Varuvan Vadivelan Institute of Technology

Dharmapuri – 636 703.

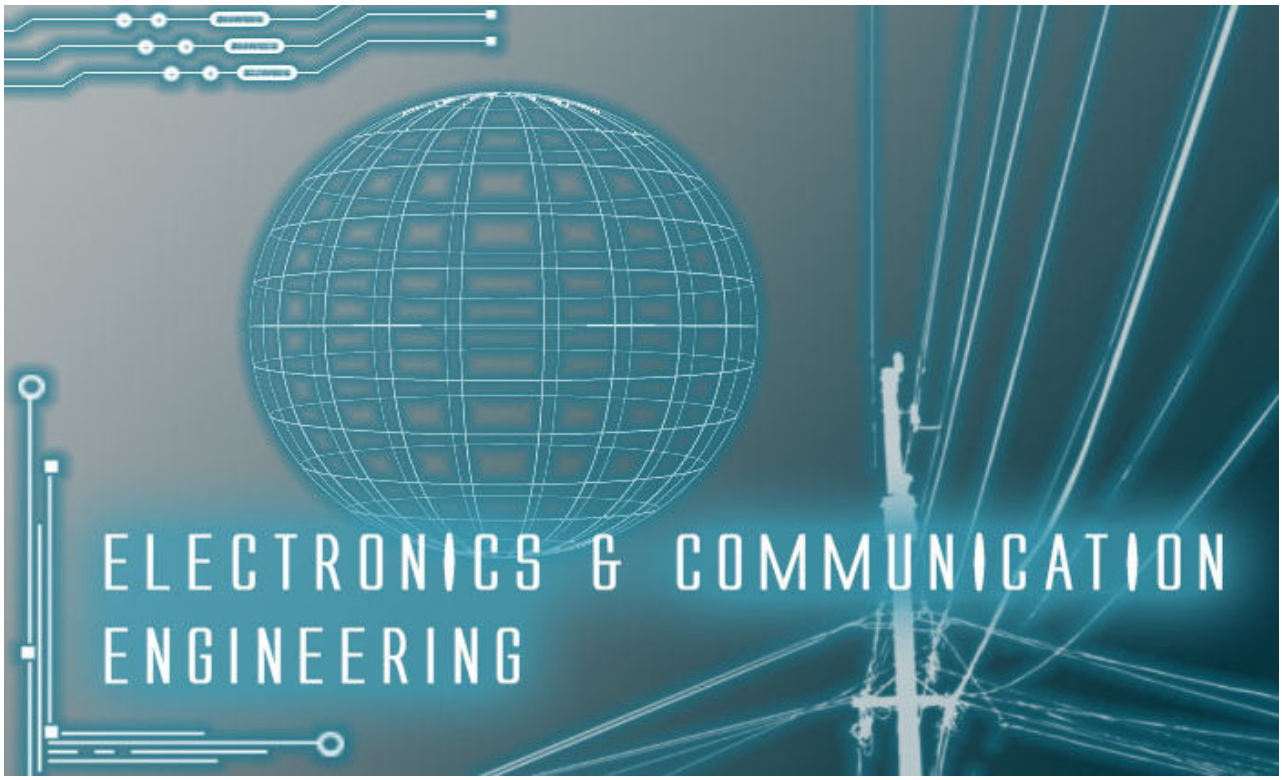
LAB MANUAL

Regulation : 2013

Branch : *B.E.* – ECE

Year & Semester : IV Year / VII Semester

EC6711- EMBEDDED LABORATORY



ANNA UNIVERSITY CHENNAI

Regulation 2013

EC6711 EMBEDDED LABORATORY

SYLLABUS

LIST OF EXPERIMENTS

1. Study of ARM evaluation system
2. Interfacing ADC and DAC.
3. Interfacing LED and PWM.
4. Interfacing real time clock and serial port.
5. Interfacing keyboard and LCD.
6. Interfacing EPROM and interrupt.
7. Mailbox.
8. Interrupt performance characteristics of ARM and FPGA.
9. Flashing of LEDS.
10. Interfacing stepper motor and temperature sensor.
11. Implementing zigbee protocol with ARM.

TOTAL: 45 Periods

INDEX

S.no	Date	Name of the Experiment	Page no	Marks	Signature
1		Study of ARM evaluation system			
2		Interfacing ADC and DAC.			
3		Interfacing LED and PWM			
4		Interfacing real time clock and serial port			
5		Interfacing keyboard and LCD			
6		Interfacing EPROM and interrupt			
7		Mailbox			
8		Interrupt performance characteristics of ARM and FPGA			
9		Flashing of LEDS			
10		Interfacing stepper motor and temperature sensor			
11		Implementing ZIGBEE protocol with ARM			

Ex. No: 1 Date:	STUDY OF ARM EVALUATION SYSTEM

AIM:

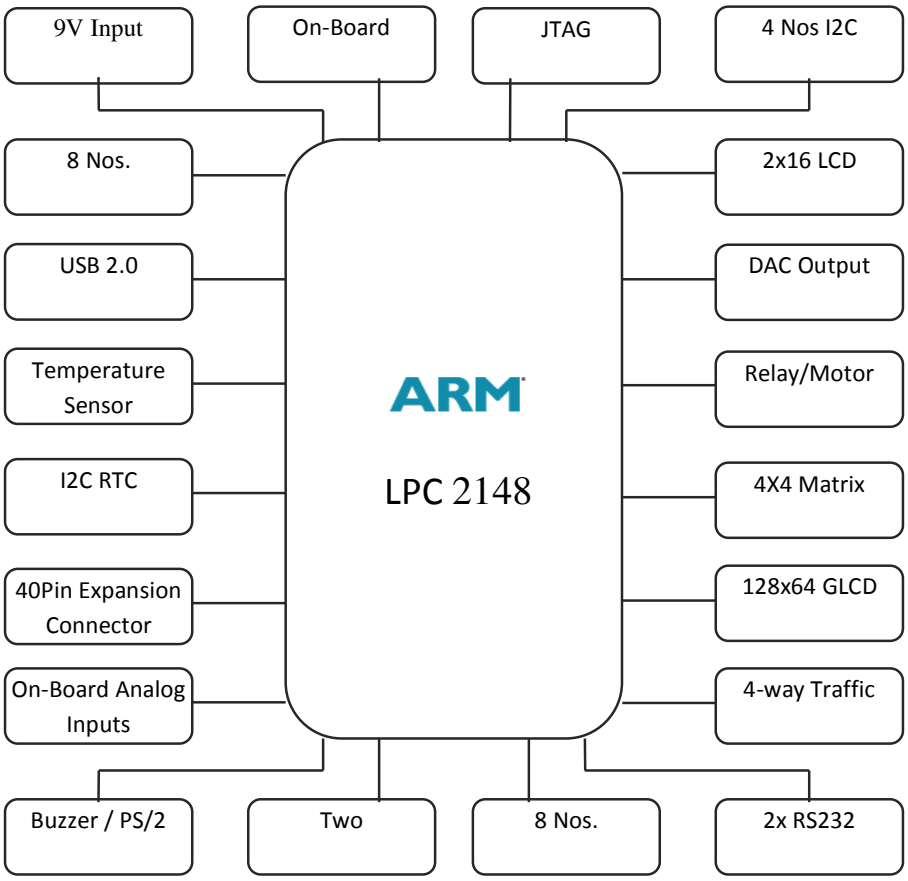
To study of ARM processor system and describe the features of architecture.

ARCHITECTURE OF ARM PROCESSOR:

1.1. Features of ARM DEVELOPMENT KIT Processor:

- 16-bit/32-bit ARM7TDMI-S microcontroller in a tiny LQFP64 package.8 kB to 40 kB of on-chip static RAM and 32 kB to 512 kB of on-chip flash memory. 128-bit wide interface/accelerator enables high-speed 60 MHz operation. In-System/In-Application Programming (ISP/IAP) via on-chip boot loader software.
- Single flash sector/full chip erase in 400 ms and programming of 256 bytes in 1 ms.USB 2.0 Full-speed compliant device controller with 2 kB of endpoint RAM. The LPC2146/48 provides 8 kB of on-chip RAM accessible to USB by DMA.
- One or two (LPC2141/42 vs. LPC2144/46/48) 10-bit ADCs provide a total of 6/14 analog inputs, with conversion times as low as 2.44 μs per channel. Single 10-bit DAC provides variable analog output (LPC2142/44/46/48 only).Two 32-bit timers/external event counters (with four capture and four compare channels each), PWM unit (six outputs) and watchdog.
- Low power Real-Time Clock (RTC) with independent power and 32 kHz clock input. Multiple serial interfaces including two UARTs (16C550), two Fast I2C-bus (400 kbit/s), SPI and SSP with buffering and variable data length capabilities.
- Vectored Interrupt Controller (VIC) with configurable priorities and vector addresses.Up to 45 of 5 V tolerant fast general purpose I/O pins in a tiny LQFP64 package.Up to 21 external interrupt pins available.
- 60MHz maximum CPU clock available from programmable on-chip PLL with settling time of 100μs.On-chip integrated oscillator operates with an external crystal from 1 MHz to 25 MHz.Power saving modes include Idle and Power-down.
- Individual enable/disable of peripheral functions as well as peripheral clock scaling for additional power optimization.Processor wake-up from Power-down mode via external interrupt or BOD.Single power supply chip with POR and BOD circuits:CPU operating voltage range of 3.0 V to 3.6 V (3.3 V ± 10 %) with 5 V tolerant I/O pads.

1.2. General Block Diagram:



1.3. Power Supply:

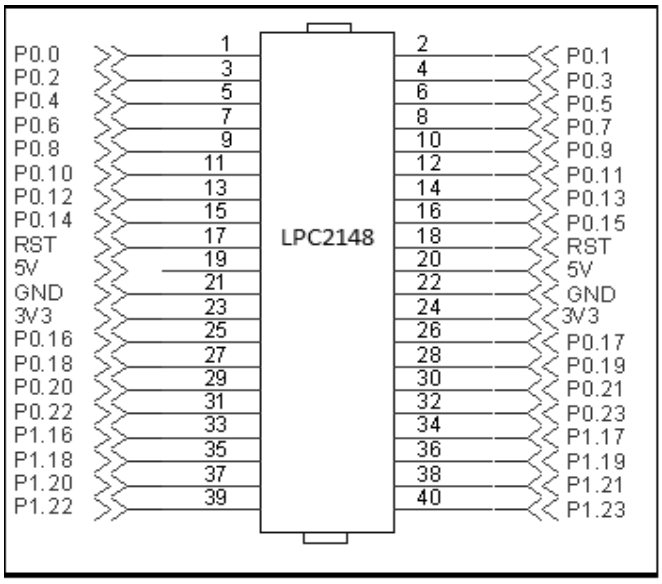
- The external power can be AC or DC, with a voltage between (9V/12V, 1A output) at 230V AC input.The ARM board produces +5V using an LM7805 voltage regulator, which provides supply to the peripherals.
- LM1117 Fixed +3.3V positive regulator used for processor & processor related peripherals.

1.4. Flash Programming Utility

- **NXP (Philips)**

NXP Semiconductors produce a range of Microcontrollers that feature both on-chip Flash memory and the ability to be reprogrammed using In-System Programming technology.

1.5. Pin Configuration:



1.6. On-board Peripherals:

- 8-Nos. of Point LED's (Digital Outputs)
- 8-Nos. of Digital Inputs (slide switch)
- 2 Lines X 16 Character LCD Display
- I2C Enabled 4 Digit Seven-segment display
- 128x64 Graphical LCD Display
- 4 X 4 Matrix keypad
- Stepper Motor Interface
- 2 Nos. Relay Interface
- Two UART for serial port communication through PC
- Serial EEPROM
- On-chip Real Time Clock with battery backup
- PS/2 Keyboard interface(Optional)
- Temperature Sensor
- Buzzer(Alarm Interface)
- Traffic Light Module(Optional)

RESULT:

Thus the study of ARM processor has been done and ensured its composition with internal features specifically.

Ex. No:2	INTERFACING ADC AND DAC
Date:	

AIM:

To develop and verify the interfacing ADC and DAC with LPC 2148 ARM microcontroller.

APPARATUS REQUIRED:

S.No	Apparatus	Range	Quantity
1	ARM Development Kit	LPC 2148	1
2	Keil µVision3 IDE	-	1

PROCEDURE:

Step 1: Give a double click on µvision 4 icon on the desk top, it will generate a window as shown below

Step 2: To create new project go to project select new micro vision project.

Step 3: select a drive where you would like to create your project.

Step 4: Create a new folder and name it with your project name.

Step 5: Open that project folder and give a name of your project executable and save it.

Step 6: After saving it will show some window there you select your microcontroller company i.e. NXP from Phillips

Step 7: Select your chip as ARM DEVELOPMENT KIT

Step 8: After selecting chip click on OK then it will display some window asking to add STARTUP. Select YES.

Step 9: A target is created and startup is added to your project target and is shown below.

Step 10: To write your project code select a new from menu bar.

Step 11: It will display some text editor, to save that select SAVE option from menu bar.

Step 12: By giving a name with extension .c and save it.

Step 13: Write the code of your project and save it.

Step 14: To add the c to target give a right click on Source Group, choose "ADD s to Group" option.

Step 15: It will display some window there select the file and click on ADD option.

Step 16: It will be added to our target and it shows in the project window.

Step 17: Select right click on target in the project window and select "Options for Target."

Step 18: It will show some window, in that go to output option and choose Create Hex option by selecting that box.

Step 20: Now to compile your project go to Project select Build Target option or press F7.

Step 21: Check the concern block of output and observe the results.

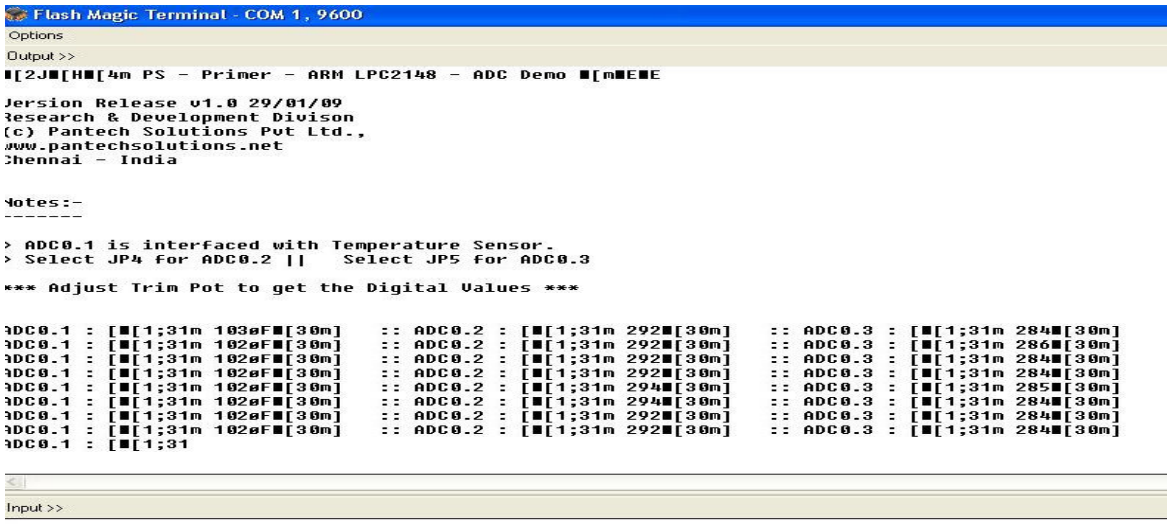
PROGRAM:

```
#include <lpc214x.h>
#include <stdio.h>
#define ESC 0x1B
#define DONE   0x80000000
#define START  0x01000000
#define PRESET 0x00230600
void serial_init(void)
{
  PINSEL0 = 0x00000005;            /* Enable RxD0 and TxD0
     */
  U0LCR = 0x83;                    /* 8 bits, no Parity, 1
Stop bit          */
  U0DLL = 195;                        /* 9600 Baud Rate @
12MHz VPB Clock       */
  U0LCR = 0x03;                   /* DLAB = 0
                          */
}
void delay(int n)
{
     int i,j;
     for(i=0;i<n;i++)
     {
          for(j=0;j<0x5000;j++)
          {;}
     }}
void main(void)
{
     unsigned long val[4];
     unsigned int ADC_CH;
     unsigned char i=0;
     PINSEL0 = 0x00000005;         //Enable RXD0 and TXD0
```

```
    PINSEL1   |=   0x01 << 24;          //Enable ADC0.1
    PINSEL1   |=   0x01 << 26;
    PINSEL1   |=   0x01 << 28;
    //PINSEL1 =    0x15000000;          //Enable ADC0.1 |
ADC0.2 | ADC0.3
    VPBDIV    =    0x02;                //Set the cclk to
30 Mhz
    AD0CR     =    0x00230602;          //ADC
    configuration bits CLK =     10clks/9Bit | BURST=1 |
    CLKDIV = 0x06
    AD0CR     |=   0x01000000;          //start ADC now
    //IO0DIR  =    0x0FFF7030;
    serial_init();                 //serial
initialization
    ADC_CH    =    1;
    printf("%c[2J%c[H",ESC,ESC);
    printf("%c[4m PS - Primer - ARM ARM DEVELOPMENT KIT
    -   ADC   Demo        %c[m%cE%cE",ESC,ESC,ESC,ESC);/*
    underline */
    printf("\n\nVersion Release v1.0 29/01/09\n");
    printf("Research & Development Divison\n");
    printf      ("(c)      Pantech      Solutions      Pvt
Ltd.,\nwww.pantechsolutions.net\n");
    printf ("Chennai - India\n");
    printf("\n\nNotes:-\n-------\n\n>      ADC0.1      is
    interfaced with Temperature   Sensor.\n> Select JP4
    for ADC0.2     ||   Select JP5 for ADC0.3\n\n");
    printf  ("*** Adjust  Trim  Pot  to  get  the  Digital
Values ***\n\n\n");
while(1)
{
    while (ADC_CH <4)   {
    do
    {
        val[ADC_CH] = AD0GDR;                    //  Read
A/D Data Register
              }
                    while  ((val[ADC_CH]  &  0x80000000)
== 0);
                    val[ADC_CH] = ((val[ADC_CH] >> 6) &
0x03FF);
                    ADC_CH++;
              delay(10);
              AD0CR     =    PRESET | (1<<ADC_CH);
              AD0CR     |=   START;
         }
         val[1]    =    (AD0DR1 >> 6) & 0x03FF;
         val[2]    =    (AD0DR2 >> 6) & 0x03FF;
         val[3]    =    (AD0DR3 >> 6) & 0x03FF;
         for (i=1;i<4;i++)
         {
```

```
                    delay(1);
                    printf("ADC0.%d : [",i);
                    putchar(0x1B);
                    printf("[1;31m%4d", val[i]);
                    if (i==1)
                    {
                         printf ("\xF8\F")  ;
                    }
                    putchar (0x1B);
                    printf ("[30m]  ");
                    if (i<3)
                    {
                         printf(" :: ");
                    }
                    delay(1);
               }
               if(ADC_CH > 3/*The number of channels used in
     PS-ARMDPK*/)
               {
                    ADC_CH    =    1;
                    AD0CR     =    PRESET | (1<<ADC_CH);
                    AD0CR     |=   START;
                    //printf ("\b\b");
                    U0THR     =    '\r';
                    }
     } }
```

OUTPUT:



RESULT:

Thus the interfacing of ADC and DAC (In-built) with ARM processor has been verified and observed the output successfully.

Ex. No:3	INTERFACING LED AND PWM.
Date:	

AIM:

To develop and verify the interfacing LED and PWM with ARM DEVELOPMENT KIT microcontroller using embedded c program.

APPARATUS REQUIRED:

S.No	Apparatus	Range	Quantity
1	ARM Development Kit	LPC 2148	1
2	Keil µVision3 IDE	-	1

PROCEDURE:

Step 1: Give a double click on µvision 4 icon on the desk top, it will generate a window as shown below

Step 2: To create new project go to project select new micro vision project.

Step 3: select a drive where you would like to create your project.

Step 4: Create a new folder and name it with your project name.

Step 5: Open that project folder and give a name of your project executable and save it.

Step 6: After saving it will show some window there you select your microcontroller company i.e NXP from Phillips

Step 7: Select your chip as LPC 2148.

Step 8: After selecting chip click on OK then it will display some window asking to add STARTUP. Select YES.

Step 9: A target is created and startup is added to your project target and is shown below.

Step 10: To write your project code select a new from menu bar.

Step 11: It will display some text editor, to save that select SAVE option from menu bar.

Step 12: By giving a name with extension .C and save it

Step 13: Write the code of your project and save it.

Step 14: To add our c to target give a right click on Source Group, choose "ADD s to Group" option.

Step 15: It will display some window there select the file and click on ADD option.

Step 16: It will be added to our target and it shows in the project window.

Step 17: Select a right click on target in the project window and select "Options for Target"

Step 18: It will show some window, in that go to output option and choose Create Hex option by selecting that box

Step 19: In the same window go to Linker option and choose Use Memory Layout from Target Dialog by selecting the box, and click OK.

Step 20: Now to compile your project go to Project select Build Target option or press F7.

Step 21: Check the output of LED's as switching from ON to OFF.

PROGRAM:

LED:

```
#include <LPC214x.h>
#include <stdio.h>
#define LED 16
#define Switch 24
void Delay(int);
int main(void)
{
     unsigned char Status=1;
     PINSEL2   &=   0xFFFFFFF3;          //Configure P1.16
- P1.31 as GPIO
     IO1DIR    =    0x00 << Switch;     //Configure P1.24
- P1.31 as Input
     IO1DIR    |=   0xFF << LED;   //Configure P1.16 -
P1.23 as Output
     IO1PIN    =    0;
     while(1)
     {
          Status = 1;
          IOSET1    =    0xFF << LED;
          Delay (10);Delay (10);
          IOCLR1    =    0xFF << LED;
          Delay (10);Delay (10);Delay (10);
          while (~Status)
          {
               Status    =    ((IO1PIN & (0xFF <<
Switch)) >> Switch);
               IO1PIN    =    ((~Status) << LED);
```

```
          }    }            }
    void Delay(int n)
    {
         int p,q;
         for(p=0;p<n;p++)
         {
              for(q=0;q<0x9990;q++);
         }}
```

PWM:

```
    #include <LPC214x.H>                        /* LPC21xx
    definitions  */
    #include <stdio.h>
    void PWM0_isr(void)  __irq
    {
    PWMIR       |= 0x00000001;               /* Clear match0
    interrupt */
    VICVectAddr  = 0x00000000;
    }
    void init_PWM (void) {
    VICVectAddr8 = (unsigned)PWM0_isr;        /* Set the PWM
    ISR vector address */
    VICVectCntl8 = 0x00000028;                /* Set channel
    */
    VICIntEnable = 0x00000100;                /* Enable the
    interrupt */
    PINSEL0 |= 0x00008008;                    /* Enable P0.7
    and P0.1 as PWM output */
    PWMPR    = 0x00000000;                    /* Load
    prescaler  */
    PWMPCR = 0x00000C0C;      /* PWM channel 2 & 3 double
    edge control, output enabled */
    PWMMCR = 0x00000003;               /* On match with timer
    reset the counter */
    PWMMR0 = 0x400;                           /* set cycle
    rate to sixteen ticks       */
    PWMMR1 = 0;                               /* set rising
    edge of PWM2 to 100 ticks    */
    PWMMR2 = 0x200;                           /* set falling
    edge of PWM2 to 200 ticks   */
    PWMMR3 = 0x100;                           /* set rising
    edge of PWM3 to 100 ticks    */
    PWMLER = 0xF;                             /* enable
    shadow latch for match 1 - 3   */
    PWMTCR = 0x00000002;                      /* Reset
    counter and prescaler           */
    PWMTCR = 0x00000009;                      /* enable
    counter and PWM, release counter from reset */
    }void Delay ()
    {
         unsigned int i,j;
         for (i=0;i<250;i++)
```

```
          for (j=0;j<700;j++);
}
int main (void)
{    unsigned long val;
     unsigned long oldval = 0;
     VPBDIV    =    0x02;
     PINSEL0   |=   0x00050000;
     PINSEL1   =    0x15400000;
     init_PWM();
     U1LCR     =    0x83;
     U1DLL     =    0xC3;
     U1LCR     =    0x03;
AD0CR   = 0x00230608;                    /* Setup A/D:
10-bit AIN0 @ 3MHz */
AD0CR |= 0x01000000;                     /* Start A/D
Conversion */
     while (1)
     {                                                /*
Loop forever */
          do
          {
          val = AD0GDR;      /* Read A/D Data Register */
          } while ((val & 0x80000000) == 0);       /*
Wait for end of A/D Conversion */
          val = ((val >> 6) & 0x3FF);               /*
Extract AIN0 Value */
          Delay (); Delay();
          if (val != oldval)
          {
               PWMMR2    =    val;
               PWMLER    =    0xF;
               oldval    =    val;
               printf ("Val : %4d \n\r", val);
          }
     }
}
```

OUTPUT:

IODIR =ox00 (OFF)
IODIR =oxFF (ON)
Delay =10

RESULT:

Thus the interfacing of LED and PWM with ARM DEVELOPMENT KIT was done by using embedded C program.

Ex. No:4	INTERFACING OF REAL TIME CLOCK AND SERIAL PORT
Date:	

AIM:

To develop and verify the interfacing of real time clock and serial port with ARM DEVELOPMENT KIT microcontroller using embedded C program.

APPARATUS REQUIRED:

S.No	Apparatus	Range	Quantity
1	ARM Development Kit	LPC2148	1
2	Keil µVision3 IDE		1

PROCEDURE:

Step 1: Give a double click on µvision 4 icon on the desk top, it will generate a window as shown below

Step 2: To create new project go to project select new micro vision project.

Step 3: select a drive where you would like to create your project.

Step 4: Create a new folder and name it with your project name.

Step 5: Open that project folder and give a name of your project executable and save it.

Step 6: After saving it will show some window there you select your microcontroller company i.e NXP from Phillips

Step 7: Select your chip as ARM DEVELOPMENT KIT

Step 8: After selecting chip click on OK then it will display some window asking to add STARTUP . Select YES.

Step 9: A target is created and startup is added to your project target and is shown below.

Step 10: To write your project code select a new from menu bar.

Step 11: It will display some text editor, to save that select SAVE option from menu bar.

Step 12: By giving a name with extension .c and save it

Step 13: To add our c to target give a right click on Source Group, choose “ADD s to Group” option.

Step 14: It will displays some window there select the you have to add and click on ADD option.

Step 15: It will be added to our target and it shows in the project window.

Step 16: Now give a right click on target in the project window and select "Options for Target"

Step 17: It will show some window, in that go to output option and choose Create Hex option by selecting that box

Step 18: In the same window go to Linker option and choose Use Memory Layout from Target Dialog by selecting the box, and click OK.

Step 19: Now to compile your project go to Project select Build Target option or press F7.

Step 20: Ensure the real time clock, displaying in output window.

PROGRAM:

RTC:

```
#include <LPC213x.h>
#include <stdio.h>
#include <ctype.h>
#include <stdlib.h>
#include "UART.h"
#define BUZZ   7
     //Buzzer Connected to P0.7
void UART1_ISR(void)__irq;
void RTC_ISR (void)__irq;
unsigned char Flag=0;
char Key;
void DelayMs(long ms)  // delay 1 ms per count @ CCLK 60 MHz
{
     long i,j;
     for (i = 0; i < ms; i++ )
     for (j = 0; j < 6659; j++ );
}void Initialize(void)
{
PLLCFG    =    0x24;     // M : 4  | P = 2  -> Fosc = 12MHz &
CCLK = 60MHz
PLLFEED   =    0xAA;               // Feed Process
PLLFEED   =    0x55;
PLLCON    =    0x01;
PLLFEED   =    0xAA;               // Feed Process
PLLFEED   =    0x55;
while (!(PLLSTAT & 0x400));   //Wait untill PLL is Locked!
PLLCON    =    0x03;     //Connect PLL as the Clock Source for
Microcontroller
PLLFEED   =    0xAA;               // Feed Process
```

```
PLLFEED   =    0x55;
MAMCR     =    0x02;     //Memory Accerleration Module Fully
Enabled
MAMTIM    =    0x04;          //MAM fetch cycles are 4 CCLKs
in duration
VPBDIV    =    0x02;               //Divide Clock for PCLK =
30MHz
}
void RTC_Setup(char *Buff)
{
     unsigned char TimE;
     char i=0;
     for(i=0;i<2;i++)
     {
          while(!isdigit(Key));                          //Wait
till Key = 0 to 9
          if (i==0)
          {
               TimE =    10 * atoi( &Key );
          }
          if (i==1)
          {
               TimE +=   atoi( &Key );
          }
          putchar(Key);
          Key  =    0;
     }
     *Buff     =    TimE;               /Load Setup New Value
}
void Delay()
{
     unsigned int i,j;
     for(i=0;i<50;i++)
          for(j=0;j<700;j++);
}
void Wait()
{
     Delay();Delay();Delay();
     Delay();Delay();Delay();
     Delay();Delay(); Delay();
}
void Alarm(void)
{
     IOSET0 = 1 << BUZZ;
     Wait();Wait();
     IOCLR0 = 1 << BUZZ;
     Wait();
}
//void Clean(void)
//{
//   unsigned char i;
```

```
//   //for(i=0;i<250;i++)
//        printf("[2M");
//}
void main(void)
{
     Initialize();
UART1_Init(9600/*Baud Rate*/);
U1IER     =    3;//Enable UART1 Recieve Interrupt
//PINSEL0 |=   (1 << 18);     //Select Pin as UART1
IO0DIR    |=   (1<<7);   //Configure P0.7 as O/p (Buzzer)
VICVectAddr0   =    (unsigned)UART1_ISR;
VICVectCntl0   =    0x20 | 7;
VICIntEnable   |=   (1 << 7);
VICVectAddr2   =    (unsigned)RTC_ISR;
VICVectCntl2   =    0x20 | 13;
VICIntEnable   |=   (1 << 13);
AMR  =    0xFF;     //Mask all valued except hh:mm:ss for
alarm comparision
PREINT = 0x00000392;     // Set RTC Prescaler for PCLK 30 MHz
   PREFRAC = 0x00004380;//    printf("[2J\0");// Clear Screen
     CCR       =    0x01;
     //CIIR    =    0x01;
     UART1_PutS("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n");
     UART1_PutS("  ARM LPC2138 RTC Demo\n\r------------------
-\n\n\n");
     UART1_PutS("> Press * to Set Time\n");
     UART1_PutS("> Press ! to Set Alarm\n");
     UART1_PutS("> Press $ to Snooze Alarm 5 Minutes\n");
     UART1_PutS("> Press . to Stop Alarm\n");
     UART1_PutS("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n\n");
     while(1)
     {
     printf("CTC : %d\t",CTC);
     printf(">>  TIME: %02d:%02d:%02d    \r",HOUR,MIN,SEC); //
Display time format hh:mm:ss
     DelayMs(100);                       // Delay for display
          if (Key == '*')
          {
               Key  =    0;
               UART1_PutS(">>    Set Time:   ");
               RTC_Setup(&HOUR);
               UART1_PutC(':');
               RTC_Setup (&MIN);
               UART1_PutC (':');
               RTC_Setup (&SEC);
//printf("\r\tTIME: %02d:%02d:%02d       \r",HOUR,MIN,SEC);
     // Display time format hh:mm:ss
     //printf("^[2J");
     U1THR     =    0x1B;     //Escape
     UART1_PutS("[2J\0");          // Clear Screen
          }
```

```
          if (Key == '!')
          {
               AMR  =    0xF8;
               Key = 0;
               UART1_PutS(">>\tSet Alarm: ");
               RTC_Setup(&ALHOUR);
               UART1_PutC(':');
               RTC_Setup(&ALMIN);
               UART1_PutC(':');
               RTC_Setup(&ALSEC)
          }
          if (Key == '$' && Flag == 1)
          {
               if (MIN >= 55)
               {
                    ALHOUR    =    HOUR + 1;
                    ALMIN     =    5    -    (60 - MIN);
               }
               else
               {
                    ALMIN     =    MIN + 5;
               }
               Key  =    0;
               Flag =    0;
          }
          if (Key == '.')
          {
               Key       =    0;
               Flag =    0;
          }
          if (Flag == 1)
          {
               Alarm (); Wait (); Alarm ();
          }
     }
}
void UART1_ISR(void)__irq
{
     char Msg;
     if(((Msg = U1IIR) & 0x01) == 0)//Check Flag Status of
                                        Recieve Interrupt
     {
          switch(Msg & 0x0E)       //Filter Msg
          {
               case 0x04: while (!(U1LSR & 0x20));
                                              //Recieve Data
               Key = U1RBR;
               case 0x02:     break;  // Interrupt
               default    :   break;
          }
     }
```

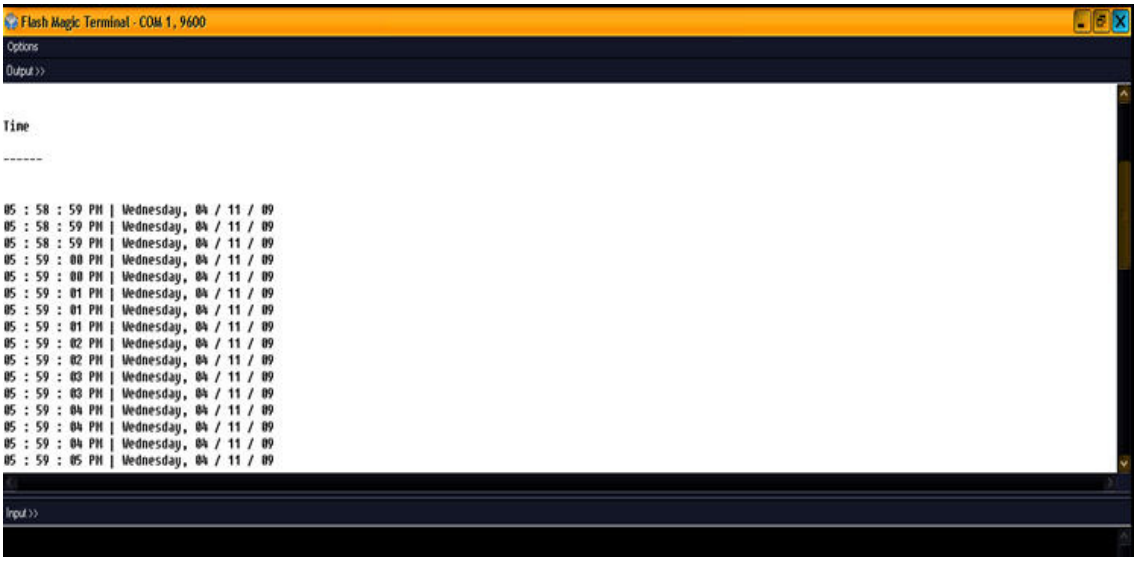
```
     VICVectAddr    =    0;         //Acknowledge Interrupt
}
void RTC_ISR (void)__irq
{
     if ((ILR & 0x02) == 0x02)
     {
          Flag = 1;
          UART1_PutS("\nALARM\n");
          ILR = 0x02;               //Clear Current Interrupt
     }

     VICVectAddr    =    0;              //Ack Interrupt
}
```

SERIAL PORT:

```
#define CR     0x0D
#include <LPC21xx.H>
void init_serial (void);
int putchar (int ch);
int getchar (void);
unsigned char test;
//<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<   Code Begins Here
int main(void)
{
     char *Ptr = "*** UART0 Demo ***\n\n\rType Characters to
be echoed!!\n\n\r";
     VPBDIV = 0x02;                            //Divide Pclk by
two
     init_serial()
     while(1)
     {
          while (*Ptr)
          {
               putchar(*Ptr++);
          }
          putchar(getchar());                       //Echo
terminal
     }
}
void init_serial (void)              /* Initialize Serial
Interface       */
{
  PINSEL0      = 0x00000005;         /* Enable RxD0 and TxD0
*/
  U0LCR   = 0x00000083;   /* 8 bits, no Parity, 1 Stop bit */
  U0DLL   = 0x000000C3; /*9600 Baud Rate @ 30MHz VPB Clock */
  U0LCR   = 0x00000003;  /* DLAB = 0 */
}
```

```
int putchar (int ch)  /* Write character to Serial Port */
{
  if (ch == '\n')  {
    while (!(U0LSR & 0x20));
    U0THR = CR;                    /* output CR */
  }
  while (!(U0LSR & 0x20));
  return (U0THR = ch);
}
int getchar (void)                   /* Read character from
Serial Port   */
{
  while (!(U0LSR & 0x01));
  return (U0RBR);

}
```

OUTPUT:



RESULT:

Thus the real time clock and serial port interfacing with ARM DEVELOPMENT KIT processor has been executed successfully.

Ex. No:5	INTERFACING KEYBOARD AND LCD
Date:	

AIM:

To develop and verify the interfacing of keyboard and LCD with ARM DEVELOPMENT KIT ARM microcontroller using embedded C program.

APPARATUS REQUIRED:

S.No	Apparatus	Range	Quantity
1	ARM DEVELOPMENT KIT		1
2	Keil µVision3 IDE		1

PROCEDURE:

Step 1: Give a double click on µvision 4 icon on the desk top, it will generate a window as shown below

Step 2: To create new project go to project select new micro vision project.

Step 3: select a drive where you would like to create your project.

Step 4: Create a new folder and name it with your project name.

Step 5: Open that project folder and give a name of your project executable and save it.

Step 6: After saving it will show some window there you select your microcontroller company i.e NXP from Phillips

Step 7: Select your chip as ARM lpc 2148.

Step 8: After selecting chip click on OK then it will display some window asking to add STARTUP. Select YES.

Step 9: A target is created and startup is added to your project target and is shown below.

Step 10: To write your project code select a new from menu bar.

Step 11: It will display some text editor, to save that select SAVE option from menu bar.

Step 12: By giving a name with extension .C and save it

Step 13: Write the code of your project and save it.

Step 14: To add our c to target give a right click on Source Group, choose "ADD s to Group" option.

Step 15: It will displays some window there select the you have to add and click on ADD option.

Step 16: It will be added to our target and it shows in the project window.

Step 17: Now give a right click on target in the project window and select "Options for Target"

Step 18: It will show some window, in that go to output option and choose Create Hex option by selecting that box

Step 19: In the same window go to Linker option and choose Use Memory Layout from Target Dialog by selecting the box, and click OK.

Step 20: Now to compile your project go to Project select Build Target option or press F7.

Step 21: Ensure the output of keyboard and display as by pressing the keys simultaneously

PROGRAM:

KEYBOARD:

```
#include <LPC214x.h>
#include <stdio.h>
#include "Keypad.h"
#include "UART_Utility.c"
extern void Delay(void);
unsigned char Row_Data, Col_Data;
unsigned char Msg[4][4] =     { '0','1','2','3',
                                  '4','5','6','7',
                                  '8','9','A','B',
                                  'C','D','E','F'
                                };void Delay(void)
            {
     unsigned int i,j;
     for(i=0;i<35;i++)
          for(j=0;j<1234;j++);
}
void main(void)
{
     VPBDIV    =    0x02;
     UART0_Init (9600);
     PINSEL2   |=   0x0;
     UART0_PutS ("\nPS-primer ARM ARM DEVELOPMENT KIT  Keypad
Demo\n\r");
```

```
      UART0_PutC (0xB8);
      UART0_PutS (" Pantech Solutions Pvt Ltd., \n\r");
      UART0_PutS (" www.pantechsolutions.net\n\r");
      UART0_PutS ("------------------------\n\n\r");
      UART0_PutS ("Keypad Ports : P1.24 - P1.31 \n\n\n\r");
      UART0_PutS ("~~~~~~~~~~~~~~~~~~~~\n\n\r");
      while (1)
      {
      Delay();
      Delay();
      KeyScan(&IOPIN1/*KeyPad Port*/,24/*Starting DataPin D0*/,
      &Row_Data/*Addr of Row*/,&Col_Data/*Addr of Col*/);
      UART0_PutS ("The Key You Pressed is :   ");
           if (Row_Data < 4 && Col_Data < 4)
           {
                U0THR = Msg[Row_Data][Col_Data];
                Delay();
                Delay();
                U0THR = '\r';
           }
      }
}
```

LCD:

```
      #include <LPC214x.H>
      #define RS     0x10000
      #define RW     0x20000
      #define EN     0x40000

      void lcd_cmd  (unsigned char);
      void lcd_data (unsigned char);
      void lcd_initialize (void);
      void lcd_display (void);
      void LCD4_Convert(unsigned char);
      const unsigned char cmd[4] = {0x28,0x0c,0x06,0x01};
                                              //lcd commands
      unsigned char msg[] = {">PS-Primer 2148<"};  //msg
      unsigned char msg1[]= {"::  LCD Demo! ::"};  //msg1
      void delay(unsigned int n)
      {
           unsigned int i,j;
           for(i=0;i<n;i++)
           {
                for(j=0;j<12000;j++)
                {;}
```

```
        }
    }
    void lcd_cmd(unsigned char data)
    {
              IOCLR0    |= RS;          //0x1000; //RS
                   IOCLR0    |= RW;    //0x2000; //RW
        LCD4_Convert(data);
    }
    void lcd_initialize(void)
    {
        int i;
        for(i=0;i<4;i++)
        {
              IOCLR0 = 0xF << 19;       //IOCLR 0/1
              lcd_cmd(cmd[i]);
              delay(15);
        }
    }
    void lcd_data (unsigned char data)
    {
        IOSET0    |= RS;          //0x1000;       //RS
        IOCLR0    |= RW;          //0x2000;       //RW
        LCD4_Convert(data);
    }
    void lcd_display (void)
    {
        char i;
        /* first line message */
        lcd_cmd(0x80);
        delay(15);
        i=0;
        while(msg[i]!='\0')
        {
              delay(5);
              lcd_data(msg[i]);
              i++;
        }
        delay(15);

        /* second line message */

        lcd_cmd(0xc0);
        delay(15);
        i=0;
```

```
        while(msg1[i]!='\0')
        {
              delay(5);
              lcd_data(msg1[i]);
              i++;
        }
        delay(15);
}
void LCD4_Convert(unsigned char c)
{
if(c & 0x80) IOSET0 = 1 << 22; else IOCLR0 = 1 << 22;
if(c & 0x40) IOSET0 = 1 << 21; else IOCLR0 = 1 << 21;
if(c & 0x20) IOSET0 = 1 << 20; else IOCLR0 = 1 << 20;
if(c & 0x10) IOSET0 = 1 << 19; else IOCLR0 = 1 << 19;
        IOSET0    = EN;     //0x4000; //EN delay(8);
        IOCLR0    = EN;     //0x4000; //EN
if(c & 0x08) IOSET0 = 1 << 22; else IOCLR0 = 1 << 22;
if(c & 0x04) IOSET0 = 1 << 21; else IOCLR0 = 1 << 21;
if(c & 0x02) IOSET0 = 1 << 20; else IOCLR0 = 1 << 20;
if(c & 0x01) IOSET0 = 1 << 19; else IOCLR0 = 1 << 19;
        IOSET0    = EN;
        //0x4000;      //EN
        delay(8);
        IOCLR0    = EN;
        //0x4000;      //EN
}
void main()
{
        PINSEL1   =    0;
        //Configure P0.16 - P0.31 as GPIO
        IODIR0    =    0xFF << 16;
        //Configure Pins P0.16 - P0.22 as Output Pins
        lcd_initialize();
        //Initialize LCD!
        lcd_display();
        //Display Message in LCD
        while(1);
}
```

OUTPUT:

RESULT:

Thus the interfacing of keyboard and LCD with ARM DEVELOPMENT KIT microcontroller using embedded C program has been verified successfully.

Ex. No:6 Date:	INTERFACING EPROM AND INTERRUPT

AIM:

To verify the interfacing of EPROM and interrupt with ARM DEVELOPMENT KIT processor using embedded C program.

APPARATUS REQUIRED:

S.No	Apparatus	Range	Quantity
1	ARM DEVELOPMENT KIT	LPC2148	1
2	Keil µVision3 IDE		1

PROCEDURE:

Step 1: Give a double click on µvision 4 icon on the desk top, it will generate a window as shown below

Step 2: To create new project go to project select new micro vision project.

Step 3: select a drive where you would like to create your project.

Step 4: Create a new folder and name it with your project name.

Step 5: Open that project folder and give a name of your project executable and save it.

Step 6: After saving it will show some window there you select your microcontroller company i.e NXP from Phillips

Step 7: Select your chip as ARM DEVELOPMENT KIT

Step 8: After selecting chip click on OK then it will display some window asking to add STARTUP . Select YES.

Step 9: A target is created and startup is added to your project target and is shown below.

Step 10: To write your project code select a new from menu bar.

Step 11: It will display some text editor, to save that select SAVE option from menu bar.

Step 12: By giving a name with extension .C and save it

Step 13: Write the code of your project and save it.

Step 14: To add our c to target give a right click on Source Group, choose "ADD s to Group" option.

Step 15: It will displays some window there select the you have to add and click on ADD option.

Step 16: It will be added to our target and it shows in the project window.

Step 17: Now give a right click on target in the project window and select "Options for Target"

Step 18: It will show some window, in that go to output option and choose Create Hex option by selecting that box

Step 19: In the same window go to Linker option and choose Use Memory Layout from Target Dialog by selecting the box, and click OK.

Step 20: Now to compile your project go to Project select Build Target option or press F7.

Step 21: Check the memory (EPROM) status with the help of interrupt execution and verify the operation.

PROGRAM:

EPROM:

```
#include <LPC214x.h>
#include <stdio.h>
#include <string.h>
#include "UART.h"
#define SW3          1<<24
#define SW4          1<<25
#define SW5          1<<26
#define SW6          1<<27
#define SW7          1<<28
#define MAX          12
#define   AA         2
#define   SI         3
#define   STO        4
#define   STA        5
#define   I2EN 6
void I2C_ISR(void)__irq;
void Wait (unsigned int);
void I2C_Init (void);
int I2C_Start (unsigned int Slave_Addr);
int I2C_Write (unsigned char *Buff, unsigned int Count);
char ReLoad[MAX]    =
{
0x00/*Address Low Bits*/,0x00/*Address Low
 Bits*/,'A','R','M','7','P','R','I', 'M', 'E', 'R'};
```

```
char Buff[MAX]      =    {0x00/*Address Low
Bits*/,0x00/*Address Low Bits*/,'A','R','M','7','P','R','I',
'M', 'E', 'R'};
unsigned char Rec[MAX]   =    {"NO-DATA!"};
unsigned char index      =    0;
unsigned char flag = 0, ii, Ready=0, Erase = 0;
void Delay(void)
{
     unsigned int i,j;
     for(i=0;i<150;i++)
          for(j=0;j<900;j++);
}
void Wait (unsigned int Delay)
{
     while(Delay--);
}
void I2C_Init (void)
{
     I2C1CONCLR     =    1 << STO;
     I2C1CONSET     =    1 << AA;
}
int I2C_Write (unsigned char *Buff, unsigned int Count)
{
int main()
{
     unsigned int i;
     VPBDIV    =    0x02;
     PINSEL0   =    0x30C00005;// P0.11-PINSEL2&=0xFFFFFFF3;
     IO1DIR    =    0x00 << SW3;
     IO1DIR    |=   0xFF << 16;
     UART0_Init (9600);
     VICIntSelect   = 0<<19;
     VICVectCntl0   = 0x020 | 19 ;
     VICVectAddr0   = (unsigned long)I2C_ISR;
     VICIntEnable   = 1<<19;
     I2C_Init();
     UART0_PutS("**** ARM Primer LPC-2148 I2C EEPROM Demo
***\n\n\r");
     UART0_PutS(">>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>\n\n\r");
     IO1CLR=0xFF << 16;
     while (1)
     {
     if ((IOPIN1 & SW3)==0)/*...To Load the Default Data to
the EEPROM ...*/
          {
               ii = 0;
               IOCLR1    =    0xFF << 16;
               IOSET1    =    1 << 16;
               UART0_PutC ('\n');
                    Wait (1000);
                    Ready     =    'F';
```

```
                }
                while (!(IOPIN1 & SW4));
           Wait (5000);Wait (5000);Wait (5000);Wait (5000);

           }
           if ((IOPIN1 & SW5) == 0)           /*To Erase the
Content in EEPROM*/
           {
                IOCLR1    =    0xFF << 16;
                IOSET1    =    1 << 18;
                ii = 2;
                Erase     =    1;
                while (ii < MAX)
                {
                Buff[ii]  =    0xFF;          //Load 0xFF to
EEPROM
                     ii++;
                }
                flag =    'W';
                I2C_Start (0x70);
                for (i=0;i<30;i++)  Wait(10000);
                I2C1CONCLR     =    1 << SI;
                while (!(IOPIN1 & SW5));
                Wait (5000);Wait (5000);Wait (5000);Wait
(5000);
           }
     }
}
void I2C_ISR(void) __irq
{
     {
     I2C1CONCLR     =    1 << STO;
     I2C1CONCLR     =    1 << STA; //Clear START Bit
     I2C1DAT        =    0xA0;//Slave Addr + W 1010 p2 p1 p0 w
     I2C1CONCLR     =    1 << SI;
     I2C1DAT        =    0xA0;//Slave Addr + R 1010 p2 p1 p0 r

     I2C_Start (0xA1);
     I2C1CONCLR     =    1 << SI;
                                  }
                                  index     =    0;
                                  break;
     I2C1DAT   =    0xA0;  //Slave Addr + W  1010 p2 p1 p0 w
     I2C1CONCLR     =    1 << STA;
     I2C1CONCLR     =    1 << STO;
     I2C1DAT        =    0xA1;//Slave Addr + R  1010 p2 p1 p0
     I2C1CONCLR     =    1 << SI;
     I2C1CONCLR     =    0x20;          //Clear START Bit
               if (Erase == 1)
                    {
     UART0_PutS ("\n\r Memory Erase Successfull..! \n");
```

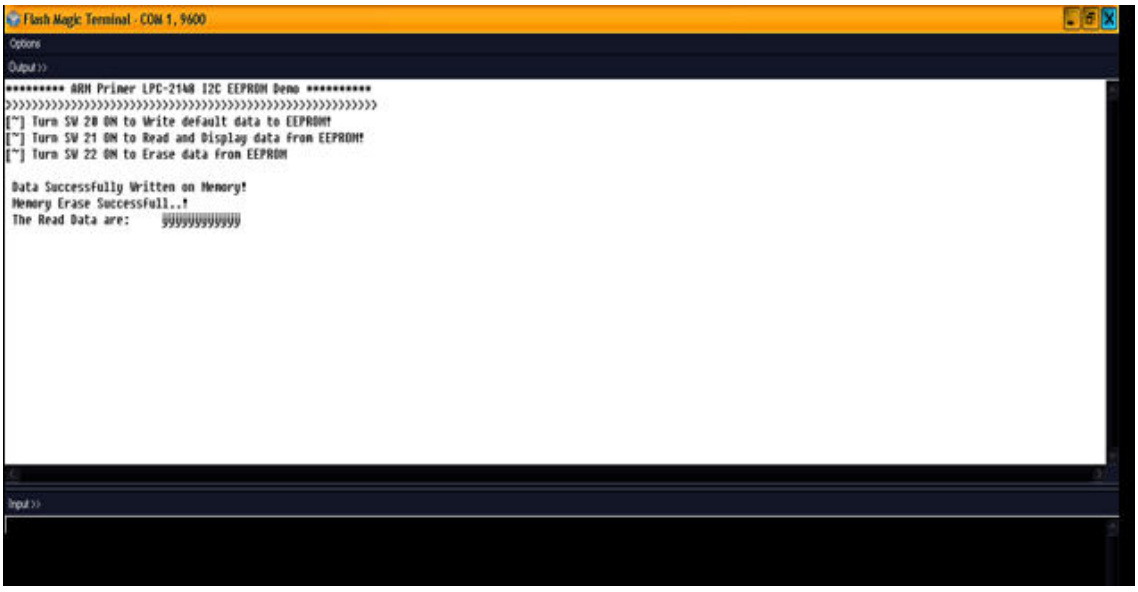
```
                Erase     =    0;
                                        }
                                        else
                                        {
UART0_PutS ("\n\r Data Successfully Written on Memory!\n");
                                        }
                                I2C1CONCLR     =    1 << STA;
                                I2C1CONCLR     =    1 << SI;
                                     }
break;
case (0x30): /*.Data byte in I2DAT has been transmitted; */
I2C1CONCLR     =    0x20; //Clear START Bit
if (index < MAX)
       break;
case (0x38): /* Arbitration lost in SLA+R/W orData bytes*/
                    I2C1CONSET     =    0x20;
                    break;
     case (0x40)    :    /*... SLA+R has been transmitted; ACK
has been received.*/
I2C1CONSET                  =  1 << AA;
I2C1CONCLR     =    1 << STA;

I2C1CONCLR     =    1 << SI;
                    break;
case (0x50):/*.Data byte has been received; ACK has been
            returned ....*/
if (index < MAX)
{
     Rec [index]    =    I2C1DAT;
     index++;
     }
     break;
     }
     }
```

INTERRUPT:

```
     #include <lpc214x.h>
     #include <stdio.h>
     int volatile EINT1  =    0;
     int volatile EINT2  =    0;
     void ExtInt_Serve1(void)__irq;
     void ExtInt_Serve2(void)__irq;
     /*--------------------------------<   INT2
     Initialization  >-----------------------------*/
     void ExtInt_Init2(void)
     {
          EXTMODE  |= 4;
               //Edge sensitive mode on EINT2
          EXTPOLAR = 0;
               //Falling Edge Sensitive
```

```
        PINSEL0  |= 0x80000000;  //Enable EINT2 on P0.15
}
void ExtInt_Init1(void)
{
        EXTMODE  |= 2;       //Edge sensitive mode on EINT1
        EXTPOLAR = 0;        //Falling Edge Sensitive
        PINSEL0  |= 0x20000000;
        VICIntEnable |= 1<<15;   //Enable EINT1
}
void Serial_Init(void)
{
        PINSEL0 |=      0X00000005;     //Enable Txd0 and Rxd0
        U0LCR     =     0x00000083;
        }
void main(void)
{
        Serial_Init();
        ExtInt_Init1();      //Initialize Interrupt 1
        DelayMs(500);
        }
void ExtInt_Serve2(void)__irq
{
        ++EINT2;
        EXTINT |= 4;
        VICVectAddr = 0;
}
```

OUTPUT:



RESULT:

Thus the circuit is designed and programmed to develop and verify Embedded C program for EPROM and interrupt Interfacing using ARM DEVELOPMENT KIT

Ex. No:7	MAILBOX
Date:	

AIM:

To develop and verify Embedded C program Mailbox for ARM DEVELOPMENT KIT microcontroller.

APPARATUS REQUIRED:

S.No	Apparatus	Range	Quantity
1	ARM Development Kit	LPC2148	1
2	Keil µVision3 IDE		1

PROCEDURE:

Step 1: Give a double click on µvision 4 icon on the desk top, it will generate a window as shown below

Step 2: To create new project go to project select new micro vision project.

Step 3: select a drive where you would like to create your project.

Step 4: Create a new folder and name it with your project name.

Step 5: Open that project folder and give a name of your project executable and save it.

Step 6: After saving it will show some window there you select your microcontroller company i.e NXP from Phillips

Step 7: Select your chip as ARM LPC 2148.

Step 8: After selecting chip click on OK then it will display some window asking to add STARTUP. Select YES.

Step 9: A target is created and startup is added to your project target and is shown below.

Step 10: To write your project code select a new from menu bar.

Step 11: It will display some text editor, to save that select SAVE option from menu bar.

Step 12: By giving a name with extension .C and save it

Step 13: Write the code of your project and save it.

Step 14: To add c file to target give a right click on Source Group, choose "ADD s to Group" option.

Step 15: It will displays some window there select the file and click on ADD option.

Step 16: It will be added to our target and it shows in the project window.

Step 17: Now give a right click on target in the project window and select "Options for Target"

Step 18: It will show some window, in that go to output option and choose Create Hex option by selecting that box

Step 19: In the same window go to Linker option and choose Use Memory Layout from Target Dialog by selecting the box, and click OK.

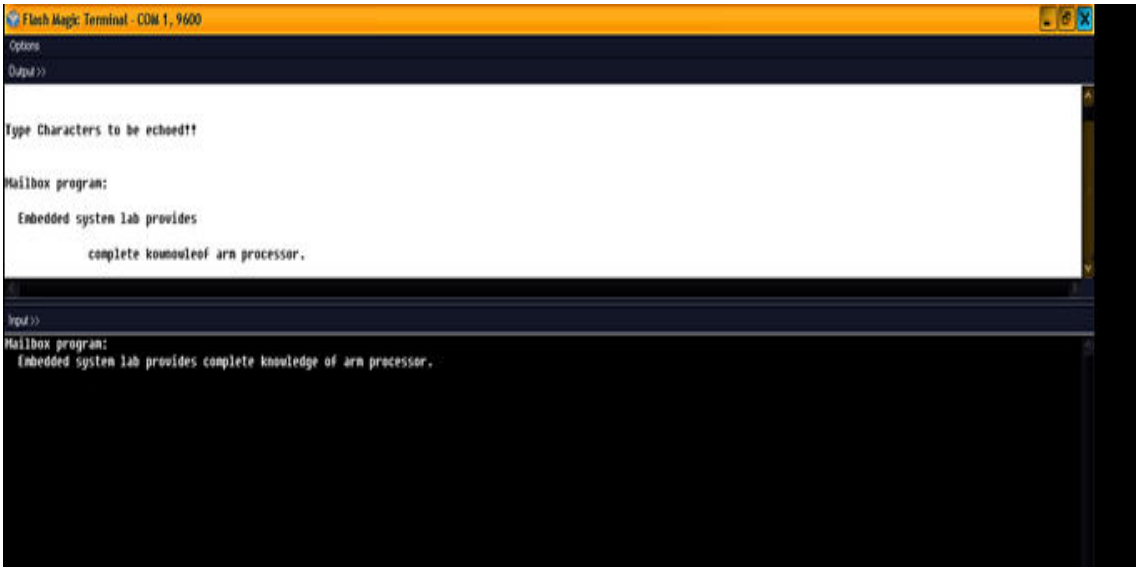
Step 20: Compile your project go to Project select Build Target option or press F7.

PROGRAM:

```
#define CR     0x0D
#include <LPC21xx.H>
void init_serial (void);
int putchar (int ch);
int getchar (void);
unsigned char test;
int main(void)
{
   VPBDIV = 0x02;           // Divide Pclk by two
   init_serial();
   while(1)
   {
      while (*Ptr)
      {
         putchar(*Ptr++);
      }
      putchar(getchar());  // Echo terminal
   }
void init_serial (void)
{
  PINSEL0    = 0x00000005; // Enable RxD0 and TxD0
  U0LCR    = 0x00000083;  //8 bits, no Parity, 1 Stop bit
}
int putchar (int ch)
 {
  if (ch == '\n')
  {
    while (!(U0LSR & 0x20));
    U0THR = CR;
  }
```

```
    while (!(U0LSR & 0x20));
    return (U0THR = ch);
}
int getchar (void)
{
    while (!(U0LSR & 0x01));
    return (U0RBR);
}
```

OUTPUT:



RESULT:

Thus the concept of Mailbox with ARM DEVELOPMENT KIT processor using embedded C program was done and verified the message successfully.

Ex. No:8 Date:	INTERRUPT CHARACTERISTICS OF ARM &FPGA

AIM:

To verify the Interrupt performance characteristics of ARM and FPGA by using embedded C program.

APPARATUS REQUIRED:

S.No	Apparatus	Range	Quantity
1	ARM Development Kit	LPC2148	1
2	Keil µVision3 IDE		1

PROCEDURE:

Step 1: Give a double click on µvision 4 icon on the desk top, it will generate a window as shown below

Step 2: To create new project go to project select new micro vision project.

Step 3: select a drive where you would like to create your project.

Step 4: Create a new folder and name it with your project name.

Step 5: Open that project folder and give a name of your project executable and save it.

Step 6: After saving it will show some window there you select your microcontroller company i.e NXP from Phillips

Step 7: Select your chip as ARM DEVELOPMENT KIT

Step 8: After selecting chip click on OK then it will display some window asking to add STARTUP. Select YES.

Step 9: A target is created and startup is added to your project target and is shown below.

Step 10: To write your project code select a new from menu bar.

Step 11: It will display some text editor, to save that select SAVE option from menu bar.

Step 12: By giving a name with extension .C for c s and save it

Step 13: Write the code of your project and save it.

Step 14: To add c file to target give a right click on Source Group, choose "ADD s to Group" option.

Step 15: It will display some window there select the file and click on ADD option.

Step 16: It will be added to our target and it shows in the project window.

Step 17: Now give a right click on target in the project window and select "Options for Target"

Step 18: Find a window and go to output option and choose Create Hex option by selecting that box.

Step 19: In the same window go to Linker option and choose Use Memory Layout from Target Dialog by selecting the box, and click OK.

Step 20: Compile the project and select Build Target option .

PROGRAM:

```
#include<LPC214x.h>// Define ARM DEVELOPMENT KIT  Header

#include <stdio.h>

int volatile EINT1    =   0;

int volatile EINT2   =   0;

void ExtInt_Serve1(void)__irq;

void ExtInt_Serve2(void)__irq;

void ExtInt_Serve1(void)__irq

{

   ++EINT1;

   EXTINT |= 2;

   VICVectAddr = 1;

}

void ExtInt_Serve2(void)__irq

{

   ++EINT2;
```

```
   EXTINT |= 4;
   VICVectAddr = 0;
}
void main(void)
{
   Serial_Init();
   ExtInt_Init1(); //Initialize Interrupt 1
   ExtInt_Init2(); //Initialize Interrupt 2
     putchar(0x0C);
        printf ("*External Interrupt Demo ***\n\n\r");
   DelayMs(100);
   printf (">>> Press Interrupt Keys SW2(INT1) and
          SW3(INT2) for Output... \n\n\n\r");
   DelayMs(100);
        while(1)
   {
           DelayMs(500);
      printf("INT1 Generated : %d  |  INT2 Generated
           : %d \r", EINT1, EINT2);
      DelayMs(500);
   }}
void ExtInt_Init2(void)
{
   EXTMODE  |= 4;          //Edge sensitive mode on EINT2
   EXTPOLAR = 0;           //Falling Edge Sensitive
```

```
 PINSEL0  |= 0x80000000; //Enable EINT2 on P0.15
   {
VICVectCntl1 = 0x20 | 16; // 16 is index of EINT2
   VICVectAddr1 = (unsigned long) ExtInt_Serve2;
   VICIntEnable |= 1<<16;    //Enable EINT2
}
}
void ExtInt_Init1(void)
{
   EXTMODE  |= 2;          //Edge sensitive mode on EINT1
   EXTPOLAR = 0;           //Falling Edge Sensitive
  {
 PINSEL0  |= 0x20000000;   //Enable EINT2 on P0.14
   VICVectCntl0 = 0x20 | 15; //15 is index of EINT1
   }
VICVectAddr0 = (unsigned long) ExtInt_Serve1;
   VICIntEnable |= 1<<15;    //Enable EINT1
}
{
void Serial_Init(void)
{   PINSEL0 |=   0X00000005; //Enable Txd0 and Rxd0
   U0LCR    =   0x00000083; //8-bit data,1-stop bit
   }
U0DLL   =   0x00000061; //XTAL = 12MHz
```

```
        U0LCR   =   0x00000003; //DLAB = 0;
     }
     void DelayMs(unsigned int count)
    {         unsigned int i,j;
            for(i=0;i<count;i++)
        {
           for(j=0;j<1000;j++);
        }}
```

OUTPUT:

RESULT:

Thus the Interrupt performance characteristics of ARM processor and FPGA has been done using embedded C program.

Ex. No:9 Date:	FLASHING OF LEDS

AIM:

To verify the flashing of LEDS in ARM DEVELOPMENT KIT microcontroller board using embedded C program.

APPARATUS REQUIRED:

S.No	Apparatus	Range	Quantity
1	ARM DEVELOPMENT KIT	LPC2148	1
2	Keil µVision3 IDE		1

PROCEDURE:

Step 1: Open the µvision 4 icon on the desk top, it will generate a window as shown below

Step 2: To create new project go to project select new micro vision project.

Step 3: select a drive where you would like to create your project.

Step 4: Create a new folder and name it with your project name.

Step 5: Open that project folder and give a name of your project executable and save it.

Step 6: After saving it will show some window there you select your microcontroller company i.e NXP from Phillips

Step 7: Select your chip as ARM DEVELOPMENT KIT

Step 8: After selecting chip click on OK then it will display some window asking to add STARTUP. Select YES.

Step 9: A target is created and startup is added to your project target and is shown below.

Step 10: To write your project code select a new from menu bar.

Step 11: It will display some text editor, to save that select SAVE option from menu bar.

Step 12: By giving a name l with extension .C and save it

Step 13: Write the code of your project and save it.

Step 14: To add our c to target give a right click on Source Group, choose "ADD s to Group" option.

Step 15: It will display some window there select the you have to add and click on ADD option.

Step 16: It will be added to our target and it shows in the project window.

Step 17: Right click on target in the project window and select "Options for Target"

Step 18: It will show some window, in that go to output option and choose Create Hex option by selecting that box

Step 19: In the same window go to Linker option and choose Memory Layout from Target Dialog by selecting the box, and click OK.

Step 20: Compile your project go to Project select Build Target option or press F7.

Step 21: observe the flashing of LED's in the ARM board.

PROGRAM:

```
#include <lpc214x.h>
unsigned int delay;
int main(void)
{
     IO0DIR = (1<<10);  // Configure P0.10 as Output
     while(1)
     {
          IO0CLR = (1<<10);// CLEAR(0)P0.10toturn LED ON
          for(delay=0; delay<500000; delay++); // delay
          IO0SET = (1<<10);   //  SET  (1)  P0.10  to  turn
LEDs OFF
          for(delay=0; delay<500000; delay++); // delay
     }
}
```

OUTPUT:

IO0SET =ox00 (OFF)
IO0CLR =oxFF (ON)
Delay =2

RESULT:

Thus the Flashing of LEDS using ARM DEVELOPMENT KIT board was observed by using embedded C program successfully.

Ex. No:10	INTERFACING OF STEPPER MOTOR AND TEMPERATURE SENSOR
Date:	

AIM:

To interface and verify the stepper motor and temperature sensor with ARM DEVELOPMENT KIT microcontroller using embedded C program.

APPARATUS REQUIRED:

S.No	Apparatus	Range	Quantity
1	ARM DEVELOPMENT KIT	LPC2148	1
2	Keil µVision3 IDE		1

PROCEDURE:

Step 1: Open a µvision 4 icon on the desk top, it will generate a window as shown below.

Step 2: To create new project go to project select new micro vision project.

Step 3: select a drive where you would like to create your project.

Step 4: Create a new folder and name it with your project name.

Step 5: Open that project folder and give a name of your project executable and save it.

Step 6: After saving it will show some window there you select your microcontroller company i.e NXP from Phillips

Step 7: Select your chip as ARM DEVELOPMENT KIT

Step 8: After selecting chip click on OK then it will display some window asking to add STARTUP. Select YES.

Step 9: A target is created and startup is added to your project target and is shown below.

Step 10: To write your project code select a new from menu bar.

Step 11: It will display some text editor, to save that select SAVE option from menu bar.

Step 12: By giving a name with extension .C for c s and save it

Step 13: Write the code of your project and save it.

Step 14: To add c file to the target give a right click on Source Group, choose "ADD s to Group" option.

Step 15: It will display some window there select file and click on ADD option.

Step 16: It will be added to our target and it shows in the project window.

Step 17: Now give a right click on target in the project window and select "Options for Target"

Step 18: It will show some window, in that go to output option and choose Create Hex option by selecting that box

Step 19: In the same window go to Linker option and choose Use Memory Layout from Target Dialog by selecting the box, and click OK.

Step 20: Compile your project go to Project select Build Target option or press F7.

PROGRAM:

STEPPERMOTOR:

```
#include <lpc21xx.h>
#include <stdio.h>
#define SW1    24              //SW1 (P1.24)
#define SW2 25           //SW2 (P1.25)
#define SW3 26                  //SW3 (P1.26)
#define COIL_A 16 // Change here if you want to change the
Stepper Motor Port!
void motor_cw(void);
void motor_ccw(void);
void delay(int);
unsigned char STEP[]={0x09, 0x0C, 0x06,  0x03};
//<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<   Delay Routine
void delay(int n)
{
     int i,j;
     for(i=0;i<n;i++)
     {
          for(j=0;j<0x3FF0;j++)
          {;}
     }}
//<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<   counter-clockwise
rotate function
void motor_ccw(void)
{
     unsigned int i=0;
     while (STEP[i] != '\0')
     {
          IOSET1 = STEP[i] << COIL_A;         delay(5);
```

```
          IOCLR1 = STEP[i] << COIL_A;        delay(5);
          i++;
     }}
//<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<   Clock-wise Rotate
Funciton
void motor_cw(void)
{
     int i = 3;
     while (i >= 0)
     {
          IOSET1 = STEP[i] << COIL_A;        delay(5);
          IOCLR1 = STEP[i] << COIL_A;        delay(5); i--;
     }}
void main(void)
{
     unsigned char i = 0;
     PINSEL2   &=   0xFFFFFFF3;    //Configure P1.16 - P1.31
as GPIO
     IODIR1 = 0x000F0000;          //Configure P1.16 - P1.19
as Output Pins
          while(1)                              // Loop
forever.............
     {
          if (!(IOPIN1 & (1<<SW1))) //Check Switch SW1 ON/OFF
          {
               IOCLR0    =    0xFF << COIL_A;
               motor_cw();/* Rotate Stepper Motor clockwise */
          }
          else if (!(IOPIN1 & (1<<SW2))) //Check Switch SW2
ON/OFF
          {
               IOCLR0    =    0xFF << COIL_A;
               motor_ccw();/* Rotate Stepper Motor counter
clockwise */
          }
          else if (!(IOPIN1 & (1<<SW3)))          //Check
Switch SW3 ON/OFF
          {
               IOCLR0    =    0xFF << COIL_A;
               while (i < 12)
               {
               motor_cw ();/* Rotate Motor Clockwise for a
particular angle */
                    i++;
               }    }
          else
               i = 0;     }     }
```

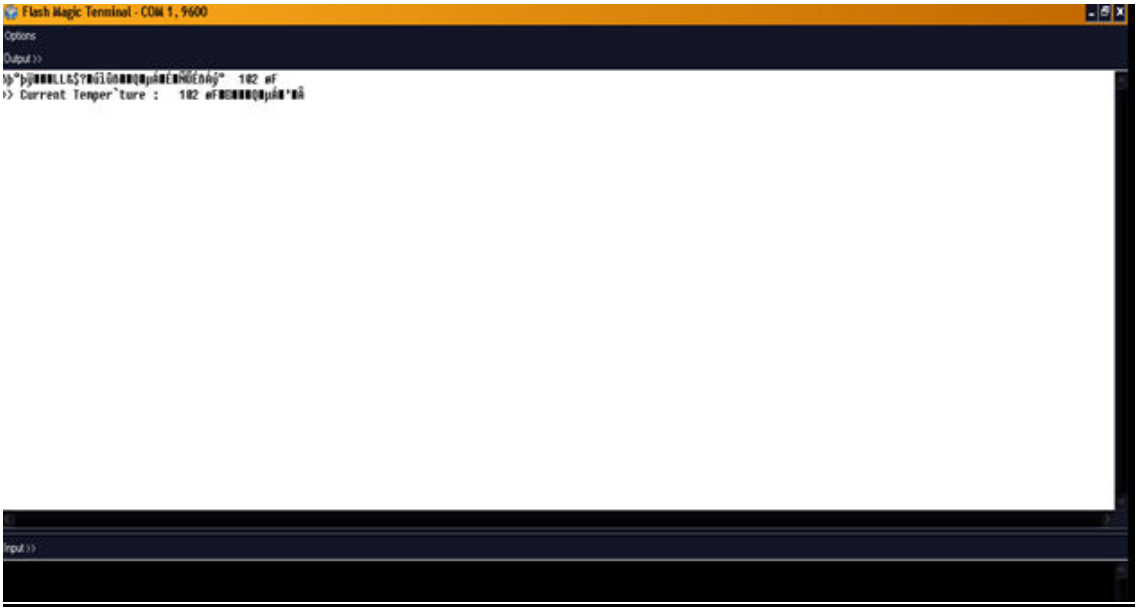
TEMPERATURESENSOR:

```
#include <LPC214x.h>
#include <stdio.h>
#define DONE   0x80000000
#define START  0x01000000
#define PRESET 0x00230600
void Delay ()
{    unsigned int i,j;
     for (i=0;i<50;i++)
          for (j=0;j<500;j++);}
void Welcome ()
{    printf ("-.-.-..-.-.-.-.-.-.\n\r");
     printf (" Developed By :  R&D Wing \n\r");
     printf (" © 2009 Pantech Solutions Pvt Ltd \n\r");
     printf ("--------------------\n\r");
     printf ("*** Temperature Sensor Interfacing with
Tyro Kit ***\n\r");
     printf ("-----------------------------\n\r");
     printf (">> Put Jumper J in 'E' Mode to Enable Temp
Sensor Block \n\r");
     printf (">> Connect UART1 to COM Port @ 9600 Baud
Rate\n\n\r");
     printf("******\n\r");
     printf ("******* Result **********\n\r");
     printf("***************\n\n\r");
}
void Serial_Init ()
{
     PINSEL0|= 0x00050000; //Configure TxD1 and RxD1@
P0.8 & P0.9
     U1LCR     =    0x83;
     U1DLL     =    195;
     U1LCR     =    0x03;
}

void main ()
{    unsigned long Val;
     VPBDIV    =    0x02;     //pclk @ 30MHz
     Serial_Init ();
     PINSEL1=0x01 << 24; //P0.28 configure as ADC0.1
     Welcome ();
```

```
        AD0CR     =    PRESET | 0x02;       //ADC Config:
        10bit/9Clock |     BURST = 1 | CLKDIV = 0x06
        AD0CR     |=   START;                    //Start
    Conversion NOW
              while (1)
        {          do
              }while ((Val & DONE) == 0);         //Check if
    Conversion is DONE
              printf (">> Current Temperature :  %4d ", Val);
              printf ("\xF8\F \r");
        }
    }
```

OUTPUT:

TEMPERATURE SENSOR:



RESULT:

Thus, the interfacing of stepper motor and temperature sensor with ARM DEVELOPMENT KIT processor has been done and executed the temperature value.

Ex. No:11 Date:	IMPLEMENTING ZIG BEE PROTOCOL WITH ARM

AIM:

To implement the ZigBee protocol with ARM DEVELOPMENT KIT microcontroller and perform the communication process.

APPARATUS REQUIRED:

S.No	Apparatus	Range	Quantity
1	ARM DEVELOPMENT KIT	LPC2148	1
2	Keil µVision3 IDE		1

PROCEDURE:

Step 1: Open the µvision 4 icon on the desk top, it will generate a window as shown below

Step 2: To create new project go to project select new micro vision project.

Step 3: select a drive where you would like to create your project.

Step 4: Create a new folder and name it with your project name.

Step 5: Open that project folder and give a name of your project executable and save it.

Step 6: After saving it will show some window there you select your microcontroller company i.e NXP from Phillips

Step 7: Select your chip as ARM DEVELOPMENT KIT .

Step 8: After selecting chip, click on OK then it will display some window asking to add STARTUP. Select YES.

Step 9: A target is created and startup is added to your project target and is shown below.

Step 10: To write your project code select a new from menu bar.

Step 11: It will display some text editor, to save that select SAVE option from menu bar.

Step 12: By giving a name with extension .C for c s and save it

Step 13: Write the code of your project and save it.

Step 14: To add c file to target give a right click on Source Group, choose "ADD s to Group" option.

Step 15: It will displays some window there select the you have to add and click on ADD option.

Step 16: It will be added to our target and it shows in the project window.

Step 17: Now give a right click on target in the project window and select "Options for Target"

Step 18: It will show some window, in that go to output option and choose Create Hex option by selecting that box

Step 19: In the same window go to Linker option and choose Use Memory Layout from Target Dialog by selecting the box, and click OK.

Step 20: Compile your project go to Project select Build Target option or press F7.

PROGRAM:

TX code:

```
#include "header.h"
#include "lcd.h"
#include "pll.h"
#include "uart.h"
int main()
{
     PINSEL1=0;
     PINSEL2=0;
     IO0DIR|=0xff<<16 | 0xf<<25;
     IO1DIR&=(~(1<<16)) & (~(1<<17)) & (~(1<<18));
     lcd_init();
     uart_init();
     lcdstring("Rain Detector");
     lcdcmd(0xc0);
     lcdstring("Using Zigbee");
     delay(50);
     lcdcmd(0x01);
     lcdstring("Project By:");
     lcdcmd(0xc0);
     lcdstring("FIRMCODES");
     delay(50);
     lcdcmd(0x01);
     while(1)
     {if(!(IO1PIN & (1<<16))) /Rain Detector Sensor
          {
               txdata('r');
               delay(100);
               IO0SET=1<<25;                    //Rain Indicator
```

```
          }
          if(!(IO1PIN & (1<<17)))                    //Rain
Detector Sensor
          {
               txdata('g');
               delay(100);
               IO0SET=1<<26;                    //Gas Indicator
          }
          else
          {
               txdata('n');
               delay(100);
               IO0CLR=1<<25;                    // Rain
Indicator:
          }}}}
```

RX code:

```
#include "header.h"
#include "lcd.h"
#include "uart.h"
#include "pll.h"
int main()
{
     PINSEL1=0;
     PINSEL2=0;
     IO0DIR|=0xff<<16 | 0xf<<25;
     IO1DIR|=(~(1<<16));
     lcd_init();
     uart_init();
     lcdstring("Rain Detector");
     lcdcmd(0xc0);
     lcdstring("Using Zigbee");
     delay(50);
     lcdcmd(0x01);
     while(1)
          {
           rxdata();
          }
    }
```

RESULT:

Thus the implementation of ZigBee protocol with ARM DEVELOPMENT KIT processor board using embedded C program has been done .