<u>ECA14 – Embedded System</u>

List of Experiments

S. No	Name of the Experiment	Remarks
1	Study Of Proteus micro Vision	
2	Blinking Of Led Using 8051 Microcontroller Using Proteus	
3	Generation Of Square Wave Using Proteus	
4	Fade In Fade Out Of Led Using 8051 Using Proteus	
5	Stepper Motor Using 8051 Using Proteus	
6	Interfacing Of Relay Using 8051 Using Proteus	
7	Led Toggle Using 8051 Using Proteus	
8	7 Segment Display Using 8051 Using Proteus	
9	Led Chaser Using 8051 Using Proteus	
10	Study of ARM Processor	
11	Write and execute C program to blink LEDs using software delay routine in LPC2148 kit	
12	Write and execute C program to read the switch and display in the LEDs using LPC2148 kit	
13	Write and execute C program to display a number in seven segment LED in LPC2148 kit	
14	Write and execute C program for serial transmission and reception using on-chip UART in LPC2148 kit.	
15	Write and execute C program for accessing an internal ADC and display the binary output in LEDS in LPC2148 kit.	

Experiment no -01

Study of Proteus and Keil Micro Vision

Aim: To study the working procedures of Proteus and Keil Micro vision softwares.

Keil Micro Vision is a free software which solves many of the main points for an embedded program developer. This software is an integrated development environment (IDE), which integrated a text editor to write programs, a compiler and it will convert your source code to hex files too. µVision4 introduces a flexible window management system, enabling us to drag and drop individual windows anywhere on the visual surface including support for Multiple Monitors.

KEIL PROCEDURE:

1. Open the software, Click on project and open new version project.
2. Create a new project file
3. Enter AT89C51
4. Click NO
5. Click [Ctrl +N] and Type the code
6. Open project and click Build target
7. Open Build target and open source file and ADD, CLOSE
8. Click build target
9. Next debug start and stop
10. Open peripherals and select port 2
11. Now run the program in Debug
12. Open project and click optional properties and in that give output as hex file.
13. Create hex file.

PROTEUS PROCEDURE:

- Open proteus by clicking run as administrator.
- Open new project and enter the file name.
- Click next, next, next and finish.
- Click P symbol and search keyword and place the required components
- Now connect the components as required
- Give input to AT89C51 as HEX file.
- Start the simulation process

Result: Thus the Proteus and Keil Micro vision softwares were studied.

Experiment 02

BLINKING OF LED USING 8051 MICROCONTROLLER USING PROTEUS

AIM:

To Write an assembly language program to LED blink using 8051

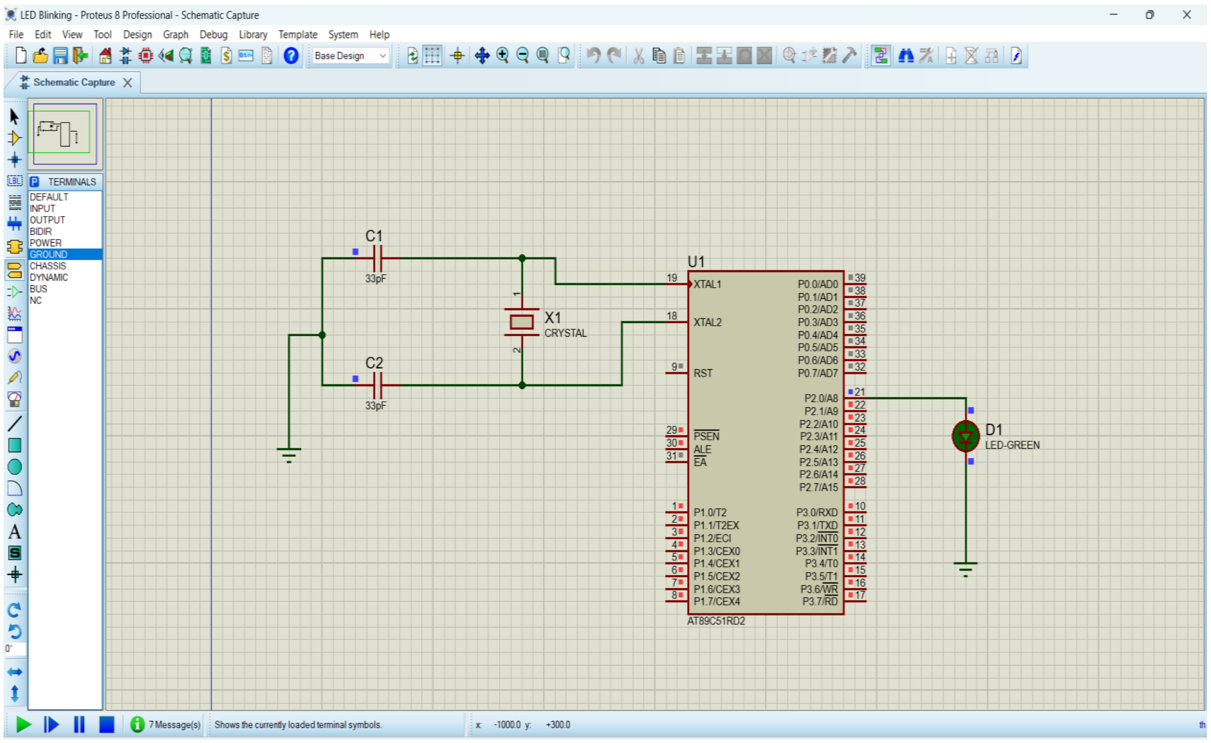
SOFTWARES REQUIRED:

- Proteus software

PROGRAM

```
            ORG 0000H
       UP: SETB P2.0
            ACALL DELAY
            CLR P2.0
            ACALL DELAY
            SJMP UP
DELAY: MOV R4,#35
       H1:MOV R3,#255
       H2:DJNZ R3,H2
           DJNZ R4,H1
           RET
           END
```

CIRCUIT DIAGRAM:


LED Blinking - Proteus 8 Professional - Schematic Capture

7 Message(s) Shows the currently loaded terminal symbols. x: -1000.0 y: +300.0

RESULT

Thus the program has been successfully verified and executed.

Experiment 3

GENERATION OF SQUARE WAVE USING PROTEUS

AIM:

Write an assembly language program to Generate square wave using 8051.

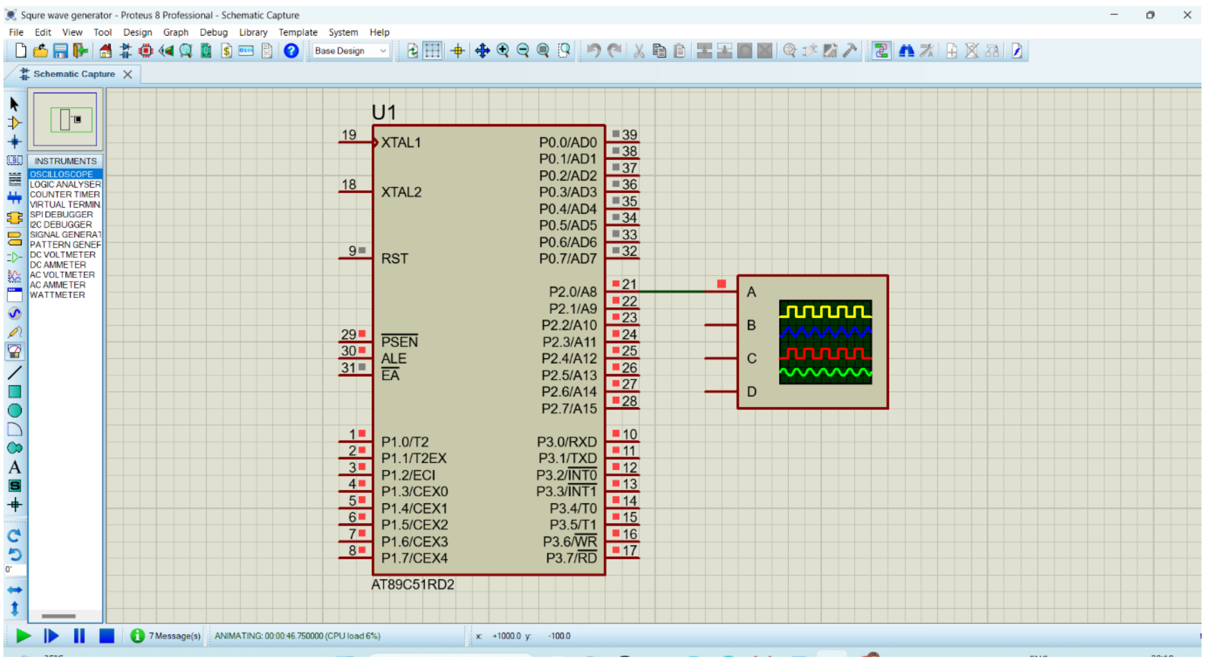
SOFTWARE REQUIRED:

- Proteus 8 software.

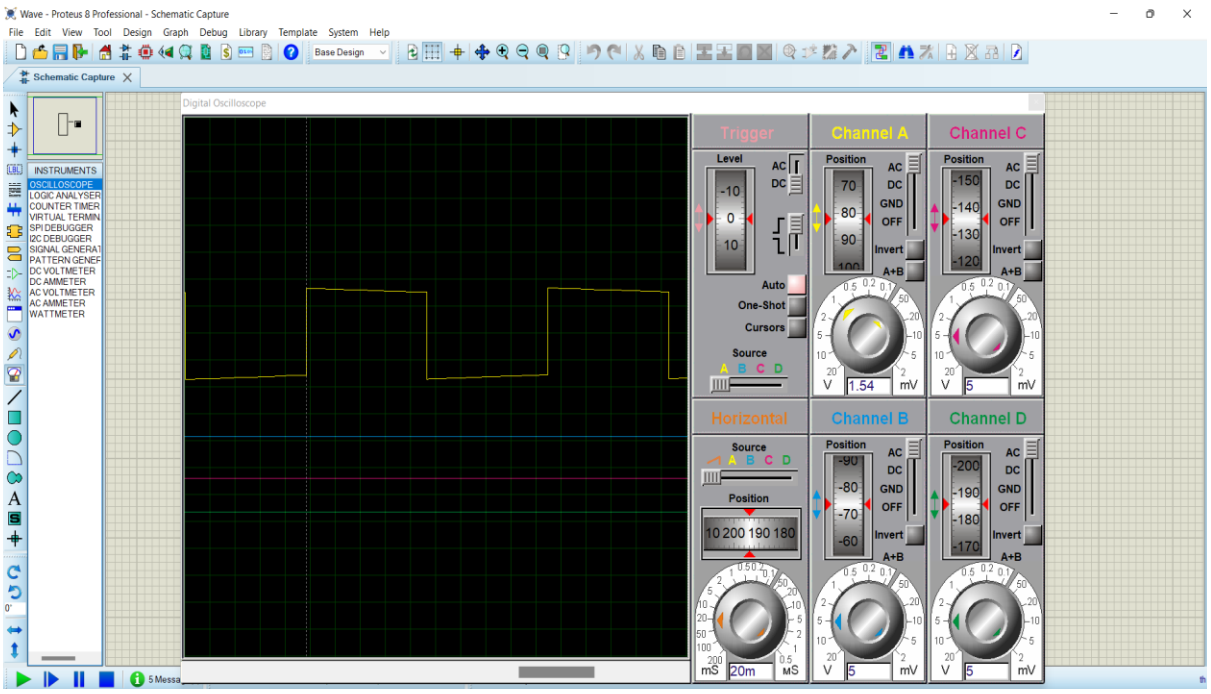
PROGRAM

```
          ORG 0000H
      UP: SETB P2.0
          ACALL DELAY
          CLR P2.0
          ACALL DELAY
          SJMP UP
DELAY: MOV R4,#35
      H1:MOV R3,#255
      H2:DJNZ R3,H2
         DJNZ R4,H1
         RET
         END
```

CIRCUIT DIAGRAM:



OUTPUT:



RESULT:

Thus the program has been successfully verified and executed.

Experiment 4

FADE IN FADE OUT OF LED USING 8051 USING PROTEUS

AIM:

Write an assembly language program for Fade in Fade out of LED Using 8051 using Keil and Proteus

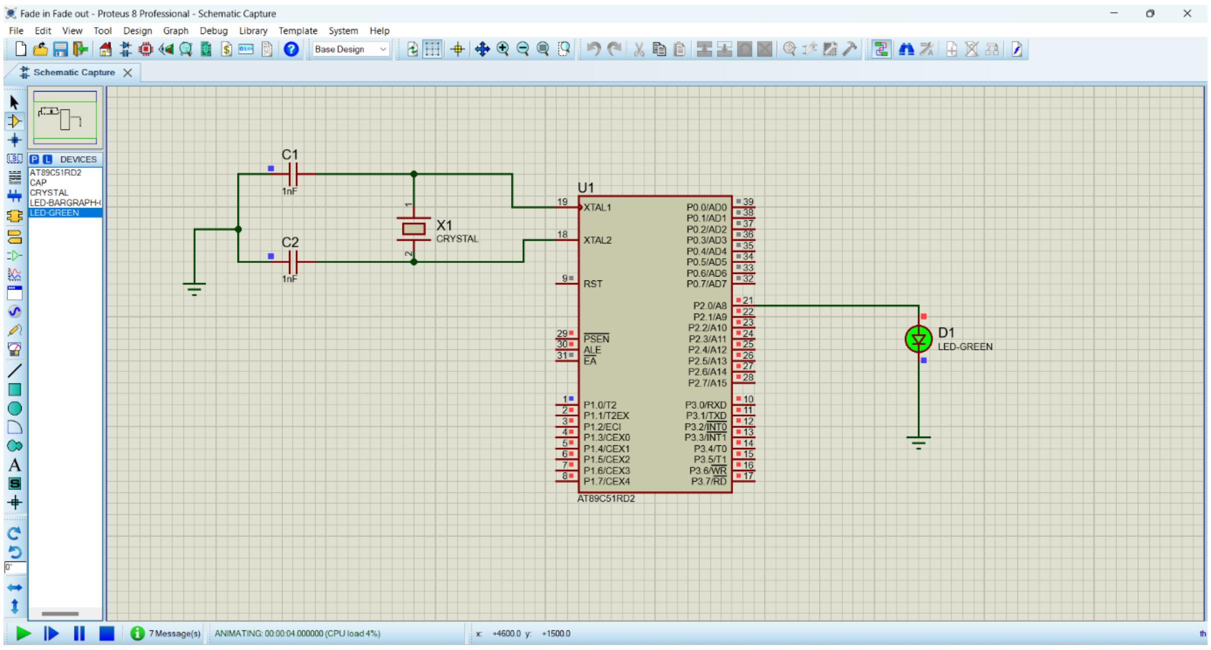
SOFTWARE REQUIRED:

- Proteus 8 software.

PROGRAM:

```
#include <REGX52.h>
delay(unsigned int y)
{
   unsigned int i,j;
for(i=0;i<y;i++)
{
   for(j=0;j<1275;j++){}
}
}
main()
{
   while(1)
{
      delay(100);
      P1_0 = 0;
      delay(100);
      P1_0 = 1;
   }
}
```

CIRCUIT DIAGRAM:


Fade in Fade out - Proteus 8 Professional - Schematic Capture
File Edit View Tool Design Graph Debug Library Template System Help
Schematic Capture
DEVICES
AT89C51RD2
CAP
CRYSTAL
LED-BARGRAPH
LED-GREEN
C1
1nF
C2
1nF
X1
CRYSTAL
U1
XTAL1
XTAL2
RST
PSEN
ALE
EA
AT89C51RD2
D1
LED-GREEN
7 Message(s)
ANIMATING: 00:00:04.000000 (CPU load 4%)

RESULT:

Thus the program has been successfully verified and executed.

Experiment 5

STEPPER MOTOR USING 8051 USING PROTEUS

AIM:

Write an assembly language program for Stepper Motor Using 8051 using Keil and Proteus

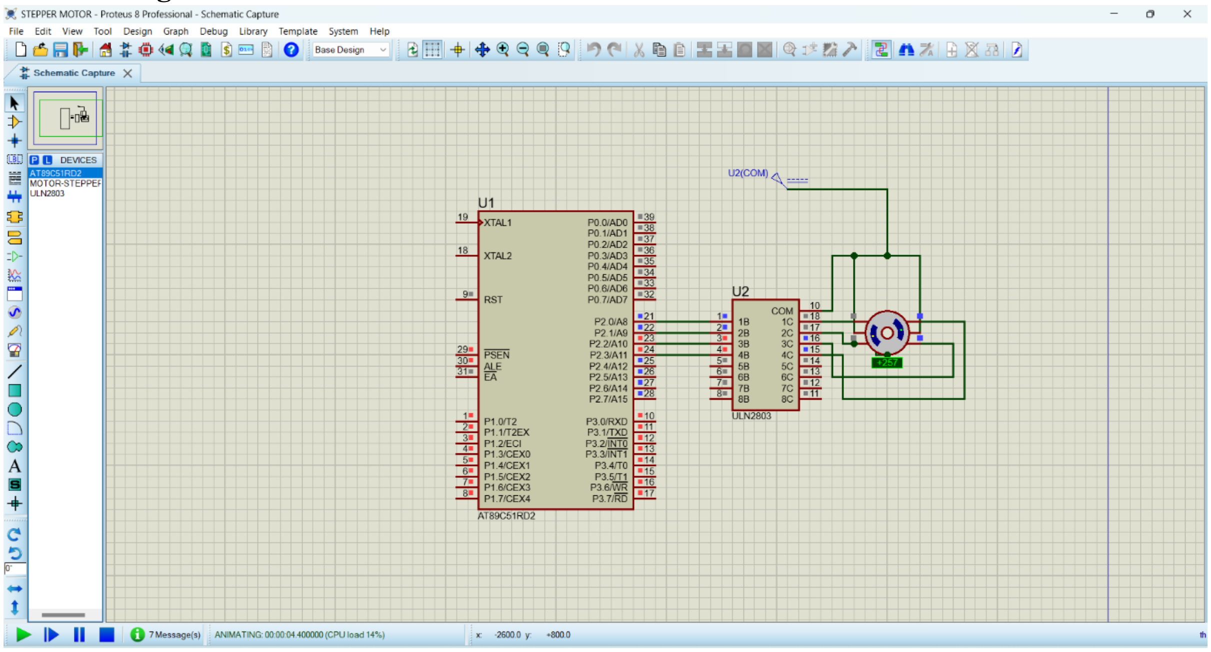
SOFTWARE REQUIRED:

- Proteus 8 software.

PROGRAM:

```
ORG 0000H
UP: MOV P2,#09H
ACALL DELAY
MOV P2,#0CH
ACALL DELAY
MOV P2,#06H
ACALL DELAY
MOV P2,#03H
ACALL DELAY
SJMP UP
DELAY:MOV R4,#18
H1:MOV R3,#255
H2:DJNZ R3,H2
DJNZ R4,H1
RET
END
```

Circuit Diagram:



RESULT:

Thus the program has been successfully verified and executed.

Experiment 6

INTERFACING OF RELAY USING 8051 USING PROTEUS

AIM:

Write an assembly language program for Interfacing of Relay Using 8051 using Keil and Proteus

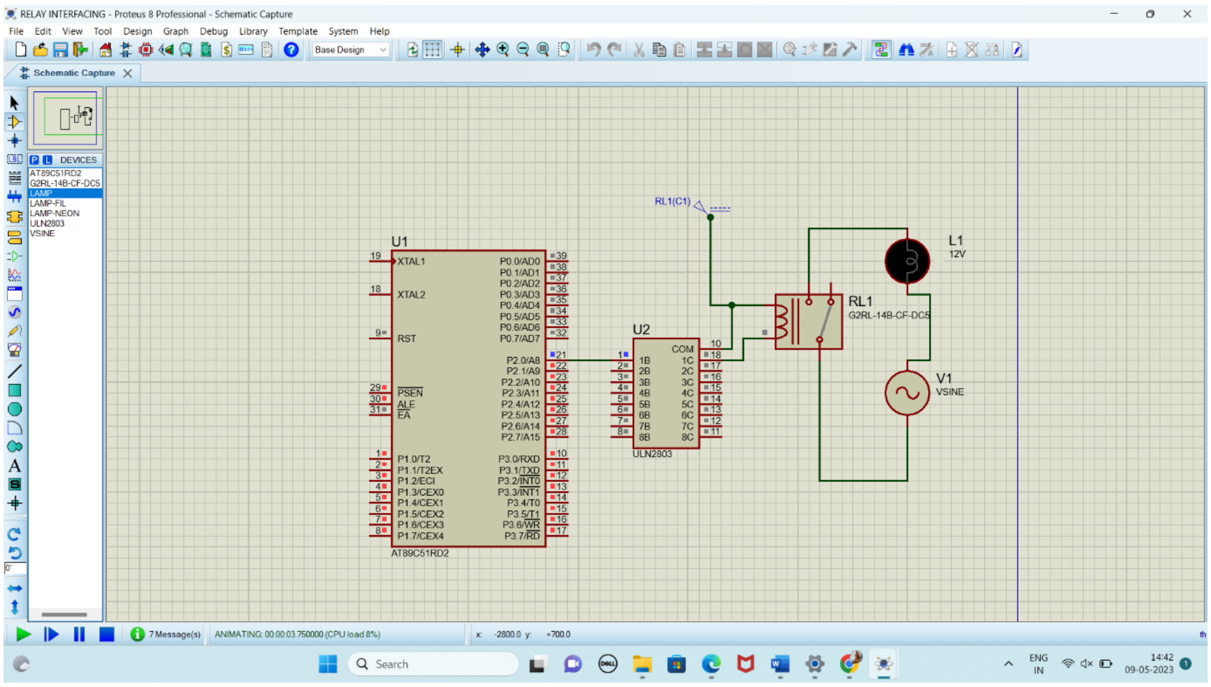
SOFTWARE REQUIRED:

- Proteus 8 software.

PROGRAM:

```
ORG 0000H
UP:SETB P2.0
ACALL DELAY
CLR P2.0
ACALL DELAY
SJMP UP
DELAY:MOV R4,#18
H1:MOV R3,#255
H2:DJNZ R3,H2
DJNZ R4,H1
RET
END
```

CIRCUIT DIAGRAM:



RESULT:

Thus the program has been successfully verified and executed.

Experiment 7

LED TOGGLE USING 8051 USING PROTEUS

AIM:

Write an assembly language program for LED Toggle Using 8051 using Keil and Proteus

SOFTWARE REQUIRED:

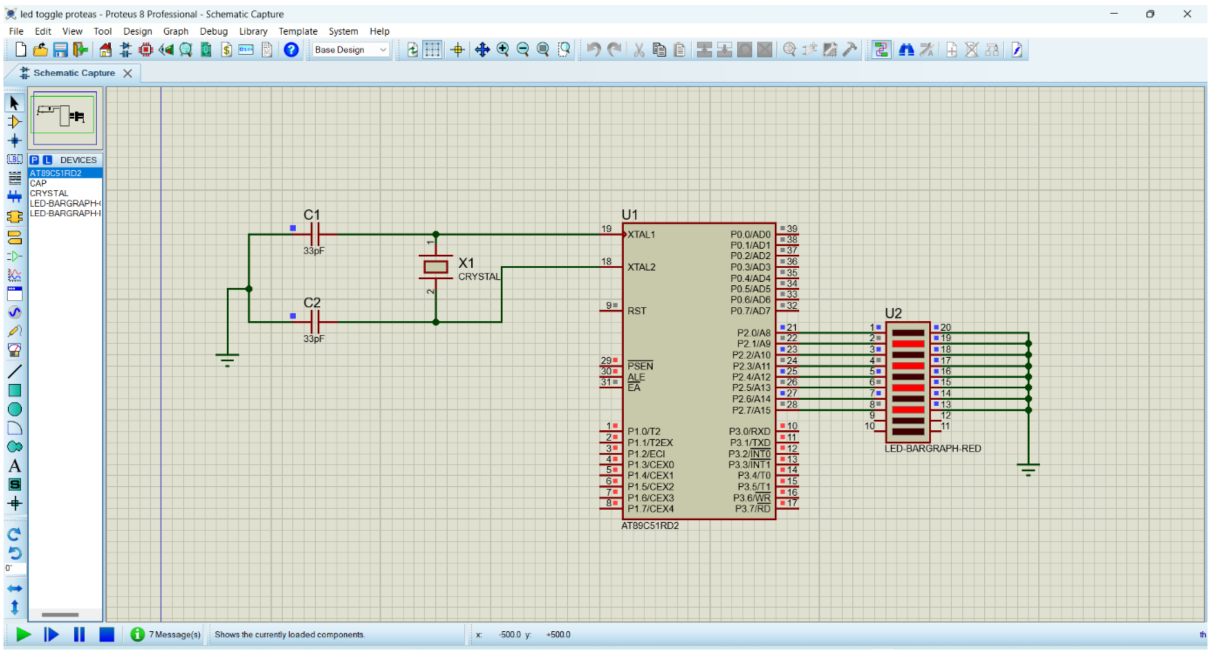
- Proteus 8 software.

PROGRAM:

```
ORG 0000H
UP: MOV P2,#55H
ACALL DELAY
MOV P2,#0AAH
ACALL DELAY
SJMP UP

DELAY:MOV R4,#10
H1:MOV R3,#255
H2:DJNZ R3,H2
DJNZ R4,H1
RET
END
```

CIRCUIT DIAGRAM:



RESULT:

Thus the program has been successfully verified and executed.

Experiment 8

7 SEGMENT DISPLAY USING 8051 USING PROTEUS

AIM:

Write an assembly language program for 7 Segment Display Using 8051 using Keil and Proteus

SOFTWARE REQUIRED:

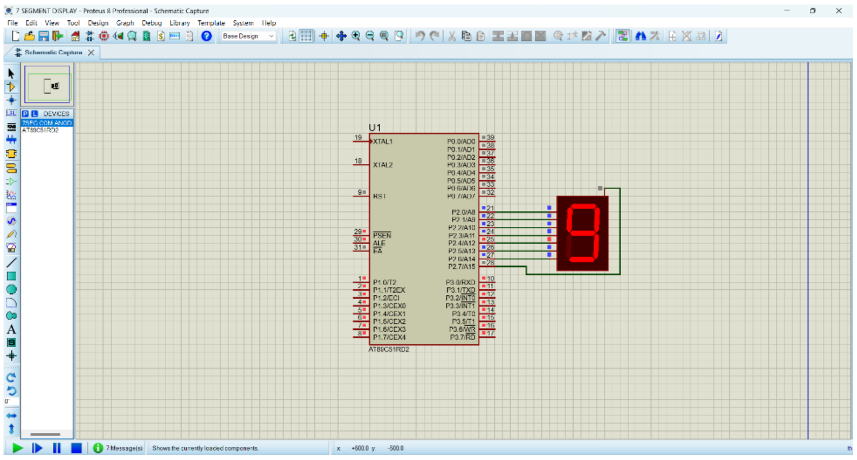
- Proteus 8 software.

PROGRAM:

```
ORG 000H
UP:MOV P2,#0C0H
ACALL DELAY
MOV P2,#0F9H
ACALL DELAY
MOV P2,#0A4H
ACALL DELAY
MOV P2,#0B0H
ACALL DELAY
MOV P2,#99H
ACALL DELAY
MOV P2,#92H
ACALL DELAY
MOV P2,#82H
ACALL DELAY
MOV P2,#0F8H
ACALL DELAY
MOV P2, #80H
ACALL DELAY
MOV P2,#90H
ACALL DELAY

DELAY: MOV R5,#10
H1:MOV R4,#180
H2:MOV R3,#255
H3:DJNZ R3,H3
DJNZ R4,H2
DJNZ R5,H1
RET
END
```

CIRCUIT DIAGRAM:



RESULT:

Thus the program has been successfully verified and executed.

Experiment 9

LED CHASER USING 8051 USING PROTEUS

AIM:

Write an assembly language program for LED Chaser Using 8051 using Keil and Proteus

SOFTWARE REQUIRED:

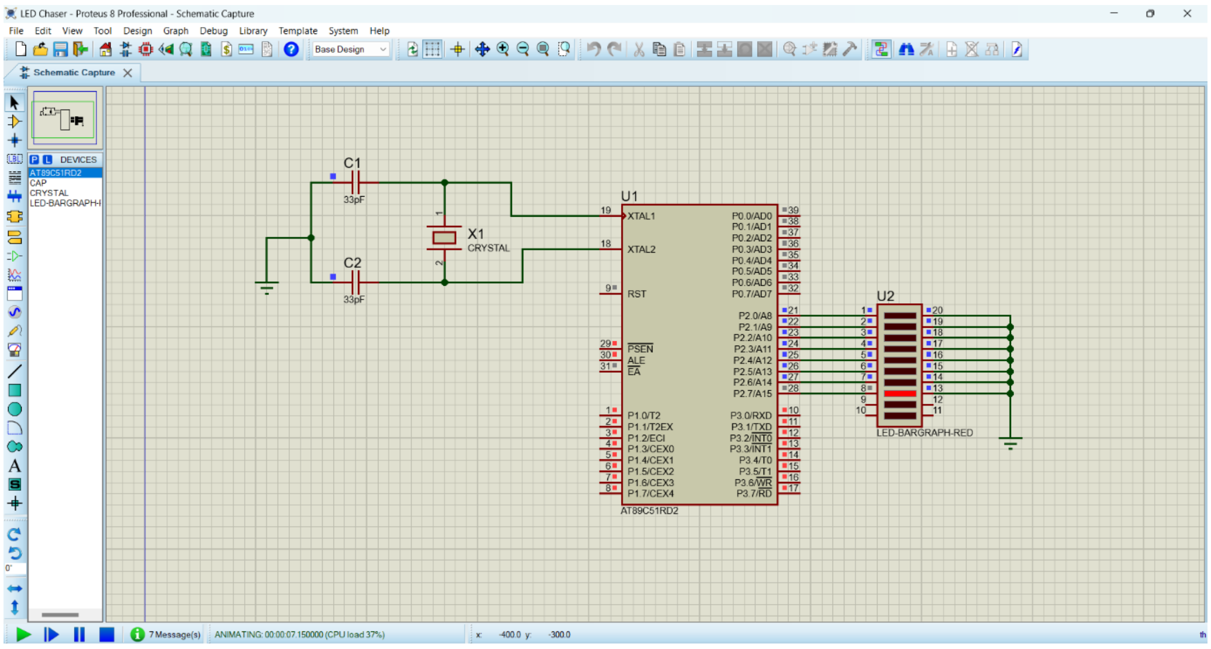
- Proteus 8 software.

PROGRAM:

```
ORG 0000H
UP: MOV P2,#01H
ACALL DELAY
MOV P2,#02H
ACALL DELAY
MOV P2,#04H
ACALL DELAY
MOV P2,#08H
ACALL DELAY
MOV P2,#10H
ACALL DELAY
MOV P2,#20H
ACALL DELAY
MOV P2,#40H
ACALL DELAY
MOV P2,#80H
ACALL DELAY
SJMP UP

DELAY: MOV R4,#255
H1: DJNZ R4,H1
RET
END
```

CIRCUIT DIAGRAM:



RESULT:

Thus the program has been successfully verified and executed.

Exp. **STUDY OF ARM PROCESSOR**

Date:

Aim:

To study the architecture and working of an ARM processor.

Component Required:

- ARM LPC2148

Theory:

ARM is a family of instruction set architectures for computer processors based on a reduced instruction set computing (RISC) architecture developed by British company ARM Holdings. A RISC-based computer design approach means ARM processors require significantly fewer transistors than typical processors in average computers. This approach reduces costs, heat and power use. These are desirable traits for light, portable, battery-powered devices—including smartphones, laptops, tablet and notepad computers), and other embedded systems. A simpler design facilitates more efficient multi-core CPUs and higher core counts at lower cost, providing higher processing power and improved energy efficiency for servers and supercomputers.

Features of LPC214x Series Controllers:

- 8 to 40 kB of on-chip static RAM and 32 to 512 kB of on-chip flash program 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 or full chip erase in 400 ms and programming of 256 bytes in 1ms.
- Embedded ICE RT and Embedded Trace interfaces offer real-time debugging with the on-chip Real Monitor software and high speed tracing of instruction execution.
- USB 2.0 Full Speed compliant Device Controller with 2 kB of endpoint RAM. In addition, the LPC2146/8 provides 8 kB of on-chip RAM accessible to USB by DMA.
- One or two (LPC2141/2 vs. LPC2144/6/8) 10-bit A/D converters provide a total of 6/14analog inputs, with conversion times as low as 2.44 us per channel.
- Single 10-bit D/A converter provides variable analog output.
- 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 with independent power and dedicated 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 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 nine edge or level sensitive external interrupt pins available.
- On-chip integrated oscillator operates with an external crystal in range from 1 MHz to30 MHz and with an external oscillator up to 50 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, USB, Brown-Out Detect (BOD) or Real-Time Clock (RTC).
- Single power supply chip with Power-On Reset (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.

LPC2148 needs the following hardware to work properly:

- Power Supply
- Crystal Oscillator
- Reset Circuit
- RTC crystal oscillator
- UART

Power Supply

LPC2148 works on 3.3 V power supply. LM 117 can be used for generating 3.3 V supply. However, basic peripherals like LCD, ULN 2003 (Motor Driver IC) etc. works on 5V. So AC mains supply is converted into 5V using below mentioned circuit and after that LM 117 is used to convert 5V into 3.3V.

Reset Circuit

Reset button is essential in a system to avoid programming pitfalls and sometimes to manually bring back the system to the initialization mode. MCP 130T is a special IC used for providing stable RESET signal to LPC 2148.

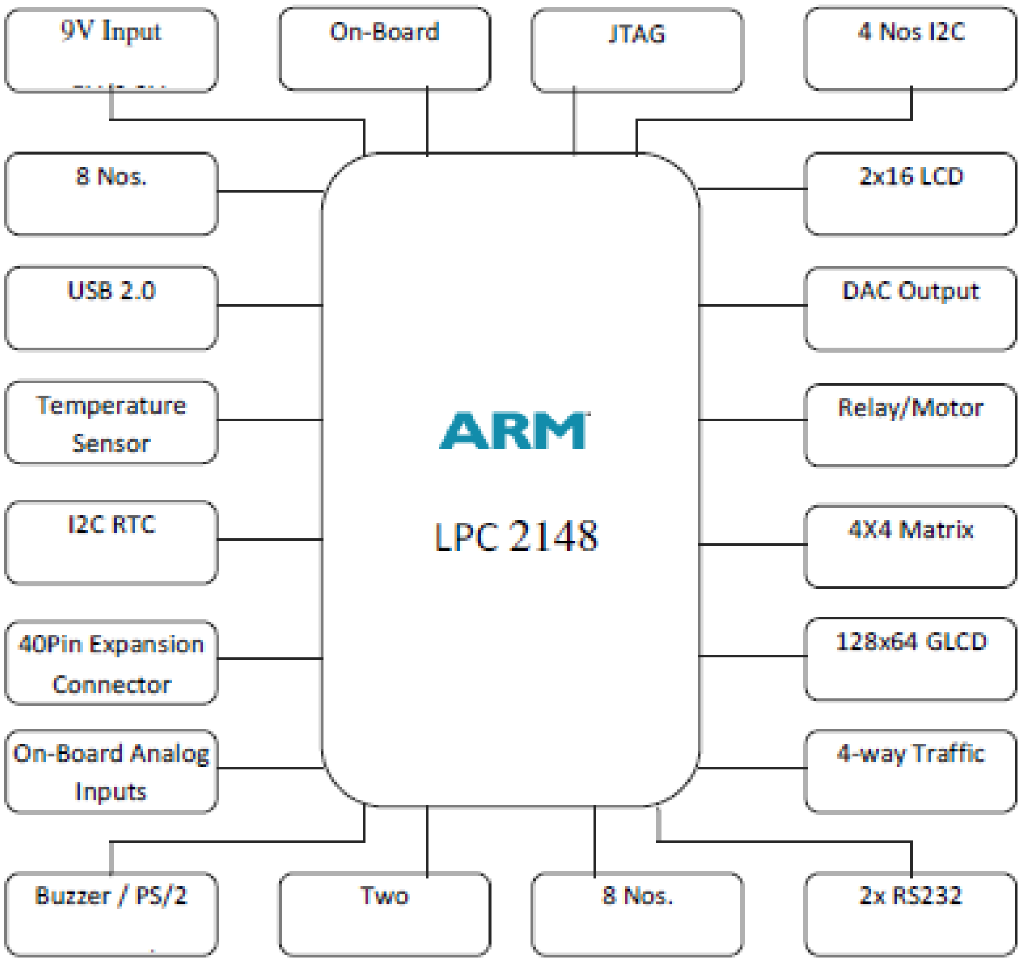


Figure - 3.1: ARM Processor

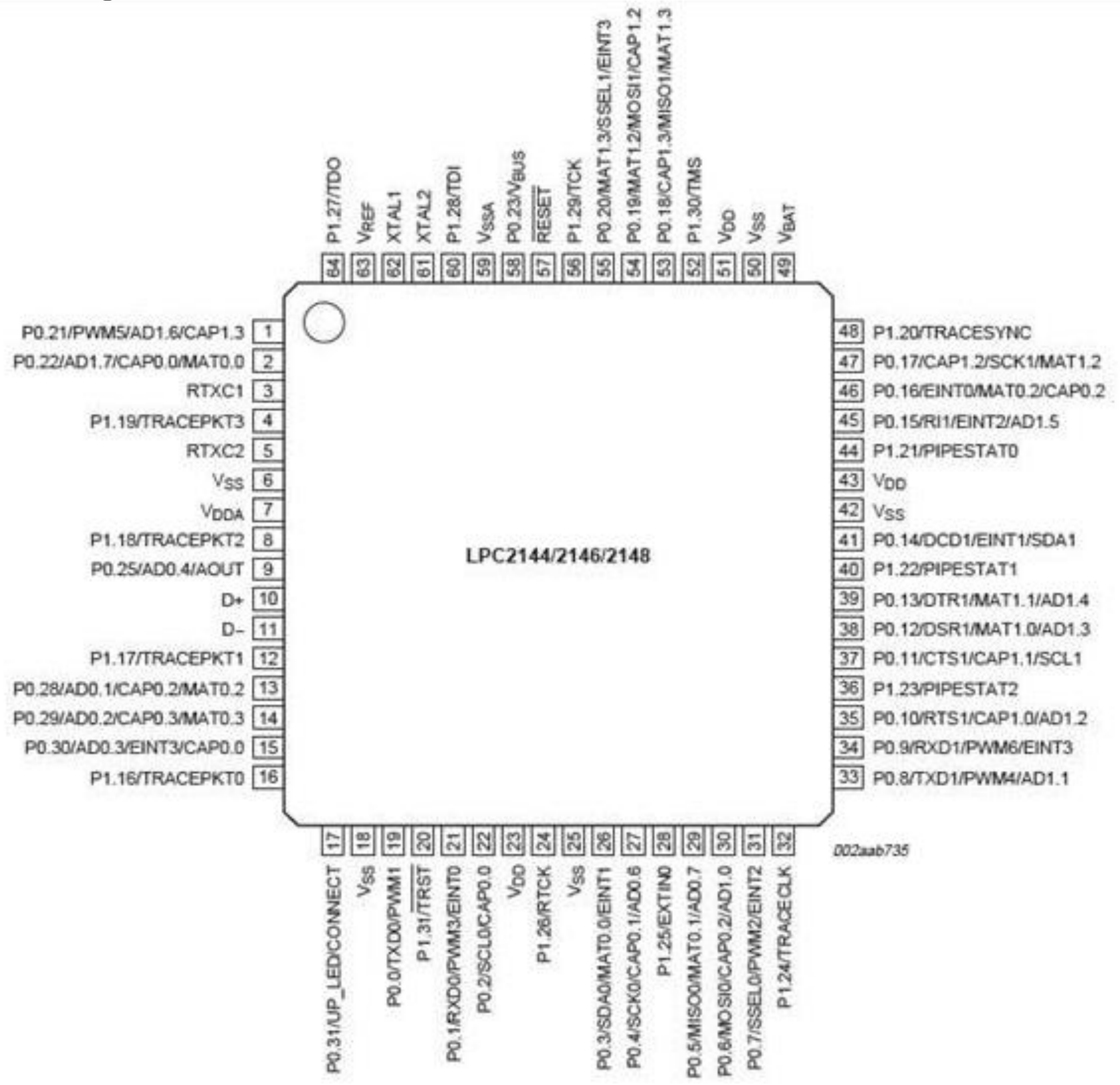
Flash Programming Utility

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.

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)

Pin Configuration



Result:

The ARM processor has been studied successfully.

Exp.

Write and execute C program to blink LEDs using software delay routine in LPC2148 kit

Date:

Aim: To write and execute a C program to blink LEDs using software delay routine in LPC 2148 kit

Apparatus Required:

Keil uVision5 Software

Philips Flah Programmer

LPC 2148 kit

Program:

```
#include "lpc214x.h"

void delay (unsigned int k);

void main(void)
{
IODIR0 = 0xFFFFFFFF; //Configure Port0 as output Port
PINSEL0 = 0;                    //Configure Port0 as General Purpose IO
while(1)
{
IOSET0 = 0x0000ff00;  //Set P0.15-P0.8 to '1'
delay(1000);        //1 sec Delay
IOCLR0 = 0x0000ff00;  //Set P0.15-P0.8 to '0'
delay(1000);        //1 Sec Delay
}
}
//Delay Program
//Input - delay value in milli seconds
void delay(unsigned int k)
{
        unsigned int i,j;
        for (j=0; j<k; j++)
```

```
                for(i = 0; i<=800; i++);
}
```

Output: LEDs P0.15-P0.8 are blinking

Result:

Thus the C program to blink LEDs using software delay routine was written and executed in LPC 2148 kit

Exp.

Write and execute C program to read the switch and display in the LEDs using LPC2148 kit

Date:

Aim: To write and execute C program to read the switch and display in the LEDs using LPC2148 kit

Apparatus Required:

Keil uVision5 Software

Philips Flah Programmer

LPC 2148 kit

Program:

```
#include "lpc214x.h"

int main(void)
{
  unsigned int sw_sts;

  IODIR0 = 0x0000ff00; //Configure Port0
          PINSEL0 = 0;                  //Configure Port0 as General Purose IO
while(1)
{
  sw_sts = IOPIN0;
  IOSET0 = 0x0000ff00;   //Set P0.15-P0.8 to '1'
  IOCLR0 = sw_sts >> 8;  //Set P0.15-P0.8 to '0'
}
}
```

Output: LEDs P0.15-P0.08 displayed the bits entered in the switches

Result:

Thus C program was written read the switch and display in the LEDs using LPC2148 kit

Exp.

Write and execute C program to display a number in seven segment LED in LPC2148 kit

Date:

Aim: To write and execute C program to display a number in seven segment LED in LPC2148 kit

Apparatus Required:

Keil uVision5 Software

Philips Flah Programmer

LPC 2148 kit

Program:

```
//SEVEN SEGMENT LED DISPLAY INTERFACE IN C
/* Program to Count 0-9 and Display it in 7 segment Display (MUX) DS4
 * Display Select DS3 ==> "P0.13"  Enable --> '0',  Disable --> '1'
 * Display Select DS4 ==> "P0.12"  Enable --> '0',  Disable --> '1'
 */

/* Segment Connection Display 1 & 2           Enable --> '1',  Disable --> '0'
 *----------------------------------------------------------------
 * MSB                                          LSB
 *   Dp    G     F     E     D     C     B     A
 * P0.23  P0.22  P0.21  P0.20   P0.19  P0.18  P0.17  P0.16
 *   0     0     0     0     0     1     1     0   --> 6 => '1'
 *----------------------------------------------------------------*/

#include <LPC214X.H>

#define DS3   1<<13            // P0.13
#define DS4   1<<12            // P0.12
#define SEG_CODE 0xFF<<16     // Segment Data from P0.16 to P0.23

unsigned char const seg_dat[]={0x3F, 0x6, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x7, 0x7F,
0x67};
```

```
void delayms(int n)
{
int i,j;
for(i=0;i<n;i++)
 {for(j=0;j<5035;j++)   //5035 for 60Mhz ** 1007 for 12Mhz
    {;}
 }
}

int main (void)
{
  unsigned char count;

PINSEL0 = 0; // Configure Port0 as General Purpose IO => P0.0  to P0.15
PINSEL1 = 0; // Configure Port0 as General Purpose IO => P0.16 to P0.31

IODIR0 = SEG_CODE | DS3 | DS4; //Configure Segment data & Select signal as output

IOSET0 = SEG_CODE | DS3 ;  //Disable DS3 display
IOCLR0 = DS4;  //Enable DS4 Display

count  = 0; //Initialize Count

//Display Count value
IOCLR0 = SEG_CODE;
IOSET0 = seg_dat[count]<<16;

while(1)
 {
            delayms(1000); //1 sec delay
    count++;               //Increment count
    if(count>9) count=0;  //Limit 0-9
    //Display Count value
    IOCLR0 = SEG_CODE;
    IOSET0 = seg_dat[count]<<16;

   }

}
```

Output: 7-Segment display counting from 0 to 9

Result:

Thus C program, was written and executed to display a number in seven segment LED in LPC2148 kit

Exp.

Write and execute C program for serial transmission and reception using on-chip UART in LPC2148 kit.

Date:

Aim: To write and execute C program for serial transmission and reception using on-chip UART in LPC2148 kit.

Apparatus Required:

Keil uVision5 Software

Philips Flah Programmer

LPC 2148 kit

Program:

```
#include <lpc214x.h>

void UART0_Init(void)
{
 PLL0CON = 0;
 PLL0FEED=0xAA;
 PLL0FEED=0x55;
 VPBDIV = 1;
 // Fpclk = 12.000.000 MHz
 // DLM,DLH = Fpclk / (19200*16) = 39 = 0x27
 PINSEL0 |= 0x5; // Select UART0 RXD/TXD
 U0FCR = 0; // Disable FIFO's
 U0LCR = 0x83; // 8N1, enable Divisor latch bit
 U0DLL = 0x27; // baud rate fixed to 19200 @ PCLK = 12 Mhz
 U0DLM = 0;
 U0LCR = 3; // Disable Divisor latch bit
}
/*---------------------------------------------------------------*/
/* Function to send one char. to Serial Port */
void sout(unsigned char dat1)
{
 while(!(U0LSR & 0x20));//Wait for Tx Buffer Empty
 U0THR = dat1;   //Send to UART1
}
/*---------------------------------------------------*/

int main (void)
{  int dat;
   UART0_Init();
 do
  {
   if(U0LSR & 1)  /* Check for RDR (Receiver Data Ready)command */
    {
       dat = U0RBR;// Receive Data from Srial Port
       sout(dat);  // Send Data to Srial Port
            }
   }while(1);
}
```

Output: Data was serially transmitted

Result:

Thus a C program was Written and executed for serial transmission and reception using on-chip UART in LPC2148 kit.

Exp.

Write and execute C program for accessing an internal ADC and display the binary output in LEDS in LPC2148 kit

Date:

Aim: To write and execute C program for accessing an internal ADC and display the binary output in LEDS in LPC2148 kit.

Apparatus Required:

Keil uVision5 Software

Philips Flah Programmer

LPC 2148 kit

Program:

```
#include <LPC214X.H>

#define LEDS  0xFF<<8   //LED => P0.8 to P0.15

////////////////////////////////////////
/*--- ADC Signal Declaration */
////////////////////////////////////////
#define AD0_1 1<< 24
#define CLK_DIV 1<<8
#define PDN 1<<21
#define SOC 1<<24
#define BURST 1<<16
#define DONE 1<<31

/*---------------------------------------------------------*/
//Delay Program
//Input - delay value in milli seconds
void delay(unsigned int k)
{
	unsigned int i,j;
	for (j=0; j<k; j++)
		for(i = 0; i<=800; i++);
}
```

```
/*-----------------------------------------------------------*/
void adc_init()
{
unsigned long int ADC_CH;

            ADC_CH = 0 | 1 << 1; //Channel AD0.1
    AD0CR = SOC | PDN | CLK_DIV | ADC_CH | BURST ;
}
/*-----------------------------------------------------------*/
unsigned int adc_read( unsigned char channel)
{
             unsigned int aval;
             unsigned long int val;

    if (channel == 1) val = AD0DR1;
    else if (channel == 2) val = AD0DR2;
    else if (channel == 3) val = AD0DR3;

    val = val >> 6;
    val = val & 0x3FF;
    aval = val;
    return (aval);
}
/*-----------------------------------------------------------*/
/////////////////////////
/*----Main Program------*/
/////////////////////////
int main(void)
{
unsigned int tp1;

  IODIR0   = LEDS; //Configure Port0 as output Port
  PINSEL0 = 0; //Configure Port0 as General Purpose IO
  PINSEL1 = 0 | AD0_1; // Enable AD0.1
```

```
   adc_init(); //Initialise on-chip ADC

 do
  {  tp1 = adc_read(1); // Channel AD0 0.1
                        tp1 = tp1 >> 2;  // ADC 10 bit But LED 8bit, Truncate lsb 2 bits
    IOSET0 =  LEDS;  //Switch OFF all LEDS
    IOCLR0 = tp1 << 8;  //Set VAlue
    delay(1000);



  }while(1);

}
```

Output: The Potentiometer knob was adjusted to generate Analog input and Digital display is observed

Result:

Thus C program was Written and executed for accessing an internal ADC and display the binary output in LEDS in LPC2148 kit.