 0xE0028008 ;Address for configuring Port0 as I/O
IO0SET EQU 0xE0028004 ;Register address for setting Port0 pins
IO0CLR EQU 0xE002800C ;Register address for clearing Port0 pins
IO1DIR EQU 0xE0028018 ; UPON RESET CONFIGURED AS INPUT
IO1PIN EQU 0xE0028010
 AREA CHANG, CODE,READONLY
    EXPORT __main  ;refer the line no 416 of startup.s file
__main
    ENTRY
        LDR r1,=IO0DIR ; load the address of the IODIR reg to R1
        LDR r0,=0x00FF0000; To set pins P0.16 to P0.23 as output pins
        STR r0,[r1]       ;This configure P0.16 to P0.23 as output pins
        LDR r2,=IO0CLR ; load the address of the IOCLR reg to R2
        LDR r3,=IO0SET ; load the address of the IOSET reg to R3
        LDR R4,=IO1PIN; TO READ INPUT
        LDR r1,=IO1DIR
        str R0,[R3]
repeat  LDR R5,[R4]     ;READ A SWITCH
        AND R5,#0XFF0000 ; RETAIN ONLY INPUT PORT CONTENT
        CMP R5,#0XFF0000 ; CHECK IF SWITCH PRESSED
        BEQ repeat
        str R0,[R3]     ;IF TRUE TURN-OFF LEDs
        EOR R5,#0XFF0000;CLEAR THE BIT CORRESPONDIG TO SWITCH PRESSED
        str R5,[R2]     ; DISPLAY CORRESPONDING LED
        B repeat
        END
```

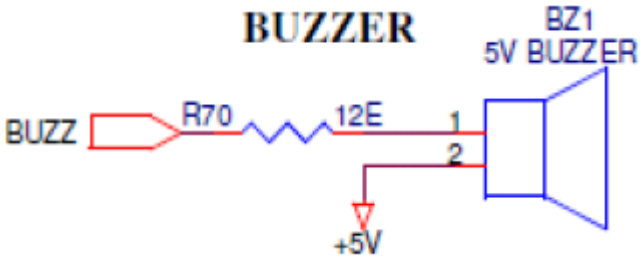
The switches **SW1** to **SW8** are organized. One end of all the switches are connected to port lines P1.16 – P1.23 and other ends are Ground.



Exercise:

d) Write a program to interface with switches to perform following operations:
If SW8 is pressed, turn on, one LED, if SW7 is pressed, turn on, 2 LEDs, if SW6 is pressed, turn on, 4 LEDs and turn on, 8 LEDs if SW5 is pressed.

3.Program to interface Buzzer

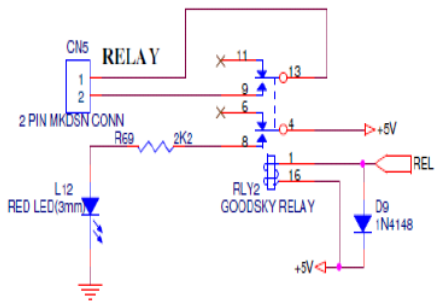


```
IO0DIR EQU 0xE0028008 ;Address for configuring Port0 as I/O
IO0SET EQU 0xE0028004 ;Register address for setting Port0 pins
IO0CLR EQU 0xE002800C ;Register address for clearing Port0 pins
 AREA CHANG, CODE,READONLY
    EXPORT __main  ;refer the line no 416 of startup.s file
__main
    ENTRY
        LDR r1,=IO0DIR ; load the address of the IODIR reg to R1
        LDR r0,=0x200; To set P0.9 as output pin for buzzer
        STR r0,[r1]       ;This configures P0.9 as output pin
        LDR r2,=IO0CLR ; load the address of the IOCLR reg to R2
        LDR r3,=IO0SET ; load the address of the IOSET reg to R3
S       STR R0,[R3];
        LDR R4,=0x2FF; Keep varying this value and hear the buzzer
delay   SUBS R4,R4,#1
        BNE delay
        STR R0,[R2]
        LDR R4,=0x2FFFF;Keep varying this value
delay1  SUBS R4,R4,#1
        BNE delay1
        B   S
        END
```

Exercise: e)Write a program to interface Buzzer to generate different tones on buzzer

4.**Program to interface relay (P0.10 pin is connected to relay)**

```
IO0DIR EQU 0xE0028008 ;Address for configuring Port0 as I/O
IO0SET EQU 0xE0028004 ;Register address for setting Port0 pins
IO0CLR EQU 0xE002800C ;Register address for clearing Port0 pins
 AREA CHANG, CODE,READONLY
    EXPORT __main  ;refer the line no 416 of startup.s file
__main
    ENTRY
        LDR r1,=IO0DIR ; load the address of the IODIR reg to R1
        LDR r0,=0x400; To set P0.10 as output pin for relay
        STR r0,[r1]       ;This configure P0.10 as output pin
        LDR r2,=IO0CLR ; load the address of the IOCLR reg to R2
        LDR r3,=IO0SET ; load the address of the IOSET reg to R3
BACK    STR R0,[R2]    ; Relay Open (Turn off LED)
        LDR R4,=0x2FFFFF
delay   SUBS R4,R4,#1
        BNE delay
        STR R0,[R3]    ;Relay Closed (Turn on LED)
        LDR R4,=0x5FFFF
delay1  SUBS R4,R4,#1
        BNE delay1
        B BACK
        END
```



Expt. No. 5
INTERFACING STEPPER MOTOR AND DC MOTOR

Aim: Assembly language programming to interface Stepper and DC motors

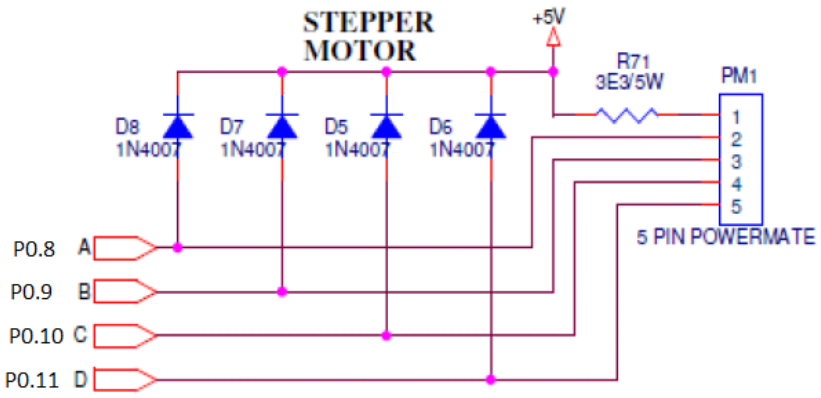
1. **Program to rotate stepper motor anticlockwise (Full stepping with single coil energized at a time). It is full stepping (step angle 1.8º per step). Note: Short JP13**

```
IO0DIR EQU 0xE0028008 ;Address for configuring Port0 as I/O
IO0SET EQU 0xE0028004 ;Register address for setting Port0 pins
IO0CLR EQU 0xE002800C ;Register address for clearing Port0 pins
 AREA CHANG, CODE,READONLY
    EXPORT __main  ;refer the line no 416 of startup.s file
__main
    ENTRY
        LDR r1,=IO0DIR ; load the address of the IODIR reg to R1
        LDR r0,=0xF000; To set P0.9 as output pin for buzzer
        STR r0,[r1]       ;This configure P0.12-P0.15 as output pins
        LDR r2,=IO0CLR ; load the address of the IOCLR reg to R2
        LDR r3,=IO0SET ; load the address of the IOSET reg to R3
S       LDR R5,=0X8000 ; Energise single coil at a time
BACK    STR R0,[R3]    ; Set all 4 pins to 1 to turnoff motor
        STR R5,[R2]    ; Clears one bit at a time to enable a coil(Active low)
        LDR R4,=0x2FFF; Keep varying this value for different speed
delay   SUBS R4,R4,#1;
        BNE delay
        MOV R5,R5,LSR #1; shift for energising next coil
        CMP R5,0X800    ; Are all coils energised?
        BNE BACK
        B S
        END
```

Note:

i. Full stepping can be achieved by energizing two adjacent coils at a time(step angle 1.8º per step).

ii. Half stepping can be achieved by alternately energizing two coils at a time and then one coil at a time (step angle 0.9º per step)



Exercise:

a) Write an alp to rotate the stepper motor clockwise to have step angle of 0.9º per step.

b) Write an alp to rotate stepper motor anticlockwise for 180º.

2. **Program to interface DC motor.**

The DC motor can be interfaced to the board by connecting it to the Reliamate RM4. The direction of the rotation can be changed through the program using relay, RLY1. Port lines used are P0.8 and P0.11.

```
05      EXPORT __main  ;refer the line no 416 of startup.s file
06  __main
07      ENTRY
08          LDR r1,=IO0DIR ; load the address of the IODIR reg to R1
09          LDR r0,=0x900; To set P0.8 AND P0.11 as output pins for the motor
10          STR r0,[r1]        ;This configure P0.8and P0.11 as output pins
11          LDR r2,=IO0CLR ; load the address of the IOCLR reg to R2
12          LDR r3,=IO0SET ; load the address of the IOSET reg to R3
13          ;LDR R5,=0X800 ; TO SWITCH DIRECTION
14          LDR R6,=0X100 ; TO TURN ON OR OFF THE MOTOR
15          STR R0,[R2] ; keep the motor off
16  BACK    STR R6,[R3]    ;TURN ON MOTOR ANTICLOCKWISE DIRECTION
17          BL delay
18          STR R0,[R2]    ;TURN OFF MOTOR
19          BL delay
20          STR R0,[R3]; Motor to rotate clockwise
21          BL delay
22          STR R0,[R2] ;TURN OFF MOTOR
23          BL delay
24          B BACK
25  delay   ;subroutine
26          LDR R4,=0x2FFFFF
27  up      SUBS R4,R4,#1;
28          BNE up
29          BX LR
30          END
```

Exercise:

c)Write an alp to control the speed of the DC motor (for some time low speed , then for some time medium speed , high speed and repeat)

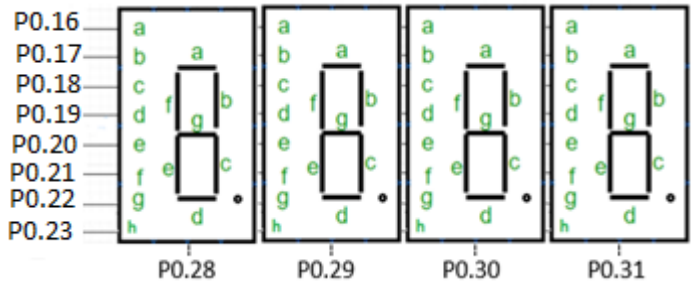
Expt. No. 6

INTERFACING SEVEN SEGMENT LEDS AND HEX KEYPAD

Aim: Assembly language programming to interface Seven segment LEDs and Hex keypad.

1.**Program to interface Seven Segment Display (Short Jumper JP2 (right two pins))**

```
    ENTRY
        LDR r1,=IO0DIR ; load the address of the IODIR reg to R1
        LDR r0,=0xF0FF0000;  pins P0.16 to P0.23 and P0.28-P0.31 as output pins
        STR r0,[r1]        ;This configure P0.16 to P0.23 as output pins
        LDR r2,=IO0SET   ; load the address of the IOCLR reg to R2
        LDR r3,=IO0CLR    ; load the address of th
S       STR R0,[R3]      ; Turn off display
        LDR R1,=0X803F0000 ; To display 0 on rightmost segment
        STR R1,[R2]
        BL DELAY
        STR R0,[R3]      ; Turn off display
        LDR R1,=0X40060000;To display 1 on third segment
        STR R1,[R2]
        BL DELAY
        STR R0,[R3]      ; Turn off display
        LDR R1,=0X205B0000;To display 2 on second segment
        STR R1,[R2]
        BL DELAY
        STR R0,[R3]      ; Turn off display
        LDR R1,=0X104F0000;To display 3 on leftmost segment
        STR R1,[R2]
        BL DELAY
        B   S
DELAY
        LDR r5, =0x2FF ; Delay program to retain the bit for some time.
delay   SUBS r5, r5, #1
        BNE delay
        BX LR
```



a) Write an ALP to display Characters of your choice (displayable) on Seven segment display.

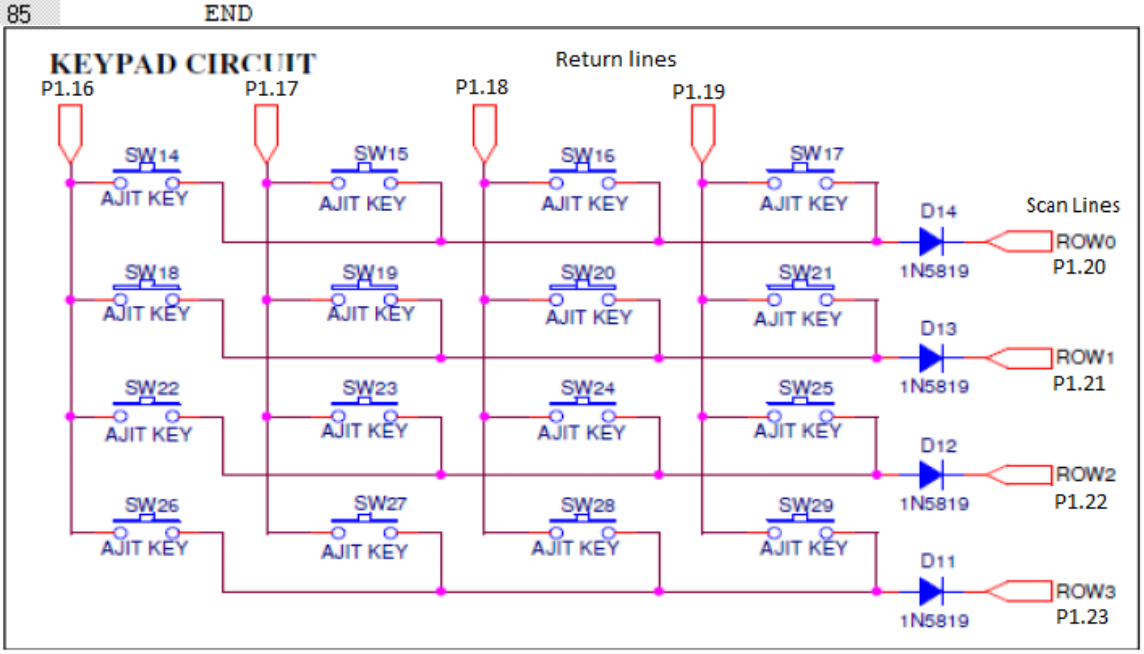
b) Write a program to display ring count on seven segment display unit (as if 4 bit ring)

2. **Program to interface hex keypad and seven segment display. Value of the key pressed displayed on the seven segment display**

```
 INCLUDE VICVPB.S
 AREA key_7, CODE,READONLY
    EXPORT __main  ;refer the line no 416 of startup.s file
__main
    ENTRY
    LDR r1,=IO0DIR ; load the address of the IODIR reg to R1
    LDR r0,=0x10FF0000;  pins P0.16 to P0.23 and P0.28 as output pins
    STR r0,[r1]        ;This configure P0.16 to P0.23 as output pins
    LDR r1,=IO1DIR ; load the address of the IODIR reg to R1
    LDR r0,=0x0F00000; configure p1.20-p1.23 as output(scan lines)p1.16-p1.19 as input
    STR R0,[R1]
```

```
12      LDR R1,=IO1PIN
13      LDR R2,=IO0PIN
14      LDR R4,=IO1CLR
15  BK1 LDR r0,=0x0F00000
16      STR R0,[R4]; CLEAR SCAN LINES TO CHECK THAT NO KEYS ARE RELEASED
17      LDR R5,[R1]; READ FROM KEYBOARD
18      AND R5,R5,#0X0F0000; RETAIN ONLY RETURN VALUE
19      CMP R5,#0XF0000    ; IF KEY STILL PRESSED, THEN R5 NOT EQUAL TO 0XF0000
20      BNE BK1 ; CHECK TILL KEYS ARE RELEASED
21  BK2 BL DELAY ; WAIT FOR SOME TIME
22      LDR R5,[R1] ;READ FROM KEYBOARD
23      AND R5,R5,#0XF0000; RETAIN ONLY RETURN VALUE
24      CMP R5,#0XF0000 ;IF KEY  PRESSED, THEN R5 NOT EQUAL TO 0XF0000
25      BEQ BK2 ; IF KEY IS NOT PRESSED THE GO BACK TO BK2
26      BL DELAY
27      LDR R5,[R1] ;READ INPUT
28      AND R5,R5,#0XF0000; RETAIN ONLY RETURN VALUE
29      CMP R5,#0XF0000
30      BEQ BK2 ; CHECK AGAIN FOR  KEY PRESSED
31      LDR R0,=0XE00000; NOW APPLY 0 ONLY TO ROW0 THROUGH P1.20
32      STR R0,[R1]
33      LDR R5,[R1]  ;READ FROM RETURN LINES
34      AND R5,R5,#0XF0000; RETAIN ONLY RETURN VALUE
35      CMP R5,#0XF0000    ; CHECK WHETHER KEY FOR CORRESPONDING ROW0 IS PRESSED OR NOT
36      BNE ROW0
37      LDR R0,=0XD00000;NOW APPLY 0 ONLY TO ROW1 THROUGH P1.21
38      STR R0,[R1]
39      LDR R5,[R1]    ;READ FROM RETURN LINES
40      AND R5,R5,#0XF0000; RETAIN ONLY RETURN VALUE
41      CMP R5,#0XF0000 ;CHECK WHETHER KEY FOR CORRESPONDING ROW0 IS PRESSED OR NOT
42      BNE ROW1
43      LDR R0,=0XB00000;NOW APPLY 0 ONLY TO ROW1 THROUGH P1.22
44      STR R0,[R1]
45      LDR R5,[R1]
46      AND R5,R5,#0XF0000; RETAIN ONLY RETURN VALUE
47      CMP R5,#0XF0000
48      BNE ROW2
49      LDR R0,=0X700000;NOW APPLY 0 ONLY TO ROW1 THROUGH P1.23
50      STR R0,[R1]
51      LDR R5,[R1]
52      AND R5,R5,#0XF0000; RETAIN ONLY RETURN VALUE
53      CMP R5,#0XF0000
54      BNE ROW3
55      B BK2
56  ROW0 LDR R6,=SEG_CODE0; STARTING ADDRESS SEVEN SEGMENT CODE FOR ROW0
57       B FIND
58  ROW1 LDR R6,=SEG_CODE1;STARTING ADDRESS SEVEN SEGMENT CODE FOR ROW1
59       B FIND
60  ROW2 LDR R6,=SEG_CODE2 ;STARTING ADDRESS SEVEN SEGMENT CODE FOR ROW3
61       B FIND
62  ROW3 LDR R6,=SEG_CODE3;STARTING ADDRESS SEVEN SEGMENT CODE FOR ROW0
63  FIND MOV R8,#16        ;SHIFTIN TO LEFT TO CHECK FOR CARRY FLAG
64  NXT MOVS R7,R5,LSL R8
65      BCC MATCH          ; CHECK TILL CARRY FLAG CLEAR
66      SUB R8,R8,#1       ; FOR NEXT BIT SHIFT
67      ADD R6,R6,#1       ; TO POINT TO NEXT MEMORY LOCATION IN THE SAME ROW
68       B NXT
69  MATCH   LDR R0,=0X10000000; SELECT AEGMENT FOR DISPLAY
70          LDRB R9,[R6]       ; FETCH CODE FOR KEY PRESSED
71          MOV R9,R9,LSL#16   ; SHIFT TO FIT INTO P0.16-P0.23
72          ORR R9,R9,R0       ;RETAIN BOTH SEGMENT SELSCTION AND CODE
73          STR R9,[R2]        ; SEND TO PORT0 TO DISPLAY
74          B BK1
```

```
75  DELAY    ;subroutine
76           LDR R10,=0x2ff
77  up       SUBS R10,R10,#1;
78           BNE up
79           BX LR
80  SEG_CODE0          DCB 0X3F,0X06,0X5B,0X4F
81  SEG_CODE1          DCB 0X66,0X6D,0X7D,0X07
82  SEG_CODE2          DCB 0X7F,0X6F,0X77,0X7C
83  SEG_CODE3          DCB 0X39,0X5E,0X79,0X71
84
85            END
```



Exercise:

c) Write a program to display square of a number pressed from the hex keypad on to seven segment display.

d) Write a program to use hex keypad and seven segment display as a simple calculator that performs single digit addition, subtraction, multiplication and division.

Expt. No. 7
INTERFACING EXTERNAL DAC AND PROGRAMMING INTERNAL DAC AND ADC

Aim: To interface external, 8-bit Digital to Analog Converter(DAC) and programming internal 10 bit, DAC and Analog to Digital Converter (ADC)

1.Program to interface external DAC to generate a square waveform (**Short JP3**)

```
01   INCLUDE VICVPB.S
02      AREA CHANG, CODE,READONLY
03      EXPORT __main  ;refer the line no 416 of startup.s file
04  __main
05      ENTRY
06           LDR r1,=IO0DIR ; load the address of the IODIR reg to R1
07           LDR r0,=0xFF0000;  P0.0 to P0.7 as output
08           STR r0,[r1]        ;This configure P0.0 to P0.7  as output pins
09           LDR r2,=IO0CLR  ; load the address of the IOCLR reg to R2
10           LDR r3,=IO0SET   ;
11  BACK     STR R0,[R2]
12           BL DELAY
13           STR R0,[R3]
14           BL DELAY
15           B BACK
16  DELAY    ;subroutine
17           LDR R5,=0x2FFF
18  up       SUBS R5,R5,#1;
19           BNE up
20           BX LR
21           END
```

Exercise:

a) Write an ALP to generate (i) Ramp wave (ii) Triangle wave (iii) Stair case wave of 10 steps

b) Write an ALP to generate sinewave [Use the formula (1+sinθ)*128, with θ varying in steps of 15^o , Maximum DAC input value is 0xFF].

2. **Programming internal DAC to generate a ramp waveform using internal DAC (10 bit). To see waveform: Open JP9 (at ADC module) and Observe the Analog output waveform by connecting Oscilloscope (CRO) probe to right side pin of JP9.**

```
01  PINSEL1 EQU 0XE002C004; register to configure P0.16-P0.31 for specific operation
02  DACR    EQU 0XE006C000; DAC register address
03      AREA CHANG, CODE,READONLY
04      EXPORT __main  ;refer the line no 416 of startup.s file
05  __main
06      ENTRY
07           LDR R0,=PINSEL1;
08           LDR R1,=DACR;
09           LDR R2,=0X00080000; P0.25 as DAC output
10           STR R2,[R0]
11  BACK     MOV R4, R3,LSL#6
12           STR R4,[R1]
13           ADD R3,R3,#1
14           B BACK
15           END
```

Refer to the datasheet, UM10139.pdf for the description of DACR register

c) Write an ALP to generate (i) Ramp wave (ii) Triangle wave (iii) Stair case wave of 10 steps

d) Write an ALP to generate sinewave [Use the formula (1+sinθ)*512, with θ varying in steps of 1.8^o (generate values using MATLAB), Maximum DAC input value is 0x3FF].

3.Program to interface internal ADC. Displays output on seven segment

```
01   include VICVPB.s
02   AREA ADC, CODE,READONLY
03      EXPORT __main  ;refer the line no 416 of startup.s file
04   __main
05      ENTRY
06           LDR R1,=VPBDIV; For Pclk = 30MHz
07           LDR R0,=0X02
08           STR R0,[R1]
09           LDR r1,=PINSEL1 ;
10           LDR r0,=0X40000;  PIN P0.25 AS AD0.
11           STR r0,[r1]
12           LDR r1,=IO0DIR ;
13           LDR r0,=0X30FF0000
14           STR r0,[r1]
15   UP1     LDR r1,=AD0CR ;
16           LDR r0,=0X00200410; To select operating mode for A/D conversion
17           STR r0,[r1]
18           LDR R2,=1
19           ORR R0,R2,LSL#24; TO START CONVERSION, SET 24TH BIT OF AD0CR
20           STR r0,[r1]
21   UP      LDR R1,=AD0DR4
22           LDR R0,[R1]
23           AND R2,R0,#0X80000000
24           CMP R2,#0X80000000
25           BNE UP
26           MOV R0,R0,LSR#6
27           LDR R3,=0X3FF
28           AND R0,R0,R3 ; RETAIN LOWER 10 BITS
29           LDR R3,=310  ; FOR VOLTAGE L.T. 1 V
30           CMP R0,R3
31           BPL  NXT1
32           LDR R3,=0
33           B CODE
34   NXT1    LDR R3,=621 ;FOR VOLTAGE L.T. 2 V
35           CMP R0,R3
36           BPL  NXT2
37           LDR R3,=0X10
38           B CODE
39   NXT2    LDR R3,=931; FOR VOLTAGE L.T. 3 V
40           CMP R0,R3
41           BPL  NXT3
42           LDR R3,=0X20
43           B CODE
44   NXT3    LDR R3,=1023;3.3V
45           CMP R0,R3
46           BPL  NXT4
47           LDR R3,=0X30
48           B CODE
49   NXT4   LDR R3,=0X33
50   CODE    LDR R1,=SEG_CODEL
51           AND R2,R3,#0X0F
52           LDRB R0, [R1,R2]
53           LDR R4,=0X20000000
54           MOV R0,R0,LSL#16  ; SHIFT TO FIT IN TO P0.16-P0.23
55           ORR R0,R0,R4
56           LDR R1,=IO0PIN
```

```
57          STR R0,[R1]
58          BL DELAY
59          LDR R1,=SEG_CODEU
60          AND R2,R3,#0XF0
61          MOV R2,R2,LSR#4
62          LDRB R0, [R1,R2]
63          LDR R4,=0X10000000
64          MOV R0,R0,LSL#16  ; SHIFT TO FIT IN TO P0.16-P0.23
65          ORR R0,R0,R4
66          LDR R1,=IO0PIN
67          STR R0,[R1]
68          BL DELAY
69          B UP1
70  DELAY   ;subroutine
71          LDR R10,=0x2FF
72  up1     SUBS R10,R10,#1;
73          BNE up1
74          BX LR
75  SEG_CODEL DCB 0X3F,0X06,0X5B,0X4F
76  SEG_CODEU DCB 0XBF,0X86,0XDB,0XCF
77          END
```

e) Write a program to interface seven segment display to show the readings with the resolution of 0.1 Volts for input analog voltage

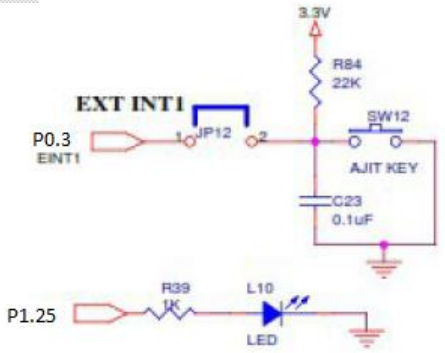
Expt. No. 8

ASSEMBLY LANGUAGE PROGRAMMING FOR ARM 7 INTERRUPTS

Aim: To program arm 7 interrupts like EINT0, EINT1, Timer0 and UART0

1. Program to understand External Hardware Interrupt1 (EINT1). Short Jumper JP12. After programming press the switch, SW12**.**

```
INCLUDE VICVPB.S
AREA INTRUPT, CODE,READONLY
EXPORT __main  ;refer the line no 416 of startup.s file
__main
    ENTRY
;   LDR R1,=MEMMAP; User Flash Mode. Interrupt vectors are
;   LDR R0,=0X01  ;not re-mapped and reside in Flash.
;   STR R0,[R1]
    LDR R1,=IO1DIR    ; to configure P1.25 as output
    LDR R0,=0X02000000
    STR R0,[R1]
    LDR R1,=PINSEL0
    LDR R0,=0XC0; TO CONFIGURE P0.3 AS EINT1
    STR R0,[R1]
    LDR R1,=EXTMODE
    LDR R0,=0X02; TO SELECT EINT1 AS EDGE TRIGGERED(1 FOR EDGE, 0 FOR LEVEL)
    STR R0,[R1]
    LDR R1,=EXTPOLAR
    LDR R0,=0X0; TO SELECT EINT1 AS negative EDGE TRIGGERED(0 for negative, 1 for positive)
    STR R0,[R1]
    LDR R1,=VICVectCntl0; slot0 with highest priority
    LDR R0,=0x2F; EINT1 number is 15 and 2  for vectored IRQ slot is enabled
    STR R0,[R1]
    LDR R1,=VICVectAddr0
    LDR R0,=IRQ_EInt1; ISR address to be stored in VICVectAddr0
    STR R0,[R1]
    LDR R1,=VICIntEnable
    LDR R0,=0x00008000; enable EINT1
     STR R0,[R1]
S   B S
IRQ_EInt1
    STMDB R13!,{R0-R1}
    LDR R1,=EXTINT
    LDR R0,=0x2
    STR R0,[R1]
    ldr r2,=0x40000000; Starting address of data memory
    ldr r0,[r2]
    LDR R1,=IO1PIN
    EOR R0,R0,#0X02000000
    STR R0,[R1]
    str r0,[r2]
    LDR R1,=VICVectAddr
    LDR R0,=0
    STR R0,[R1]
    LDMIA R13!,{R0-R1}
    subs pc,r14,#4 ; Adjust the PC to jump back to where it was interrupted
    END
```



Exercise:

a) Write an ARM7 ALP to program EINT0 (P0.16) to perform same as done in the above program

b) Write an ARM7 ALP to display the count of events on EINT1, on seven segment LED.

c) Write an ALP to generate following waveforms using DACR, based on interrupts on EINT1:

No Interrupt	Sinewave
First Interrupt	Square wave
Second Interrupt	Saw tooth wave
Third Interrupt	Triangular wave
Fourth Interrupt and so on	Repeat sine wave

2. Program for Timer0 Interrupt to display every second count on LEDs.

```
 INCLUDE VICVPB.S
 AREA T0_INT, CODE,READONLY
 EXPORT __main  ;refer the line no 416 of startup.s file
__main
    ENTRY
    LDR R1,=VPBDIV; For Pclk = 30MHz
    LDR R0,=0X02
    STR R0,[R1]
;   LDR R1,=MEMMAP; User Flash Mode. Interrupt vectors are
;   LDR R0,=0X01  ;not re-mapped and reside in Flash.
;   STR R0,[R1]
    LDR R1,=IO0DIR    ;
    LDR R0,=0x00FF0000;Configure P0.16-23 and P0.28,29 as output for seven segment
    STR R0,[R1]
    STR R0,[R1]
    LDR R1,=VICVectCntl0; slot0 with highest priority
    LDR R0,=0x24; T0 number is 4 and 2  for vectored IRQ slot is enabled
    STR R0,[R1]
    LDR R1,=VICVectAddr0
    LDR R0,=IRQ_TIM0; ISR address for Timer0 to be stored in VICVectAddr0
    STR R0,[R1]
    LDR R1,=VICIntEnable
    LDR R0,=0x10; enable Timer0
    STR R0,[R1]
    LDR R1,=T0PR;LOAD 29 SO THAT 30MHz/30= 1 microseconds
    LDR R0,=29;
    STR R0,[R1]
    LDR R1,=T0TCR
     LDR R0,=1;  Enable timer
     STR R0,[R1]
     LDR R1,=T0MR0;
     LDR R0,=1000000; Load timer match register0 to generate 1 second
     STR R0,[R1]
     LDR R1,=T0MCR;
     LDR R0,=3;  To generate interrupt on match Interrupt on MR0
     STR R0,[R1] ; and Reset on MR0: the TC will be reset if MR0 matches it.
     LDR R2,=0XFF; initial count 0
S    B S
IRQ_TIM0
     STMDB R13!,{R0-R1}
     LDR R1,=T0IR;
     LDR R0,=1
     STR R0,[R1]
     LDR R1,=VICVectAddr
     LDR R0,=0
     STR R0,[R1]
```

```
47      LDR R1,=IO0PIN
48      SUB R2,R2,#1
49      MOV R0, R2,LSL#16
50      STR R0,[R1]
51      LDMIA R13!,{R0-R1}
52      subs pc,r14,#4 ;Adjust the PC to jump back to where it was interrupted
53      END
```

Exercise:

d) Write a program to display every second count in decimal on seven segment display

e) Write a program to display any 4 waveforms one after the other every 5 seconds

3.Program to send a character received

After downloading is completed, Open the hyper terminal (Flash Magic->Tools-> TERMINAL), set the Com1 port, baud rate as programmed and Newlines: CR. Push both pins of dip-switch SW11 to OFF position. Open the jumper JP7. Press RESET switch (SW9). Pres any key on the PC keyboard the same will be displayed on the monitor.

```
01    INCLUDE VICVPB.S
02   AREA SERIAL, CODE,READONLY
03   EXPORT __main  ;refer the line no 416 of startup.s file
04  __main
05      ENTRY
06      LDR R1,=VPBDIV ; PCLK = 30MHzS
07      LDR R0,=2;
08      STR R0,[R1]
09      LDR R1,=PINSEL0 ; P0.0 and P0.1 configured as
10      LDR R0,=0X05    ; TX0 and RX0 respectively
11      STR R0,[R1]
12      LDR R1,=U0LCR ; DLAB=1 and 8 bit Caharacter length
13      LDR R0,=0X83;   and one stop bit
14      STR R0,[R1]
15      LDR R1,=U0DLL ; for generating baud of 9600
16      LDR R0,=195;
17      STR R0,[R1]
18      LDR R1,=U0DLM ;
19      LDR R0,=0;
20      STR R0,[R1]
21      LDR R1,=U0LCR
22      LDR R0,=0X03;
23      STR R0,[R1];DLAB =0
24      LDR R1,=U0IER
25      LDR R0,=0X03; Enable THRE(bit1) and RBR(bit0) interrupt
26      STR R0,[R1]
27      LDR R1,=VICVectCntl0; slot0 with highest priority
28      LDR R0,=0x26; UART0(slot6) and 2  for vectored IRQ is enabled
29      STR R0,[R1]
30      LDR R1,=VICVectAddr0
31      LDR R0,=IRQ_UART0; ISR address to be stored in VICVectAddr0
32      STR R0,[R1]
33      LDR R1,=VICIntEnable
34      LDR R0,=0x40; enable UART0 interrupt
35      STR R0,[R1]
36  S   B S
```

```
37  IRQ_UART0
38          STMDB R13!,{R0-R1}
39          LDR R1,=U0IIR; Interrupt identification reg
40  UP1     LDR R0,[R1]
41          AND R2,R0,#0X4; Check for receive data available
42          CMP R2,#0X4    ;interrupt
43          BNE UP1       ; if not equal keep monitoring for interrupt
44          LDR R1,=U0RBR;If interrupt, data is in U0RBR
45          LDRB R0,[R1]
46          LDR R2,=U0THR ; Same Character placed in U0THR
47          STRB R0,[R2]
48          LDR R1,=VICVectAddr
49          LDR R0,=0;
50          STR R0,[R1]
51          LDMIA R13!,{R0-R1}
52          subs pc,r14,#4 ; Adjust the PC to jump back
53          END
```

Exercise:

f) Write a program to display the ASCII hex equivalent of the character received by the ARM7 kit sent from the computer keyboard on the LED/ seven segment LED unit

g) Should be able to program any task such as generation of desired waveform of desired amplitude/frequency, controlling the speed of motors, controlling display units based on the command/s sent from the computer keyboard.

Expt. No. 9
ARM 7 C PROGRAMMING - I

Aim: To program ARM7 in C programming language

1.Program to display up count with a delay of 1 second on LED display unit using Timer.

```
#include<lpc21xx.h>
void delay(void);
int main(void)
{
 unsigned long int c=0xFF0000;
  IO0DIR = c;
  VPBDIV=2;
  T0MCR=(1<<0)|(1<<2);    //Generates interrupt and reset on match
   T0MR0=1000000;          //Loading match register value
   T0PR=29;                //Loading Prescalar register value

  while(1)
   {
   IO0SET = c; // Turn ? LEDs
        delay();
        c-=0x10000;
        IO0CLR = 0xfF0000; // Turn them ?
   }
}

void delay(void)
{

   T0TCR=(1<<0);           //Starting Timer
   while(!(T0IR&(1<<0)));//Waiting for interrupt
   T0IR=(1<<0);          //Clearing interrupt
   T0TCR=(1<<1);           //Reset Timer0
}
```

Exercise:

a) Write a program to generate (i) down count (ii) ring count (iii) Johnson's count (iv) decimal count

2.**Program to read input from switches(P1.16-P1.23) and display corresponding LED of LED unit.**

```
#include <lpc214x.h>
int main(void)
{   unsigned long int c=0xFF0000;
    IO0DIR = c; // Configure pins(?) on Port 0 as Output
    IO0SET=c;
    while(1)
    {   c=IO1PIN;
        IO0PIN = c; // Turn ? LEDs
    }
}
```

Exercise:

b) Write a C program to interface with switches to perform following operations:
   If SW8 is pressed one LED, if SW7 is pressed display 2 LEDs, if SW6 is pressed display 4 LEDs and, display 8 LEDs if SW5 is pressed.

c) Write a C program to interface Buzzer

3. Programming LCD to display desire message on it.

```
#include<lpc21xx.h>
#include<stdio.h>
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 display(void);
void delay(unsigned int);
unsigned char temp,temp1;
unsigned int r,r1,com[]={0x30,0x30,0x20,0x28,0x0c,6,0x80};
unsigned char *ptr,disp[] = "WELCOME MANIPAL";
int main()
{   IO0DIR = 0x000000FC;     //port intialisation for LCD
    lcd_init();              //lcd intialisation
    delay(3200);             //delay
    clr_disp();
    delay(3200);
    ptr = disp;
    while(*ptr!='\0')
    {
        temp1 = *ptr;
        lcd_data();
        ptr ++;
    }
 while(1);
}
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();
```

```
057     delay(800);
058
059 }
060 void lcd_data(void)
061 {
062     temp = temp1 & 0xf0;
063     wr_dn();
064     temp= temp1 & 0x0f;
065     temp= temp << 4;
066     wr_dn();
067     delay(100);
068 }
069 void wr_dn(void)              ////write data reg
070 {
071     IO0CLR  = 0x000000FC;     // clear the port lines.
072     IO0SET = temp;            // Assign the value to the PORT lines
073     IO0SET = 0x00000004;      // set bit  RS = 1
074     IO0SET = 0x00000008;      // E=1
075     delay(10);
076     IO0CLR = 0x00000008;
077 }
078 void lcd_com(void)
079 {
080     temp = temp1 & 0xf0;
081     wr_cn();
082     temp = temp1 & 0x0f;
083     temp = temp << 4;
084     wr_cn();
084     wr_cn();
085     delay(500);
086 }
087 void wr_cn(void)                  //write command reg
088 {
089     IO0CLR  = 0x000000FC;         // clear the port lines.
090     IO0SET  = temp;               // Assign the value to the PORT lines
091     IO0CLR  = 0x00000004;         // clear bit  RS = 0
092     IO0SET  = 0x00000008;         // E=1
093     delay(10);
094     IO0CLR  = 0x00000008;
095 }
096 void clr_disp(void)
097 {
098 // command to clear lcd display
099     temp1 = 0x01;
100     lcd_com();
101     delay(500);
102 }
103
104 void delay(unsigned int r1)
105 {
106     for(r=0;r<r1;r++);
107 }
```

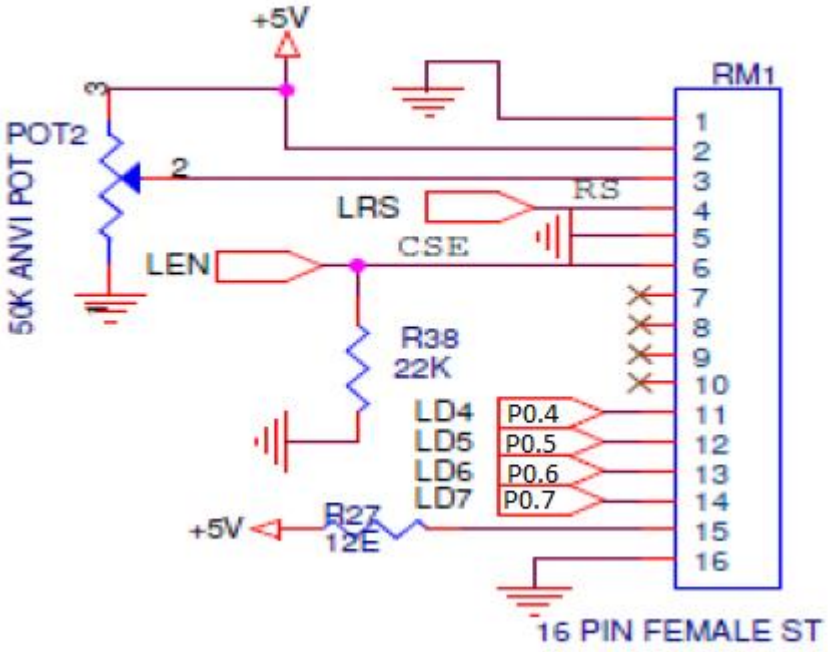
16X2 LCD interface details:

A 16X2 Alphanumeric LCD Display with back light is provided along with the Evaluation Board. The LCD is interfaced using 4 – bit mode.

RS = 0 for sending Command to the LCD, controlled by port P0.2

RS = 1 for sending Data to the LCD, controlled by port P0.2
R/W = 1 for reading from the LCD
R/W = 0 for writing to the LCD, normally it is grounded
EN = 0 for disabling the LCD
EN = 1 for enabling the LCD, controlled by port P0.3
D4= P0.4
D5= P0.5
D6= P0.6
D7= P0.7

LCD



Exercise:

d) Write a C program to interface stepper motor/ DC motor clockwise/anticlockwise and control the speed.
e) Write a C program to interface hex keypad and seven segment LEDs to display data corresponding to the key pressed.
f) Write a C program to interface hex keypad and LCD to display data corresponding to the key pressed.
g) Write a C program to display waveforms using internal DAC to display on the CRO.

Expt. No. 10
ARM 7 C PROGRAMMING - II

Aim: To program ARM7 interrupts, and PWM in C programming language

1. C-program to turn on or off the LED based in the external hardware interrupt1.

Note: After programming press the switch, SW12 (Short JP12).

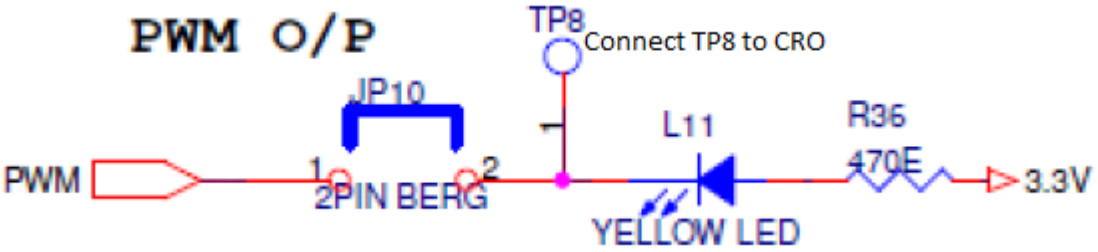
```
#include<lpc214x.h>
void Extint1_Isr(void) __irq;   //declaration of ISR
unsigned char int_flag=1;
unsigned long int c= 0X02000000;
int main(void)
{   IO1DIR |=c;      //P1.25 output
    IO1SET = c;
    PINSEL0 =0X000000c0;     //P0.3 EINT1
    EXTMODE = 0x02;      // falling egge trigger and active low
    EXTPOLAR = 0X00;
    VICVectAddr0 = (unsigned long) Extint1_Isr; // EINT1 ISR function name
    VICVectCntl0 = 0x20 | 15;//EINT1 to interrupt priority 0
    VICIntEnable |= 0x00008000;//Enable the EINT1 interrupt
    while(1)                 //looping for interrupt
    {
    if(int_flag == 0x01)
       IO1SET = 0x02000000;
       else
       IO1CLR = 0X02000000;
    }
}
void Extint1_Isr(void)__irq  //whenever there is a low edge on EINT0
{
    EXTINT = 0x02;       //clear the interrupt

    VICVectAddr=0;//Acknowledge Interrupt
     int_flag =~int_flag ;// complement the flag
}
```

2. C-program to display up count on LEDs using Timer0 interrupt

```
#include <lpc214x.h>
unsigned long int c=0xff;
__irq void T0_ISR (void)
{   c-=1;
    IO0PIN =  c<<16;    /* up count*/
    T0IR = ( T0IR | (0x01) );
    VICVectAddr = 0x00;
}
int main (void)
{
VPBDIV = 2; /* For Pclk = 30MHz */
IO0DIR = ( IO0DIR | (0X00ff0000) ); /* p1.25 as output pin for LED */
IO0PIN = IO0PIN | 0X00ff0000; /* Writing 1 to LED pin P1.25 */
VICVectAddr0 = (unsigned) T0_ISR; /* T0 ISR Address */
VICVectCntl0 = 0x00000024; /* Enable T0 IRQ slot */
VICIntEnable = 0x00000010; /* Enable T0 interrupt */
VICIntSelect = 0x00000000; /* T0 configured as IRQ */
T0TCR = 0x02; /* Reset TC and PR */
T0CTCR = 0x00; /* Timer mode, increment on every rising edge */
T0PR = 0x1D; /* Load Pre-Scalar counter with 29, timer to count every 1usec */
T0MR0 = 1000000; /* Load timer counter for 1sec delay */
T0MCR = 0x0003; /* Interrupt generate on match and reset timer */
T0TCR = 0x01; /* Enable timer */
while (1);
}
```

3.Program to generate **duty cycle** using PWM interrupt. This program generates 1 msec square wave.

```
#include <LPC21xx.h>
__irq void PWM_ISR (void)
{       PWMIR = 0x100;  /* Clear PWM4 interrupt */
    VICVectAddr = 0x00000000;
}
int main (void)
{
    VPBDIV      = 0x00000002;
    PINSEL0 = 0x00020000 ;// configure p0.8
    VICVectAddr0 = (unsigned) PWM_ISR; /* PWM ISR Address */
    VICVectCntl0 = (0x00000020 | 8); /* Enable PWM IRQ slot */
    VICIntEnable = VICIntEnable | 0x00000100; /* Enable PWM interrupt */
//  VICIntSelect = VICIntSelect | 0x00000000; /* PWM configured as IRQ */
// For PWM4 double edge
    PWMTCR = 0x02;  /* Reset and disable counter for PWM */
    PWMPR = 0x1D;   /* Prescale value for 1usec, Pclk=30MHz*/
    /* Duty Cycle=(PWMMR4/PWMMR0)*100*/
    PWMMR0 = 1000;  /* Time period of PWM wave, 1msec */
    PWMMR4 = 100;   /* Value for duty cycle (10%) */
    PWMMCR = 0x1003;/* Reset and interrupt on MR0 match, interrupt on MR4 match */
    PWMLER = 0x0D;  /* Latch enable for PWM3, PWM2 and PWM0 */
    PWMPCR = 0x1010;/* Enable PWM4 double edge controlled PWM on PWM4 */
    PWMTCR = 0x09;  /* Enable PWM and counter */
    while (1);
}
```



Exercise:

a) Write a program to control the directions of stepper motor and DC motor using EINT1.
b) Write a c program to display count of hardware interrupt1 on seven segment LEDs
c) Write a c program to display waveforms based on hardware interrupts usind DACR
d) Write a c program to display two digit decimal upcount on seven segment LEDs every second.