Channabasaveshwara Institute of Technology

(Affiliated to VTU, Belgaum & Recognized by A.I.C.T.E. New Delhi)

(An ISO 9001:2015 Certified Institution)

NH 206, (B.H. Road), Gubbi, Tumkur – 572 216. Karnataka

Department of Computer Science & Engineering

MICROCONTROLLER AND EMBEDDED SYSTEMS

LABORATORY MANUAL

SEMESTER – IV

21CS43

FACULTY IN-CHARGE

Mr. Chethan Balaji	**Mrs. Deepika K S**
Associate Professor	**Assistant Professor**

Dept. of CSE

Channabasaveshwara Institute of Technology

(Affiliated to VTU, Belgaum & Approved by AICTE, New Delhi)
(**ISO 9001:2015 Certified Institution)**
NH 206 (B.H. Road), Gubbi, Tumkur – 572 216. Karnataka

Department of Computer Science & Engineering

MICROCONTROLLER AND EMBEDDED SYSTEMS			
Course Code	**21CS43**	CIE Marks	50
Teaching Hours/Week (L:T:P: S)	3:0:2:0	SEE Marks	50
Total Hours of Pedagogy	40 T + 20 P	Total Marks	100
Credits	04	Exam Hours	03

LAB PROGRAMS LIST

1.	Using Keil software, observe the various registers, dump, CPSR, with a simple ALP programme.
2.	Write a program to find the sum of the first 10 integer numbers.
3.	Write a program to find the factorial of a number
4.	Write a program to add an array of 16 bit numbers and store the 32 bit result in internal RAM.
5.	Write a program to find the square of a number (1 to 10) using a look-up table.
6.	Write a program to find the largest or smallest number in an array of 32 numbers.
7.	Write a program to arrange a series of 32 bit numbers in ascending/descending order.
8.	Write a program to count the number of ones and zeros in two consecutive memory locations.
9.	Display “Hello World” message using Internal UART.
10.	Interface and Control a DC Motor.
11.	Interface a Stepper motor and rotate it in clockwise and anti-clockwise direction.
12.	Determine Digital output for a given Analog input using Internal ADC of ARM controller.
13.	Interface a DAC and generate Triangular and Square waveforms.
14.	Interface a 4x4 keyboard and display the key code on an LCD.
15.	Demonstrate the use of an external interrupt to toggle an LED On/Off.
16.	Display the Hex digits 0 to F on a 7-segment LED interface, with an appropriate delay in between.
17.	Demonstration of IoT applications by using Arduino and Raspberry Pi.

Course outcome (Course Skill Set)

CO 1. Explain C-Compilers and optimization
CO 2. Describe the ARM microcontroller's architectural features and program module.
CO 3. Apply the knowledge gained from programming on ARM to different applications.
CO 4. Program the basic hardware components and their application selection method.
CO 5. Demonstrate the need for a real-time operating system for embedded system applications.

HOD

PROGRAM NO.1

AIM: USING KEIL SOFTWARE, OBSERVE THE VARIOUS REGISTERS, DUMP, CPSR, WITH A SIMPLE ALP PROGRAMME

1. SAMPLE PROGRAM FOR ARITHMETIC INSTRUCTIONS

```
        AREA PROG1, CODE, READONLY

        ENTRY

START

        LDR R1, =0X00000006

        LDR R2, =0X00000002

        ADD R4, R1, R2

        ADC R5, R1, R2

        SUB R6, R1, R2

        SBC R8, R1, R2

        RSB R7, R1, R2

        RSC R3, R1, R2

STOP B STOP
```

TRACING

R1=0X00000006

R2=0X00000002

R4=R1+R2=0X00000008 (6+2=8)

R5=R1+R2+C=0X00000008(6+2+0)=8

R6=R1-R2=0X00000004(6-2=4)

R8=R1-R2-!C=0X00000003(6-2-!0=3)

R7=R2-R1=0XFFFFFFFC (2-6= -4 = 0XFFFFFFFC in hexadecimal)

RESULT: R4=0X00000008, R5=0X00000008 , R6=0X00000004 , R8=0X00000003, R7=0XFFFFFFFC

2. SAMPLE PROGRAM FOR LOGICAL INSTRUCTIONS

```
        AREA LOGIC,CODE,READONLY

        ENTRY

        MOV R1, #0X00000006

        MOV R2, #0X00000004

        ORR R3,R2,R1

        AND R5,R1,R2
        EOR R6,R1,R2
        BIC R4,R1,R2
        STOP B STOP
        END
```

TRACING

R1=0X00000006

R2=0X00000004

R3=R2|R1=0X00000006

R5=R1&R2=0X00000004

R6=R1^R2=0X00000002

R4=R1&(!R2)=0X00000002

RESULT: R3=0X00000006, R5=0X00000004, R6=0X00000002, R4=0X00000002

3.SAMPLE PROGRAM ON BRANCH INSTRUCTIONS

```
        AREA Branch, CODE, READONLY
        ENTRY
START
        LDR R0, =0XFFFFFFFF
        ADDS R0, #1   /CMN  R0,#1
STOP B STOP
```

TRACING: R0 = 0XFFFFFFFF

R0=R0+1=0X00000000 BUT IT WILL UPDATE FLAGS IN THE CPSR(N=1,Z=1,C=1,V=0)

RESULT: R0=0X00000000 BUT CPSR (N=1, Z=1, C=1, V=0)

4.WRITE AN ALP PROGRAM TO EVALUATE THE ARITHMETIC EXPRESSION

X= (A + C) - D

```
        AREA EX, CODE, READONLY
        ENTRY
START LDR R4,=A             ; get address for A
        LDR R0,[R4]         ; get value of A
        LDR R4, =C          ; get address for C , reusing R4
        LDR R1, [R4]        ; get value of C
        ADD R3,R0,R1        ; compute A+C
        LDR R4, =D          ; get address for D
        LDR R2,[R4]         ; get value of D
        SUB R3,R3,R2        ; complete computation of X
        LDR R4, =X          ; get address for X
        STR R3, [R4]        ; store value of X
STOP B STOP
A DCD 0X45
C DCD 0X25
D DCD 0X05
        AREA DATA1 ,DATA, READWRITE
X DCD 0
END
```

TRACING

```
R4=0X0000002C
R0=0X00000045
R4=0X0000002C
R1=0X00000025
R3=(0X45+0X25) = 0X0000006A
R4=0X00000034
R2=0X00000005
R3=R3-R2=0X00000065
R4=0X40000000
```

RESULT: R3=0X00000065 AND WITH MEMORY ADDRESS 0X40000000=0X00000065

PROGRAM NO.2

AIM: TO WRITE A PROGRAM TO FIND THE SUM OF THE FIRST 10 INTEGER NUMBERS.

1+2+3+4+5+6+7+8+9+10=55=0X37

PROGRAM

```
        AREA SUM, CODE, READONLY
        ENTRY
START
        MOV R0,#10      ;set the counter=10
        MOV R1,#0       ; initialize the register to store result
        MOV R2,#1        ;take 1st number to add
NEXT
        ADD R1,R1,R2   ; add the numbers
        ADD R2,#1        ; increment the integer
        SUBS R0,#1       ; decrement counter
        BNE NEXT         ;branch to the loop if not equal to zero
STOP B STOP
        END
```

TRACING:

R0=10=0XA

R1=0

R2=1

R1=R1+R2=0+1=1

R2=R2+1=1+1=2

R0=R0-1=10-1=9

R1=1+2=3

R2=2+1=3

R0=9-1=8

R1=3+3=6

R2=3+1=4

R0=8-1=7

R1=6+4=10=0XA

R2=4=1=5

R0=7-1=6

R1=10+5=15=0XF

R2=5+1=6

R0=6-1=5

R1=15+6=21=0X16

R2=6+1=7

R0=5-1=4

R1=21+7=28=0X1C

R2=7+1=8

R0=4-1=3

R1=28+8=36=0X24

R2=8+1=9

R0=3-1=2

R1=36+9=45=0X2D	R1=45+10=55=0X37
R2=9+1=10	R2=10+1=11
R0=2-1=1	R0=1-1=0

RESULT: R1=55=0X37

PROGRAM NO.3

AIM: WRITE A PROGRAM TO FIND THE FACTORIAL OF A NUMBER.

Ex: 5! = 5*4*3*2*1=120=0x78

```
        AREA FACT, CODE, READONLY
ENTRY
        MOV R1,#5   ; take the factorial number
        MOV R2,#1   ; initialize register to store result
BACK
        CMP R1,#0   ;  compare R1=0 if r1=0 stop and return result (R2 holds result)
        BEQ STOP   ; else
        MUL R2,R1,R2  ; multiply R1 with R2,
        SUB R1,#1   ; decrement R1 by 1 branch to step 3
        B BACK      ; repeat until R1=0

STOP B STOP
        END
```

TRACING:

R1=5

R2=1

CHECK R1=0 ,NO R1=5

R2=R1*R2=5*1=5

R1=4

CHECK R1=0 NO R1=4

R2=R1*R2=4*5=20

R1=3

CHECK R1=0 NO R1=3

R2=R1*R2=3*20=60

R1=2

CHECK R1=0 NO R1=2

R2=R1*R2=2*60=120

R1=1

CHECK R1=0 NO R1=1

R2=R1*R2=1*120=120

R1=0

CHECK R1=0 YES R1=0

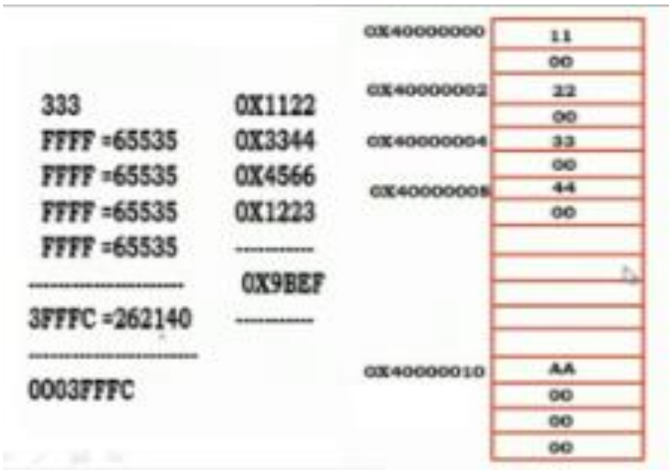
STOP EXECUTION

RESULT: R2=120=0X78

PROGRAM NO.4

AIM:TO WRITE A PROGRAM TO ADD AN ARRAY OF 16 BIT NUMBERS AND STORE THE 32-

BIT RESULT IN INTERNAL RAM.



```
AREA ARRAY1, CODE, READONLY
ENTRY
            LDR R0, MEMORY       ;load starting address of the array
            MOV R1, #4          ; load array size
            LDRH R2, [R0]        ;load 1st number
            ADD R1, #-1         ; decrement counter
UP
            ADD R0, R0, #2       ; increment pointer by 2
            LDRH R3, [R0]       ;load second number
            ADD R2, R3, R2       ;R2=R3+R2
NEXT
            ADD R1, #-1         ;decrement counter
            CMP R1, #0          ;is counter=0?
            BNE UP               if counter!=0? then repeat
            LDR R0, RESULT
            STR R2, [R0]        ;store the result
STOP B STOP

MEMORY DCD 0X40000000            ;starting address of the array
RESULT DCD 0X40000010            ;starting address of the result
END
```

TRACING:

R0=0X40000000

R1=4

R2=[0X40000000]=0X00000011

R1=3

R0=0X40000002

R3=[0X40000002]= 0X00000022

R2=0X00000033

R1=2

R0=0X40000004

R3=[0X40000002]= 0X00000033

R2=0X00000066

R1=1

R0=0X40000006

R3=[0X40000002]= 0X00000044

R2=0X000000AA

R1=0

R0=0X40000010

R2=[0X40000010]= 0X000000AA

RESULT: R2=[0X40000010]= 0X000000AA

PROGRAM NO.5

AIM: TO WRITE A PROGRAM TO FIND THE SQUARE OF A NUMBER (1 TO 10) USING A LOOK-UP TABLE.

```
AREA SUARES,CODE,READONLY

ENTRY

        MOV R1,#3               ;take the number to find square

        LDR R0,=LOOKUP          ; data in lookup table address moved to R0

        MOV R1,R1,LSL#02        ; the content in R1 left shift by 2

        ADD R0,R0,R1            ; add R0 and R1

        LDR R3,[R0]             ;data in address of R0 loaded to R3

STOP B STOP

LOOKUP DCD 0X0,0X1,0X4,0X9,0X16,0X25,0X36,0X49,0X64,0X81,0X100

END
```

TRACING:

R1=3

R0=0x00000018

R0=0x0000000C

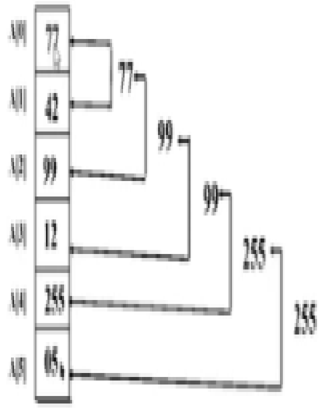
R0=R0+R1=[0x00000024] pointing to address

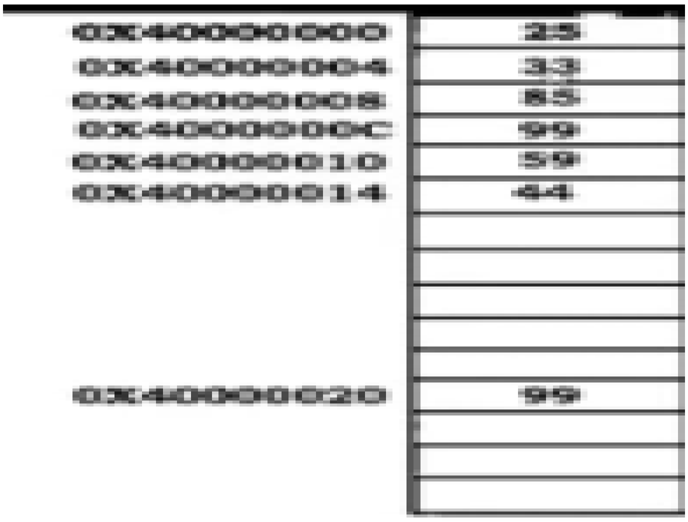
R3=[0x00000024] = 0x00000009

RESULT: R3=[0x00000024] = 0x00000009

PROGRAM NO.6

AIM: TO WRITE A PROGRAM TO FIND THE LARGEST OR SMALLEST NUMBER IN AN ARRAY OF 32 NUMBERS.





0X40000000	25
0X40000004	33
0X40000008	85
0X4000000C	99
0X40000010	59
0X40000014	44
0X40000020	99

```
AREA LARGEST,CODE,READONLY
        ENTRY
                MOV R5,#5
                LDR R0,A
                LDR R2,[R0]
NEXT            ADD R0,#4
                LDR R3,[R0]
                CMP R2,R3
                BHS LARGE
                MOV R2,R3
LARGE                   SUBS R5,#1
                BNE NEXT
                LDR R1,RES
                STR R2,[R1]
STOP  B STOP
A DCD 0X40000000
RES DCD 0X40000020
END
```

TRACING:

R5=5

R0=0X40000000

R2=[0X40000000]=25

RO=0X40000004

NEXT R3=[0X40000004]=33

COMPARE 25 AND 33 IS 25>33 NO THEN

R2=R3=33

R5=4 (R5 !=1) THEN BRANCH TO NEXT

RO=0X40000008

R3=[0X40000008]=85

IS 33>85 NO THEN

R2=R3=85

R5=3 (R5 !=1) THEN BRANCH TO NEXT

RO=0X4000000C

R3=[0X4000000C]=99

IS 85>99 NO THEN

R2=R3=99

R5=2 (R5 !=1) THEN BRANCH TO NEXT

RO=0X40000010

R3=[0X40000010]=59

IS 99>59 TES THEN

R5=1 (R5 !=0) THEN BRANCH TO NEXT

RO=0X40000014

R3=[0X40000014]=44

IS 44>99 NO THEN

R5=0 (R5 !=0) THEN

R1=0X40000020

R2=[0X40000020]=99

RESULT: R2=[0X40000020]=99

PROGRAM NO.7

AIM: TO WRITE A PROGRAM TO ARRANGE A SERIES OF 32-BIT NUMBERS IN ASCENDING/DESCENDING ORDER.

```
AREA ASCENDING, CODE, READONLY
          ENTRY
                              MOV R0,#0X00000003
NXTPASS                       MOV R1,#03
                              MOV R2,#0X40000000
NXTCMP
                              LDR R3,[R2]
                              ADD R2,R2,#04
                              LDR R4,[R2]
                              CMP R3,R4
                              BLT NOEXCG
                              STR R3,[R2]
                              SUB R2,R2,#04
                              STR R4,[R2]
                              ADD R2,R2,#04
NOEXCG
                              SUB R1,R1,#01
                              CMP R1,#00
                              BNE NXTCMP
                              SUB R0,R0,#01
                              CMP R0,#00
                              BNE NXTPASS
STOP B STOP
                    END
```

TRACING:

22 **0X40000000**	**11** **0X40000004**	**44** **0X40000008**	**33** **0X4000000C**

R0=3

R1=3

R2=**0X40000000**

R3=22

R2=0X40000004

R4=11

COMPARE R3 AND R4 IS R3<R4 THEN

R3=11

R2=**0X40000000**

R4=22

11 **0X40000000**	**22** **0X40000004**	**44** **0X40000008**	**33** **0X4000000C**

R2= **0X40000004**

R1=2

CMP R1!=0 ITS 2

R3=22

R2=0X40000008

R4=44

COMPARE R3 AND R4 IS R3<R4 YES THEN

R1=1

R3=44

R2=0X4000000C

R4=33

COMPARE R3 AND R4 IS R3<R4 THEN

11 **0X40000000**	**22** **0X40000004**	**33** **0X40000008**	**44** **0X4000000C**

R3=33

R2=0X40000008

R4=44

R2=0X4000000C

R1=0 THEN

R0=2

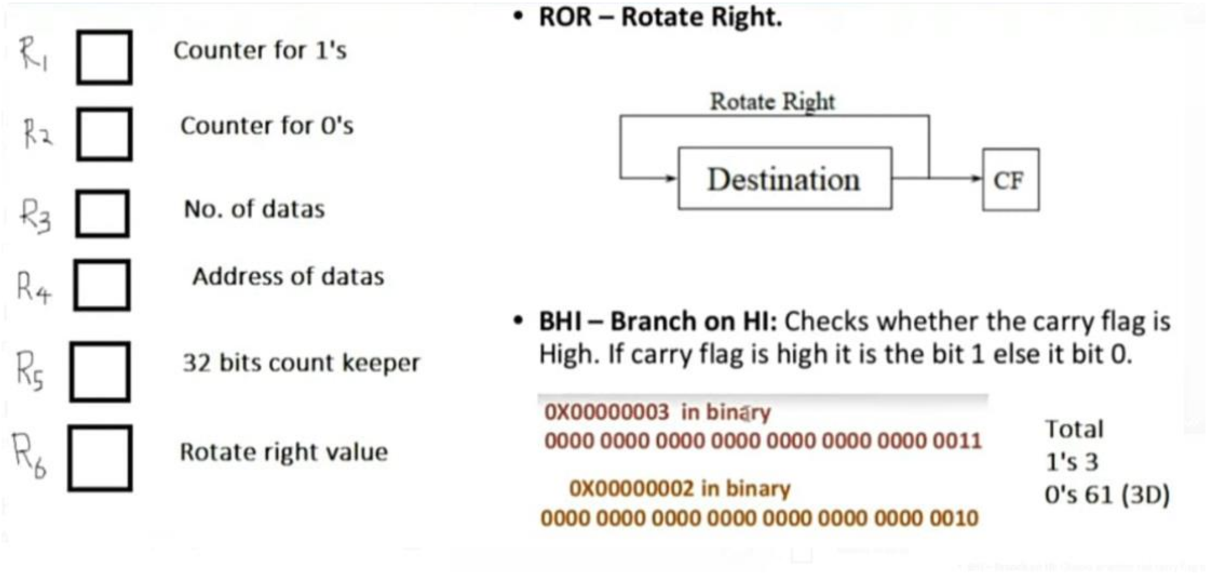
COMPARE R0=0 NO LOOP REPEATS

RESULT:

11	**22**	**33**	**44**
0X40000000	**0X40000004**	**0X40000008**	**0X4000000C**

PROGRAM NO.8

AIM: TO WRITE A PROGRAM TO COUNT THE NUMBER OF ONES AND ZEROS IN TWO CONSECUTIVE MEMORY LOCATIONS.



R1 □ Counter for 1's

R2 □ Counter for 0's

R3 □ No. of datas

R4 □ Address of datas

R5 □ 32 bits count keeper

R6 □ Rotate right value

- **ROR – Rotate Right.**

- **BHI – Branch on HI:** Checks whether the carry flag is High. If carry flag is high it is the bit 1 else it bit 0.

0X00000003 in binary
0000 0000 0000 0000 0000 0000 0000 0011

0X00000002 in binary
0000 0000 0000 0000 0000 0000 0000 0010

Total
1's 3
0's 61 (3D)

```
AREA ONESS,CODE,READONLY
          ENTRY
                    MOV R1,#0                    ;counter for ones
                    MOV R2,#0                    ;counter for zeros
                    MOV R3,#2                    ;counter to set two words
                    LDR R4,=VALUE                ;loads the address of value
LOOP2
                    MOV R5,#32
                    LDR R6,[R4],#4
LOOP0
                    MOVS R6,R6,ROR #1
                    BHI ONES
                    ADD R2,R2,#1
                    B LOOP1
ONES
                    ADD R1,R1,#1
LOOP1
                    SUBS R5,R5,#1
                    BNE LOOP0
```

```
            SUBS R3,R3,#1
            BNE LOOP2
STOP B STOP
VALUE DCD 0X3,0X2
END
```

TRACING:

R1=0

R2=0

R3=2

R4=0X00000040

R5=32 OR 0X00000020

R6=0X00000003, R4=0X00000044

R6=80000001

IS C=1? YES BRANCH TO ONES

R1=0X00000001

R5=0X0000001F OR 31

CHECKS R5=0 NO ITS 31 BRANCH TO LOOP0

R6=0XC000000

IS C=1? YES BRANCH TO ONES

R1=0X00000002

R5=0X0000001E OR 30

CHECKS R5=0 NO ITS 31 BRANCH TO LOOP0

R6=0X60000000

IS C=1? NO ITS ZERO THEN

R2=0X00000001 BRANCH TO LOOP1 THEN

R5=0X0000001D OR 29

CHECKS R5=0 NO ITS 29 BRANCH TO LOOP0

THE LOOP WILL REPEAT UNTILL R5=0

RESULT:

No. of ones =3 = R1=0x00000003

No of zeros = 61 = R2=0x0000003D

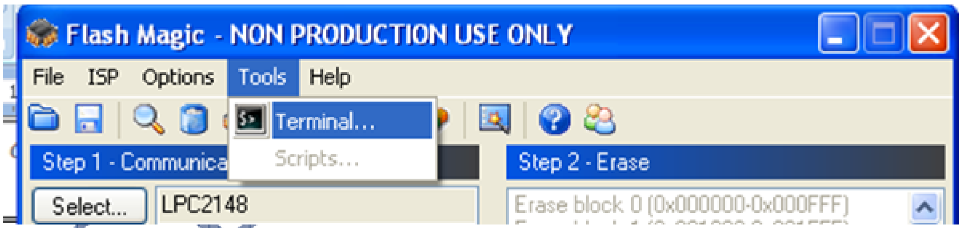
PROGRAM NO.9

AIM: TO DISPLAY “HELLO WORLD” MESSAGE USING INTERNAL UART

PROGRAM:

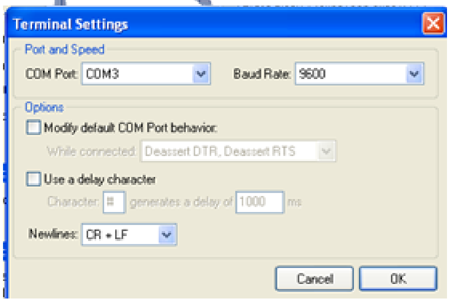
```
#include <LPC21xx.H>                          /* LPC21xx definitions */
Voidart0_init (void);
void  uart0_putc(char);
void uart0_puts(char *);         // declarations

void delay_ms(int count)                      // delay subroutine
 {
   int j=0,i=0;
   for(j=0;j<count;j++)
   {
     for(i=0;i<35;i++);
   }
 }

 int main (void)  // main program
 {
   uart0_init();                              // Initialize UART0
   delay_ms(100000);

   while (1)    // continuous loop
    {

   uart0_puts ("\n\r Hello World\n\r");   // string to be displayed.
   delay_ms(1000000);
   }
 }

 void uart0_init() // UART 0 initialization

      {
              PINSEL0 = 0x00000005;

              U0LCR = 0x83;

              U0DLL = 97;

              U0LCR = 0x03;

              }
```

```
void uart0_putc(char c)
{
        while(!(U0LSR & 0x20));

// Wait until UART0 ready to send character The U0LSR is a read-only
register that provides status information on the UART0 TX and RX
blocks.

        U0THR = c;
// Send character. Transmit holding register holds very recent data

}

void uart0_puts(char *p)
{
        while(*p) // Point to character
        {
                uart0_putc(*p++);
// Send character then point to next character
        }
}
```



LPC2148

Erase block 0 (0x000000-0x000FFF)



COM Port: COM3

Baud Rate: 9600

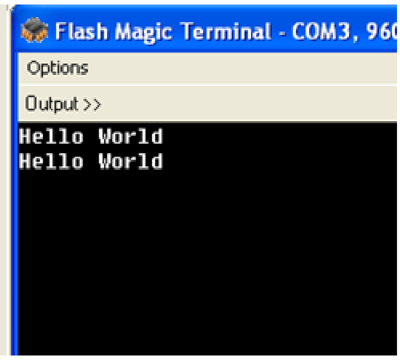
□ Modify default COM Port behavior

While connected: Deassert DTR, Deassert RTS

□ Use a delay character

Character: # generates a delay of 1000 ms

Newlines: CR + LF



```
Hello World
Hello World
```

PROGRAM NO.10

AIM: TO INTERFACE AND CONTROL A DC MOTOR.

In most of the applications controlling the speed of DC motor is essential where the precision and protection are the essence. Here we will use the PWM technique to control the speed of the motor. LPC 2148 has one PWM channel with six ports. PWM changes the average output voltage by fast switching. By changing the on time, the output voltagecan be 0 to 100%. There are two software parameters that need a little explanation: cycle and offset. Cycle is the length of a PWM duty cycle and offset is the one time of a duty cycle.

PROGRAM:

```
#include <LPC214x.H>
void delay_led(unsigned long int); // Delay Time Function
int main(void)
{

IO1DIR = 0xC0000000;
IO0DIR = 0x00200000;
while(1) // Loop Continue
{
IO0SET = 0x00200000;
delay_led(15000);
IO1SET = 0x80000000;
IO1CLR = 0x40000000; // Clear Pin P0.7, 6, 5, 4 (ON LED)
delay_led(1500000); // Display LED Delay
IO1SET = 0x40000000;
IO1CLR = 0x80000000; // Set Pin P0.7, 6, 5, 4 (OFF LED)
delay_led(1500000); // Display LED Delay
}
}
/***********************/
/* Delay Time Function */
/***********************/
void delay_led(unsigned long int count1)
{
while (count1 > 0) {count1--;} // Loop Decrease Counter
}
```

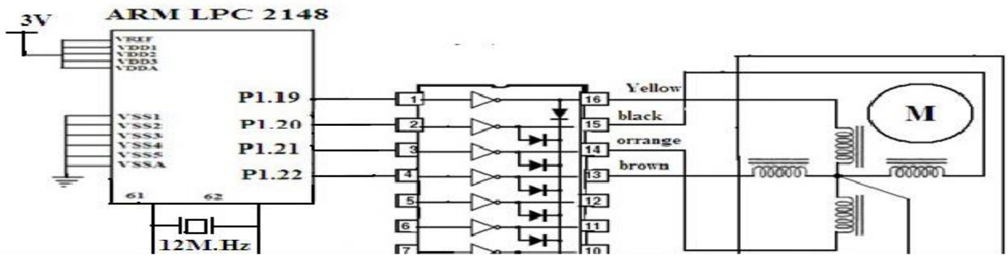
PROGRAM NO.11

AIM: INTERFACE A STEPPER MOTOR AND ROTATE IT IN CLOCKWISE AND ANTI-CLOCKWISE DIRECTION.

Stepper motors consist of a permanent magnetic rotating shaft, called the rotor, and electromagnets on the stationary portion that surroundsthe motor, called the stator. Figure 1 illustrates one complete rotation ofa stepper motor. At position 1, we can see that the rotor is beginning at the upper electromagnet, which is currently active (has voltage applied to it). To move the rotor clockwise (CW), the upper electromagnet is deactivated and the right electromagnet is activated, causing the rotor to move 90 degrees CW, aligning itself with the active magnet. This process is repeated in the same manner at the south and west electromagnets until we once again reach the starting position.

PROGRAM:

```
#include <LPC214X.h>

void delay();


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


int main()
{

  IO0DIR=0x000F0000;                    //Consider ARM port Pin from 16-19
                                         //And set these pins
  while (1)
  {
  //while (IO0PIN & 0x00008000);
  //while (! (IO0PIN & 0x00008000));


                IO0PIN=0x00010000;
                delay ();
                IO0PIN=0x00020000;
                delay ();
                IO0PIN=0x00040000;
                delay ();
                IO0PIN=0x00080000;
                delay();


                 }
                }
```


3V
ARM LPC 2148
VREF
VDD1
VDD2
VDD3
VDDA
VSS1
VSS2
VSS3
VSS4
VSS5
VSSA
P1.19
P1.20
P1.21
P1.22
61
62
12M.Hz
1
2
3
4
5
6
7
16
15
14
13
12
11
10
Yellow
black
orrange
brown
M

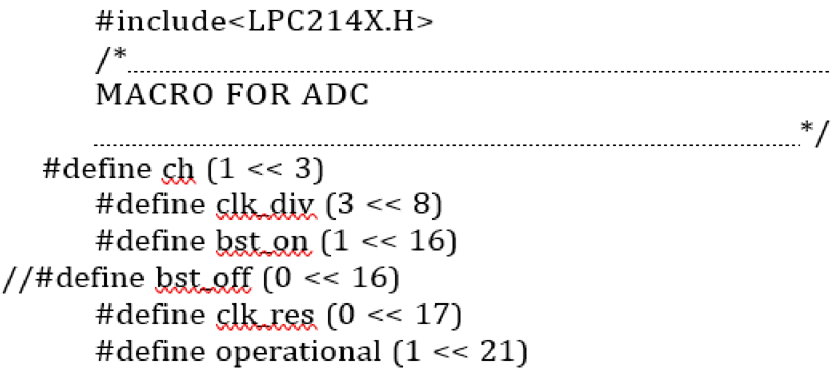
PROGRAM NO.12

AIM: TO DETERMINE DIGITAL OUTPUT FOR A GIVEN ANALOG INPUT USING INTERNAL ADC OF ARM CONTROLLER.

Analog to Digital Converter (ADC) is used to convert analog signal into digital form. LPC2148 has two inbuilt 10-bit ADC i.e. ADC0 & ADC1.ADC0 has 6 channels &ADC1 has 8 channels. Hence, we can connect 6 distinct types of input analog signals to ADC0 and 8 distinct types of input analog signals to ADC1.

ADCs in LPC2148 use Successive Approximation technique to convert analog signal into digital form. This Successive Approximation process requires a clock less than orequal to 4.5 MHz. We can adjust this clock using clock divider settings. Both ADCs in LCP2148 convert analog signals in the range of 0V toVREF (typically 3V; not to exceed VDDA voltage level).

PROGRAM:



```
        #include<LPC214X.H>
        /*...........................................................
        MACRO FOR ADC
        ..............................................................*/
    #define ch (1 << 3)
        #define clk_div (3 << 8)
        #define bst_on (1 << 16)
//#define bst_off (0 << 16)
        #define clk_res (0 << 17)
        #define operational (1 << 21)
        #define start (0 << 24)
        #define adc_init_macro ch | clk_div | bst_on | clk_res |
operational | start
        /*..........................................................................
        MACRO FOR LCD
        ...........................................................................*/
        #define EN (1 << 28)
        #define RW (1 << 29)
        #define RS (1 << 22)
        #define DATA (0Xff << 6)
        #define port EN | RW | RS | DATA

/*..........................................................................
FUNCTION DECLARATIONS
...........................................................................*/
                void adc_init(void);
                void delay(int count);
                void cmd(int c);
                void data(char d);
                void lcd_string(char *str);
                void display(unsigned int n);

/*..........................................................................
                GLOBAL VARIABLES
...........................................................................*/
                unsigned int result;
            float voltage;
                char volt[18];
/*..........................................................................
                FUNCTION DEFINITIONS
...........................................................................*/
                void adc_init(void)

{
        AD0CR = adc_init_macro;
}
void cmd(int c)
{
        IOPIN0 = c << 6;
        IOCLR0 = RW;
        IOCLR0 = RS;
        IOSET0 = EN;
        delay(100);
        IOCLR0 = EN;
}
```

```
void data(char d)
{
    IOPIN0 = d << 6;
    IOCLR0 = RW;
    IOSET0 = RS;
    IOSET0 = EN;
    delay(100);
    IOCLR0 = EN;
}

void lcd_string(char *str)
{
    while(*str)
    {
        data(*str);
        str++;
        delay(20);
    }
}
void display(unsigned int n)
{
    if(n == 0)
        data(n+0x30);
    if(n)
    {
        display(n /  10);
        data((n % 10) + 0x30);
    }
}


      void delay(int count)
      {
          int i,j;
          for(i = 0;i < count;i++)
              for(j = 0;j < 5000;j++);
      }
/*-----------------------------------------------------------------------
MAIN
-----------------------------------------------------------------------*/
          int main()
          {
              int c = 0;
              IODIR0 |= port ;
              PINSEL1|=0x10000000;
        cmd(0x38);
        cmd(0x0E);
        cmd(0X80);
        cmd(0X01);
        adc_init();
        lcd_string("ADC PROGRAM");
        cmd(0X01);
        while(1)
        {
            cmd(0x01);
            while((AD0DR3 & (0x80000000)==0));
            result = (AD0DR3 & (0X3FF << 6));
            result = result >> 6;
            lcd_string("ADC:");
            cmd(0x86);
            display(result);
            voltage = ( (result/1023.0) * 3.3 );
            cmd(0xc0);
            sprintf(volt, "Voltage=%.2f V  ", voltage);
            lcd_string(volt);
            //delay(1000);
        }
    }
```

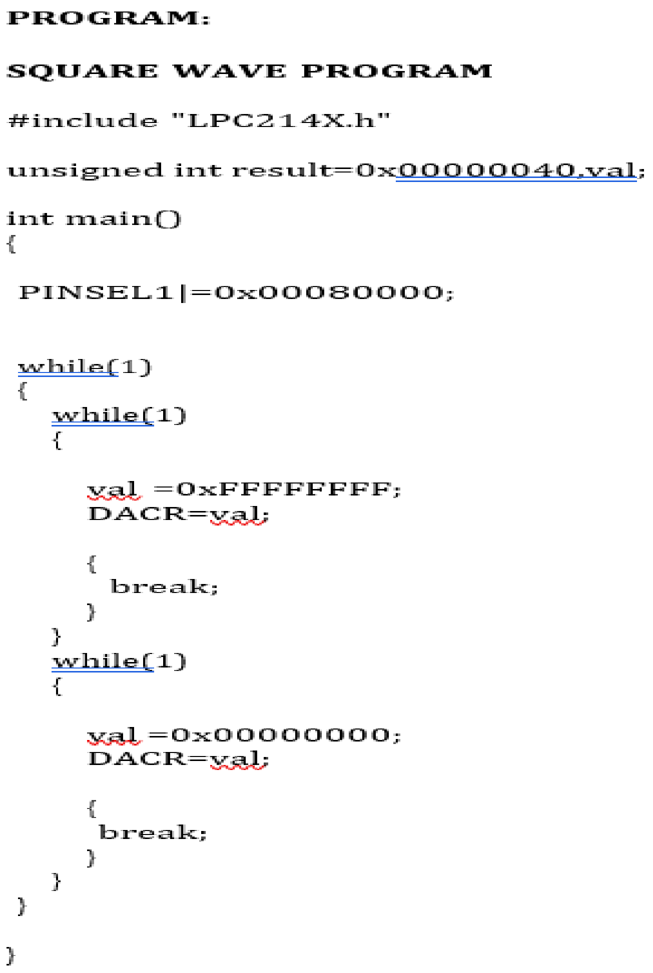
PROGRAM NO.13

AIM: INTERFACE A DAC AND GENERATE TRIANGULAR AND SQUARE WAVEFORMS.

Digital to Analog Converter (DAC) are mostly used to generate analog signals (e.g. sine wave, triangular wave etc.) from digital values.

- LPC2148 has 10-bit DAC with resistor string architecture. It also works in Power down mode.
- LPC2148 has Analog output pin (AOUT) on chip, where we can get digital value in the form of Analog output voltage.
- The Analog voltage on AOUT pin is calculated as ((**VALUE**/1024) * VREF). Hence, we can change voltage by changing **VALUE**(10-bit digital value) field in **DACR** (DAC Register).

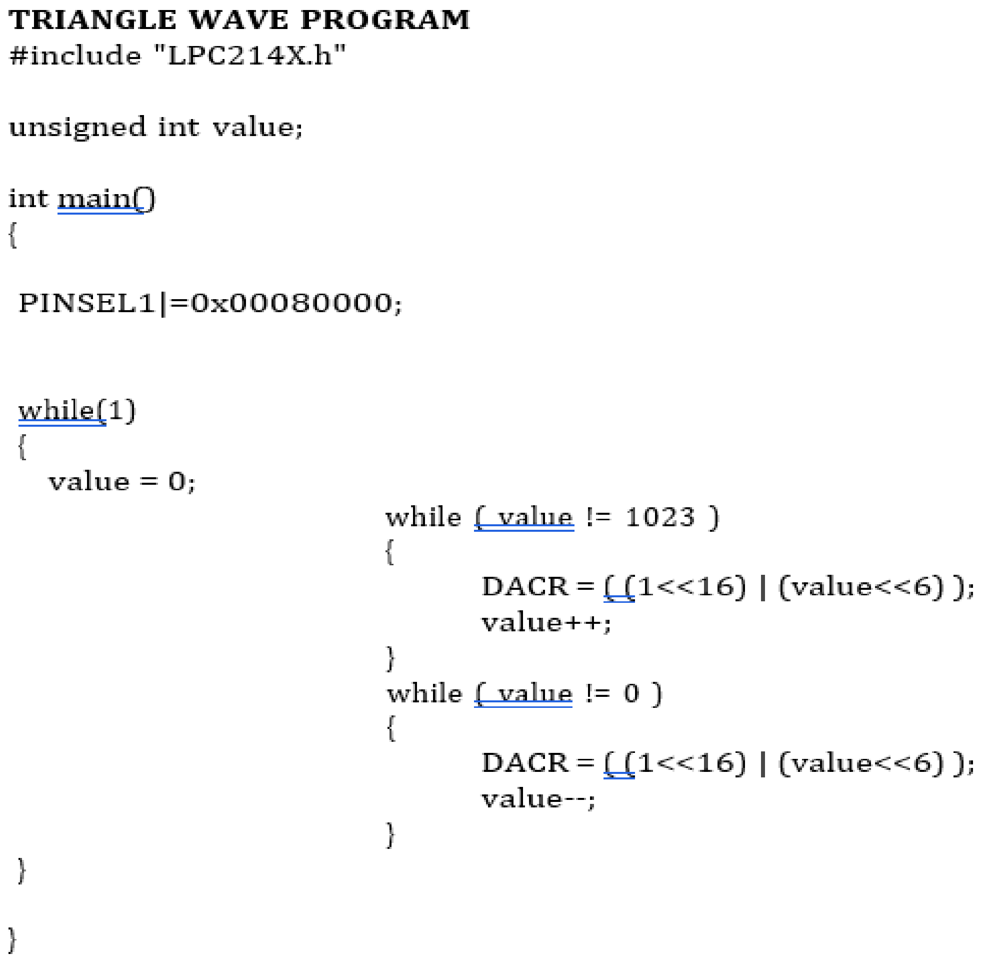
e.g. if we set **VALUE** =512,then, we can get analog voltage on AOUT pin as ((512/1024) * VREF) = VREF/2.



PROGRAM:

SQUARE WAVE PROGRAM

```
#include "LPC214X.h"

unsigned int result=0x00000040.val;

int main()
{

 PINSEL1|=0x00080000;

 while(1)
 {
    while(1)
    {

       val =0xFFFFFFFF;
       DACR=val;

       {
         break;
       }
    }
    while(1)
    {

       val =0x00000000;
       DACR=val;

       {
        break;
       }
    }
 }

}
```

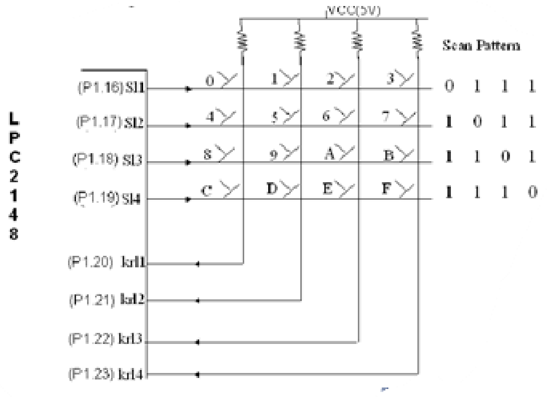


TRIANGLE WAVE PROGRAM

```
#include "LPC214X.h"

unsigned int value;

int main()
{

 PINSEL1|=0x00080000;


 while(1)
 {
    value = 0;
                                  while ( value != 1023 )
                                  {
                                          DACR = ( (1<<16) | (value<<6) );
                                          value++;
                                  }
                                  while ( value != 0 )
                                  {
                                          DACR = ( (1<<16) | (value<<6) );
                                          value--;
                                  }
 }

}
```

PROGRAM NO.14

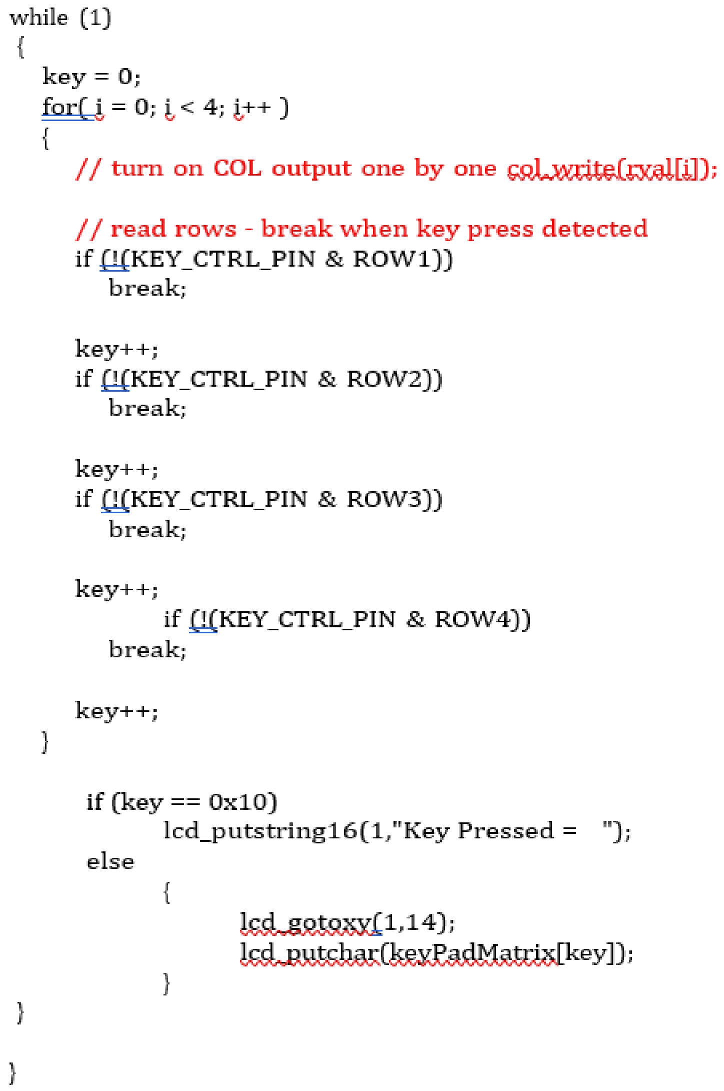
AIM: TO INTERFACE A 4X4 KEYBOARD AND DISPLAY THE KEY CODE ON AN LCD.



PROGRAM:

```
#include <LPC214x.H>                    /* LPC214x definitions */
#include "lcd.h"

//////////////////////////////////////////////////
// Matrix Keypad Scanning Routine
//
// COL1 COL2 COL3 COL4
// 0    1    2    3    ROW 1
// 4    5    6    7    ROW 2
// 8    9    A    B    ROW 3
// C    D    E    F    ROW 4
//////////////////////////////////////////////////

#define SEG7_CTRL_DIR          IO0DIR
#define SEG7_CTRL_SET       IO0SET
#define SEG7_CTRL_CLR       IO0CLR

#define COL1                (1 << 16)
#define COL2                (1 << 17)
#define COL3                (1 << 18)
#define COL4                (1 << 19)

#define ROW1                (1 << 20)
#define ROW2                (1 << 21)
```

```
#define ROW3            (1 << 22)
#define ROW4            (1 << 23)

#define COLMASK                 (COL1 | COL2 | COL3 | COL4)
#define ROWMASK                        (ROW1 | ROW2 | ROW3 | ROW4)

#define KEY_CTRL_DIR    IO1DIR
#define KEY_CTRL_SET      IO1SET
#define KEY_CTRL_CLR      IO1CLR
#define KEY_CTRL_PIN      IO1PIN

/////////////// COLUMN WRITE ////////////////////
void col_write( unsigned char data )
{
  unsigned int temp=0;

  temp=(data << 16) & COLMASK;

  KEY_CTRL_CLR |= COLMASK;
  KEY_CTRL_SET |= temp;
}


/////////////////////////////////// MAIN
///////////////////////////////////////
int main (void)
{
unsigned char key, i;
unsigned char rval[] = {0x7,0xB,0xD,0xE,0x0};
unsigned char keyPadMatrix[] =
{
    '4','8','B','F',
    '3','7','A','E',
    '2','6','0','D',
    '1','5','9','C'
};

  init_lcd();
```

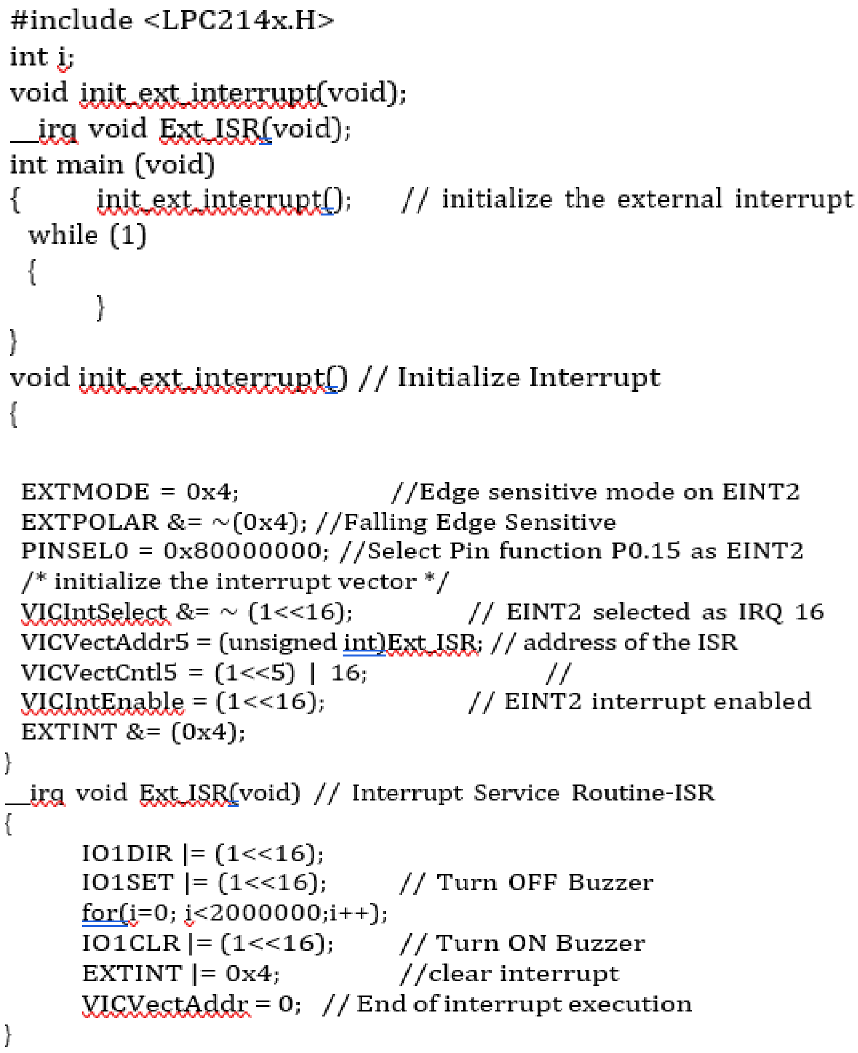


```
while (1)
 {
    key = 0;
    for( i = 0; i < 4; i++ )
    {
       // turn on COL output one by one col_write(rval[i]);

       // read rows - break when key press detected
       if (!(KEY_CTRL_PIN & ROW1))
          break;

       key++;
       if (!(KEY_CTRL_PIN & ROW2))
          break;

       key++;
       if (!(KEY_CTRL_PIN & ROW3))
          break;

       key++;
               if (!(KEY_CTRL_PIN & ROW4))
          break;

       key++;
    }

        if (key == 0x10)
               lcd_putstring16(1,"Key Pressed =   ");
        else
               {
                       lcd_gotoxy(1,14);
                       lcd_putchar(keyPadMatrix[key]);
               }
 }

}
```

PROGRAM NO.15

AIM: TO DEMONSTRATE THE USE OF AN EXTERNAL INTERRUPT TO TOGGLE AN LED ON/OFF.

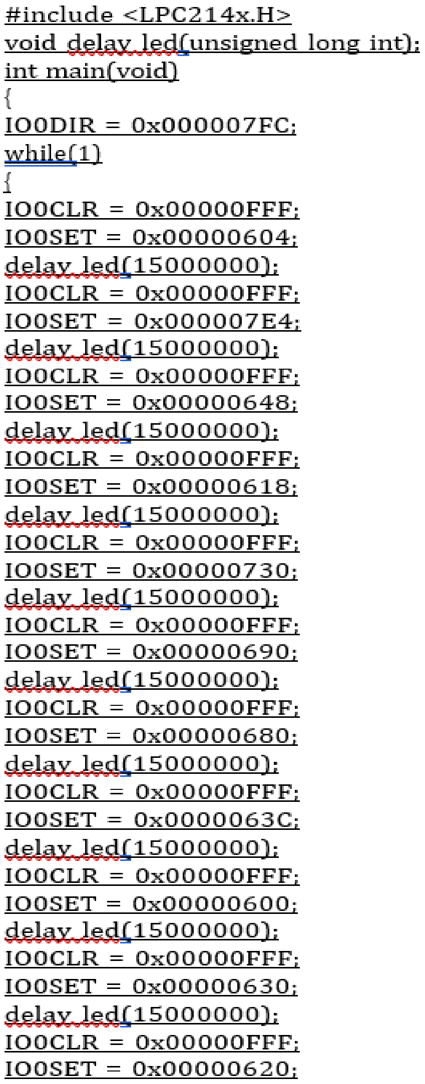
PROGRAM:



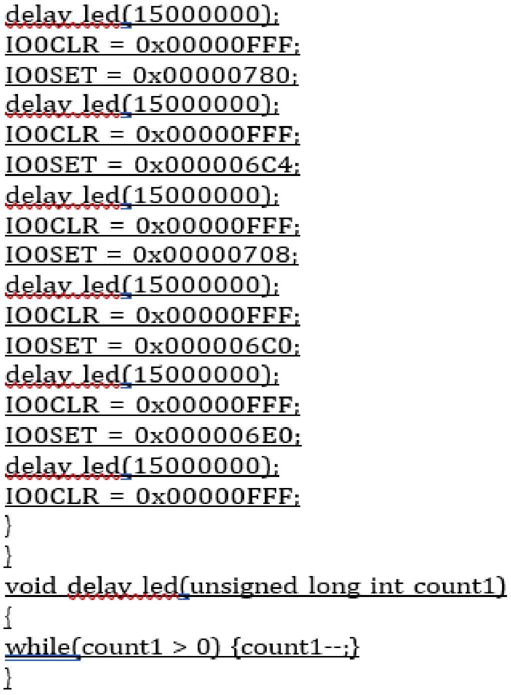
```
#include <LPC214x.H>
int i;
void init_ext_interrupt(void);
__irq void Ext_ISR(void);
int main (void)
{       init_ext_interrupt();     // initialize the external interrupt
  while (1)
  {
        }
}
void init_ext_interrupt() // Initialize Interrupt
{


 EXTMODE = 0x4;                  //Edge sensitive mode on EINT2
 EXTPOLAR &= ~(0x4); //Falling Edge Sensitive
 PINSEL0 = 0x80000000; //Select Pin function P0.15 as EINT2
 /* initialize the interrupt vector */
 VICIntSelect &= ~ (1<<16);            // EINT2 selected as IRQ 16
 VICVectAddr5 = (unsigned int)Ext_ISR; // address of the ISR
 VICVectCntl5 = (1<<5) | 16;                    //
 VICIntEnable = (1<<16);              // EINT2 interrupt enabled
 EXTINT &= (0x4);
}
__irq void Ext_ISR(void) // Interrupt Service Routine-ISR
{
        IO1DIR |= (1<<16);
        IO1SET |= (1<<16);        // Turn OFF Buzzer
        for(i=0; i<2000000;i++);
        IO1CLR |= (1<<16);        // Turn ON Buzzer
        EXTINT |= 0x4;            //clear interrupt
        VICVectAddr = 0;   // End of interrupt execution
}
```

PROGRAM NO.16

AIM: TO DISPLAY THE HEX DIGITS 0 TO F ON A 7-SEGMENT LED INTERFACE, WITH AN APPROPRIATE DELAY IN BETWEEN.



```
#include <LPC214x.H>
void delay_led(unsigned long int);
int main(void)
{
IO0DIR = 0x000007FC;
while(1)
{
IO0CLR = 0x00000FFF;
IO0SET = 0x00000604;
delay_led(15000000);
IO0CLR = 0x00000FFF;
IO0SET = 0x000007E4;
delay_led(15000000);
IO0CLR = 0x00000FFF;
IO0SET = 0x00000648;
delay_led(15000000);
IO0CLR = 0x00000FFF;
IO0SET = 0x00000618;
delay_led(15000000);
IO0CLR = 0x00000FFF;
IO0SET = 0x00000730;
delay_led(15000000);
IO0CLR = 0x00000FFF;
IO0SET = 0x00000690;
delay_led(15000000);
IO0CLR = 0x00000FFF;
IO0SET = 0x00000680;
delay_led(15000000);
IO0CLR = 0x00000FFF;
IO0SET = 0x0000063C;
delay_led(15000000);
IO0CLR = 0x00000FFF;
IO0SET = 0x00000600;
delay_led(15000000);
IO0CLR = 0x00000FFF;
IO0SET = 0x00000630;
delay_led(15000000);
IO0CLR = 0x00000FFF;
IO0SET = 0x00000620;
```



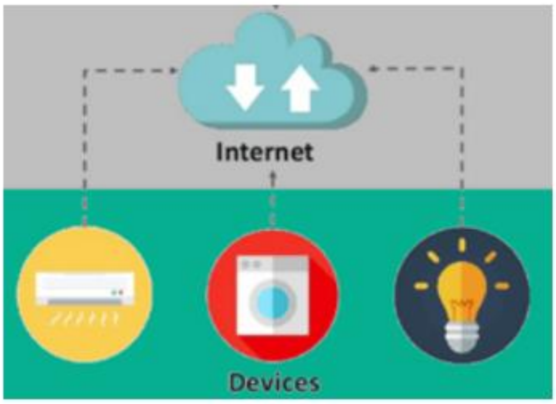
```
delay_led(15000000);
IO0CLR = 0x00000FFF;
IO0SET = 0x00000780;
delay_led(15000000);
IO0CLR = 0x00000FFF;
IO0SET = 0x000006C4;
delay_led(15000000);
IO0CLR = 0x00000FFF;
IO0SET = 0x00000708;
delay_led(15000000);
IO0CLR = 0x00000FFF;
IO0SET = 0x000006C0;
delay_led(15000000);
IO0CLR = 0x00000FFF;
IO0SET = 0x000006E0;
delay_led(15000000);
IO0CLR = 0x00000FFF;
}
}
void delay_led(unsigned long int count1)
{
while(count1 > 0) {count1--;}
}
```

PROGRAM NO.17

AIM: TO DEMONSTRATION THE IOT APPLICATIONS BY USING ARDUINO AND RASPBERRY PI.

Internet of Things (IoT) is a network of physical objects or people called “things” that are embedded with software, electronics, network, and sensors that allows these objects to collect and exchange data. The goal of IoT is to extend to internet connectivity from standard devices like computer, mobile, tablet to relatively dumb devices like a toaster.
IoT makes virtually everything “smart,” by improving aspects of our life with the power of data collection, AI algorithm, and networks. The thing in IoT can also be a person with a diabetes monitor implant, an animal with tracking devices, etc. This IoT tutorial for beginners covers all the Basics of IoT.

How IoT works?



How IoT Works

The entire IoT process starts with the devices themselves like smartphones, smartwatches, electronic appliances like TV, Washing Machine which helps you to communicate with the IoT platform.

Now in this IoT tutorial, we will learn about four fundamental components of an IoT system:

1) Sensors/Devices: Sensors or devices are a key component that helps you to collect live data from the surrounding environment. All this data may have various levels of complexities. It could be a simple temperature monitoring sensor, or it may be in the form of the video feed.

A device may have various types of sensors which performs multiple tasks **apart** from sensing. Example, A mobile phone is a device which has multiple sensors like GPS, camera but your smartphone is not able to sense these things.

) Connectivity: All the collected data is sent to a cloud infrastructure. The sensors should be connected to the cloud using various mediums of communications. These communication mediums include mobile or satellite networks, Bluetooth, WI-FI, WAN, etc.

3) Data Processing: Once that data is collected, and it gets to the cloud, the software performs processing on the gathered data. This process can be just checking the temperature, reading on devices like AC or heaters. However, it can sometimes also be very complex like identifying objects, using computer vision on video.

4)User Interface: The information needs to be available to the end-user in some way which can be achieved by triggering alarms on their phones or sending them notification through email or text message. The user sometimes might need an interface which actively checks their IoT system. For example, the user has a camera installed in his home. He wants to access video recording and all the feeds with the help of a web server.

However, it’s not always one-way communication. Depending on the IoT application and complexity of the system, the user may also be able to perform an action which may create cascading effects.

For example, if a user detects any changes in the temperature of the refrigerator, with the help of IoT technology the user should able to adjust the temperature with the help of their mobile phone.

IoT Applications

HOME
CONSUMER
TRANSPORT
MOBILITY
HEALTH
BODY
BUILDINGS
INFRASTRUCTURE
CITIES
INDUSTRY

IoT Applications

Application type	Description
Smart Thermostats	Helps you to save resource on heating bills by knowing your usage patterns.
Connected Cars	IoT helps automobile companies handle billing, parking, insurance, and other related stuff automatically.
Activity Trackers	Helps you to capture heart rate pattern, calorie expenditure, activity levels, and skin temperature on your wrist.
Smart Outlets	Remotely turn any device on or off. It also allows you to track a device’s energy level and get custom notifications directly into your smartphone.

Parking Sensors	IoT technology helps users to identify the real-time availability of parking spaces on their phone.
Connect Health	The concept of a connected health care system facilitates real-time health monitoring and patient care. It helps in improved medical decision-making based on patient data.
Smart City	Smart city offers all types of use cases which include traffic management to water distribution, waste management, etc.
Smart home	Smart home encapsulates the connectivity inside your homes. It includes smoke detectors, home appliances, light bulbs, windows, door locks, etc.
Smart supply chain	Helps you in real time tracking of goods while they are on the road, or getting suppliers to exchange inventory information.

VIVA QUESTIONS:

1. What is the processor used by ARM7?

a) 8-bit CISC b) 8-bit RISC

c) 32-bit CISC

d) 32-bit RISC

2. What is the instruction set used by ARM7?

a) 16-bit instruction set

b) 32-bit instruction set

c) 64-bit instruction set

d) 8-bit instruction set

3. How many registers are there in ARM7?

a) 35 register(28 GPR and 7 SPR)

b) 37 registers(28 GPR and 9 SPR)

c) 37 registers(31 GPR and 6 SPR)

d) 35 register(30 GPR and 5 SPR)

Explanation: ARM7TDMI has 37 registers(31 GPR and 6 SPR). All these designs use a Von Neumann architecture, thus the few versions comprising a cache do not separate data and instruction caches.

4. ARM7 has an in-built debugging device?

a) True

b) False

5. What is the capability of ARM7 f instruction for a second?

a) 110 MIPS

b) 150 MIPS

c) 125 MIPS

d) 130 MIPS

6. We have no use of having silicon customization?

a) True

b) False

7. Which of the following has the same instruction set as ARM7?

a) ARM6

b) ARMv3

c) ARM71a0

d) ARMv4T

8. What are t, d, m, I stands for in ARM7TDMI?

a) Timer, Debug, Multiplex, ICE

b) Thumb, Debug, Multiplier, ICE

c) Timer, Debug, Modulation, IS

d) Thumb, Debug, Multiplier, ICE

9. ARM stands for __________

a) Advanced RISC Machine

b) Advanced RISC Methadology

c) Advanced Reduced Machine

d) Advanced Reduced Methadology

10. What are the profiles for ARM architecture?

a) A,R b) A,M **c) A,R,M** d) R,M

11. ARM7DI operates in which mode?

a) Big Endian

b) Little Endian

c) Both big and little Endian

d) Neither big nor little Endian

12. In which of the following ARM processors virtual memory is present?

a) ARM7DI b) ARM7TDMI-S c) ARM7TDMI d) ARM7EJ-S

13. How many instructions pipelining is used in ARM7EJ-S?

a) 3-Stage b) 4-Stage **c) 5-Stage** d)2-stage

14. How many bit data bus is used in ARM7EJ-s?

a) 32-bit b) 16-bit c) 8- d) Both 16 and 32 bit

15. What is the cache memory for ARM710T?

a) 12Kb b) 16Kb c) 32Kb **d) 8Kb**