(An Autonomous Institute of Government of Maharashtra)
Government College of Engineering, Aurangabad
शासकीय अभियांत्रिकी महाविद्यालय, औरंगाबाद
In pursuit of Technical Excellence
महाराष्ट्र शासन

Department of Electronics and Telecommunication Engineering

ET3014: Lab Embedded Systems

Name: Ritik Nandanwar

Enrollment No.: BE18F04F033

Class: TY EnTC

Batch: T1

Experiment 2: **(A) Write a program to Interface LEDs with LPC 2148.**

(B) Write a program to Interface seven segment display with LPC2148.

Scanned with CamScanner

EXPERIMENT NO. 2(A)

Aim: Write a program to Interface LEDs with LPC 2148.

Apparatus: Keil µ vision 5

Theory:

LPC2148 is a 32-bit Microcontroller based on the ARM7TDMI-S Family. It is manufactured by NXP Semiconductors (formerly Philips) and is one of the widely used and highly successful ARM7 based Microcontroller.

Even though ARM7 family is considered obsolete with the introduction of many advanced processors and controllers, it is one of the best and easiest microcontrollers to work around for beginners in ARM based microcontrollers.

Before going in to the details of the input/output pins, we'll see some of the key features of the LPC2148 microcontroller.

- LPC2148 is a -bit microcontroller based on ARM7 family (ARM7TDMI-S to be specific) and is available in a 64 pin LQFP Package (Low Profile Quad Flat Package).
- It has an on-chip static RAM of 32kB and an on-chip flash memory of 512kB.
- LPC2148 has an on-chip boot loader software that can be used to program the microcontroller using the feature of In-System Programming.
- It also has the feature of in-Application Programming using which the user can program the microcontroller without using the boot loader software.
- It has an embedded in-circuit emulator that helps in real time debugging.
- LPC2148 has two 10-bit ADCs (ADC0 and ADC1) of Successive Approximation type.
- It also has a 10-bit DAC of Resistor String type.

Other features can be found in the official datasheet and user manual.

General Purpose Input Output (GPIO) pins of a microcontroller are the first thing we need to learn before starting its embedded programming as input/output pins are the only way to interface with the microcontroller. GPIO pins can be used for driving loads, reading digital and analog signal, controlling external components, generating triggers for external devices etc.

LPC2148 has two IO ports namely PORT0 (P0) and PORT1 (P1). These two IO ports are of 32-bit wide and are provided by the 64 pins of the microcontroller.

The naming convention of the I/O pins on the LPC2148 Microcontroller is Pa.bc where 'a' is the number of the port i.e. 0 or 1 (as LPC2148 has only two ports) and 'bc' is the number of the pin in the port a. For example, P0.1 indicates pin number 1 of PORT0 and P1.10 indicates pin number 10 of PORT1.

PORT0 is a 32-bit wide input/output port with dedicated direction control bits for each physical pin. Out of the 32 pins, 28 pins can be used as general purpose bidirectional I/O pins. Pin P0.31 is an output only pin. Pins P0.24, P0.26 and P0.27 of PORT0 are not available.

PORT1 is also a 32-bit wide input/output port. In PORT1, pins P1.0 to P1.15 are not available and pins P1.16 to P1.31 are the available general purpose input/output pins.

Most of the pins in both the I/O ports of the LPC2148 have more than one function i.e. they are multiplexed with different functions. For example, Pin 19 of the LPC2148 has three functions namely P0.0, a general purpose I/O pin, TXD0, the transmitter O/P for UART0 and PWM1, the pulse width modulator O/P 1.

At any point of operation, each pin can have a single function and the function can be selected with the help of three Configuration Registers which control the multiplexers to allow connection between the external pin and the on-chip peripheral.

The configuration register is called PINSEL and is classified in to three registers: PINSEL0, PINSEL1 and PINSEL2. These configuration registers are of 32-bit wide. Any pin on the LPC2148 can have a maximum of 4 functions. Hence in order to select one of the four functions, two corresponding bits of the PINSEL register are needed. So, a 32-bit PINSEL register can control 16 pins with 2-bits to control each pin.

PINSEL0 controls PORT0 pins P0.0 to P0.15, PINSEL1 controls PORT0 pins P0.16 to P0.31 and PINSEL2 controls PORT1 pins P1.16 to P1.31.

The following table shows the PINSEL0 and corresponding functions on the PORT0.

PINSEL0	*Pin Name*	*Function when 00*	*Function when 01*	*Function when 10*	*Function when 11*	*Reset Value*
1:0	P0.0	GPIO Port 0.0	TXD (UART0)	PWM1	Reserved	00
3:2	P0.1	GPIO Port 0.1	RXD (UART0)	PWM3	EINT0	00
5:4	P0.2	GPIO Port 0.2	SCL0 (I^2C0)	Capture 0.0 (Timer 0)	Reserved	00
7:6	P0.3	GPIO Port 0.3	SDA0 (I^2C0)	Match 0.0 (Timer 0)	EINT1	00
9:8	P0.4	GPIO Port 0.4	SCK0 (SPI0)	Capture 0.1 (Timer 0)	AD0.6	00
11:10	P0.5	GPIO Port 0.5	MISO0 (SPI0)	Match 0.1 (Timer 0)	AD0.7	00
13:12	P0.6	GPIO Port 0.6	MOSI0 (SPI0)	Capture 0.2 (Timer 0)	AD1.0	00
15:14	P0.7	GPIO Port 0.7	SSEL0 (SPI0)	PWM2	EINT2	00
17:16	P0.8	GPIO Port 0.8	TXD (UART1)	PWM4	AD1.1	00
19:18	P0.9	GPIO Port 0.9	RXD (UART1)	PWM6	EINT3	00
21:20	P0.10	GPIO Port 0.10	RTS (UART1)	Capture 1.0 (Timer 1)	AD1.2	00
23:22	P0.11	GPIO Port 0.11	CTS (UART1)	Capture 1.1 (Timer 1)	SCL1 (I^2C1)	00
25:24	P0.12	GPIO Port 0.12	DSR (UART1)	Match 1.0 (Timer 1)	AD1.3	00
27:26	P0.13	GPIO Port 0.13	DTR (UART1)	Match 1.1 (Timer 1)	AD1.4	00
29:28	P0.14	GPIO Port 0.14	DCD (UART1)	EINT1	SDA1 (I^2C1)	00
31:30	P0.15	GPIO Port 0.15	RI (UART1)	EINT2	AD1.5	00

The default function of all the Pins is GPIO. But it is a good programming practice to mention "PINSEL0=0" in order to select the GPIO function of the Pins.

GPIO function is the most frequently used functionality of the microcontroller. The GPIO function in both the Ports are controlled by a set of 4 registers: IOPIN, IODIR, IOSET and IOCLR.

IOPIN: It is a GPIO Port Pin Value register and can be used to read or write values directly to the pin. The status of the Pins that are configured as GPIO can always be read from this register irrespective of the direction set on the pin (Input or Output).

The syntax for this register is IOxPIN, where 'x' is the port number i.e. IO0PIN for PORT0 and IO1PIN for PORT1.

IODIR: It is a GPIO Port Direction Control register and is used to set the direction i.e. either input or output of individual pins. When a bit in this register is set to '0', the corresponding pin in the microcontroller is configured as Input. Similarly, when a bit is set as '1', the corresponding pin is configured as Output.
The syntax for this register is IOxDIR, where 'x' is the port number i.e. IO0DIR for PORT0 and IO1DIR for PORT1.

IOSET: It is a GPIO Port Output Set Register and can be used to set the value of a GPIO pin that is configured as output to High (Logic 1). When a bit in the IOSET register is set to '1', the corresponding pin is set to Logic 1. Setting a bit '0' in this register has no effect on the pin.
The syntax for this register is IOxSET, where 'x' is the port number i.e. IO0SET for PORT0 and IO1SET for PORT1.

IOCLR: It is a GPIO Port Output Clear Register and can be used to set the value of a GPIO pin that is configured as output to Low (Logic 0). When a bit in the IOCLR register is set to '1', the corresponding pin in the respective Port is set to Logic 0 and at the same time clears the corresponding bit in the IOSET register. Setting '0' in the IOCLR has no effect on the pin.
The syntax for this register is IOxCLR, where 'x' is the port number i.e. IO0CLR for PORT0 and IO1CLR for PORT1.

An important note to remember is that since the LPC2148 is a 32-bit microcontroller, the length of all the registers mentioned is also 32-bits. Each bit in the above mentioned registers is directly linked to the corresponding pin in the microcontroller i.e. bit 'a' in IO0SET corresponds to Pin 'a' in the PORT0.
Registers in LPC2148 follow Big Endian format i.e. bit 0 is the LSB on the extreme right of the register and bit 31 is the MSB on the extreme left of the register.
Another important note is that when reset, all the pins are set as GPIO pins and the direction of each pin is set as Input.
Now, we'll see how to use the above mentioned registers in programming. First, is to set the direction of a pin. For example, if we want to set 4th pin of PORT0 i.e. P0.3 as output, then it can be set in various ways as shown below.

Method 1: IO0DIR = (1<<3);

This is a direct assignment method where the binary value (1) is set directly on the pin. All the other pins are set to 0. This method should be avoided as the value is directly being assigned in the register and while P0.3 is assigned as '1', all the other pins are forced to be assigned '0'.
An alternative to this method is ORing the register and then assigning the value. This can be done in two ways.

Method 2: IO0DIR | = 0x00000008;

In this method, the hexadecimal value of the register is assigned after ORing the register with itself. In this way, the other pins other than the desired pin (P0.3 in this case) are not affected. This method is useful if we want to assign many pins without affecting the other pins.

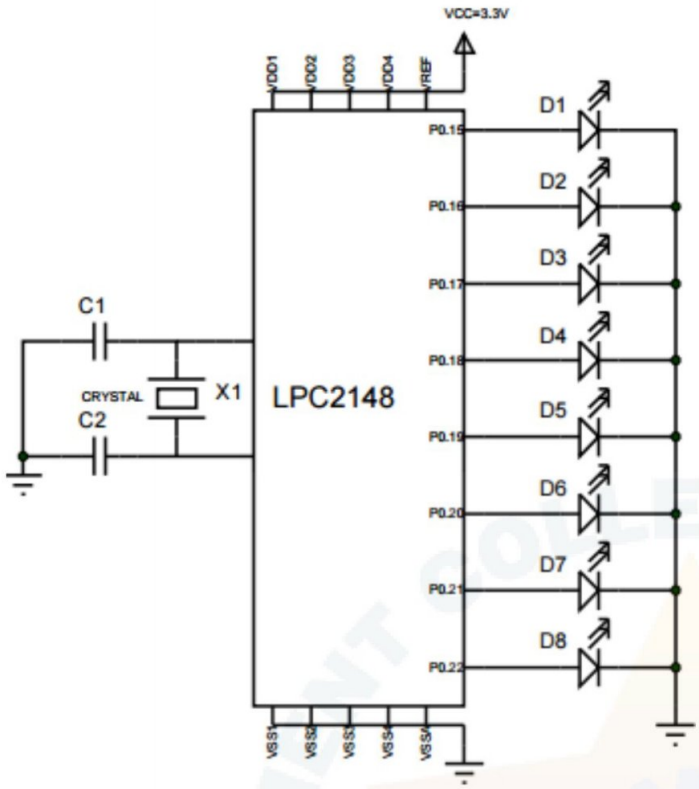
Method 3: IO0DIR | = (1<<3);

This is similar to the above method except that only a single pin is affected.
Other registers can also be set using the same methods. Now, we'll see an example of setting pin 15 of PORT0 i.e. P0.15 as output and drive the pin High.
For this we need to use two registers: IODIR and IOSET.
IO0DIR | = (1<<15); // Configuring P0.15 as output.

IO0SET | = (1<<15); // Make the O/P pin P0.15 as High
We'll see another example where pin 11 of PORT0 is set as output and the output of this pin is set to logic 1 and then to logic 0.
For this we need to use three registers: IODIR, IOSET and IOCLR.
IO0DIR | = (1<<11); // Configuring P0.11 as output.
IO0SET | = (1<<11); // Make the O/P pin P0.11 as High.
IO0CLR | = (1<<11); // Make the O/P pin P0.11 as Low.
Now, we'll see an example where more than one pin has to be set as output and the value of that pin must be HIGH. Consider pins 7 and 14 of PORT0 (P0.7 and P0.14).
IO0DIR | = (1<<7) | (1<<14); // Configuring pins P0.7 and P0.14 as output.
IO0SET | = (1<<7) | (1<<14); // Make the O/P of pins P0.7 and P0.14 as High.
Once we have seen how to set the direction of pins, setting the pin as High or Low, now we'll jump into real time embedded programming. Similar to "Hello, World" program in C Language, blinking an LED is the basic program in embedded system.
The IDE we are going to use for the development of LPC2148 based projects is Keil µVision. All the registers for LPC214X series of microcontrollers are defined in the "lpc214x.h" header file.
In this example, we'll blink the LEDs repeatedly that are connected to PORT1 pins of the MCU.

```
#include <lpc214x.h>
int delay;
int main (void)
{
PINSEL2 = 0x00000000;
IO1DIR = 0xFFFFFFFF; // All the pins of PORT1 are configured as Output
while (1)
{
IO1SET = 0xFFFFFFFF; // Set Logic 1 to all the PORT1 pins i.e. turn on LEDs
for (delay = 0; delay<500000; delay++)
IO1CLR = 0xFFFFFFFF; // Set Logic 0 to all the PORT1 pins i.e. turn off LEDs
for (delay = 0; delay<500000; delay++)
}
return 0;
}
```

The program is self-explanatory. First, the PORT1 pins are configured as outputs using IO1DIR register. Then in an infinite loop, the pins (or LEDs connected to them) are turned ON using IO1SET register and turned OFF using IO1CLR register. A delay is introduced between the turning ON and OFF of the LEDs using a "for" loop, so that the blinking of LEDs is visible.

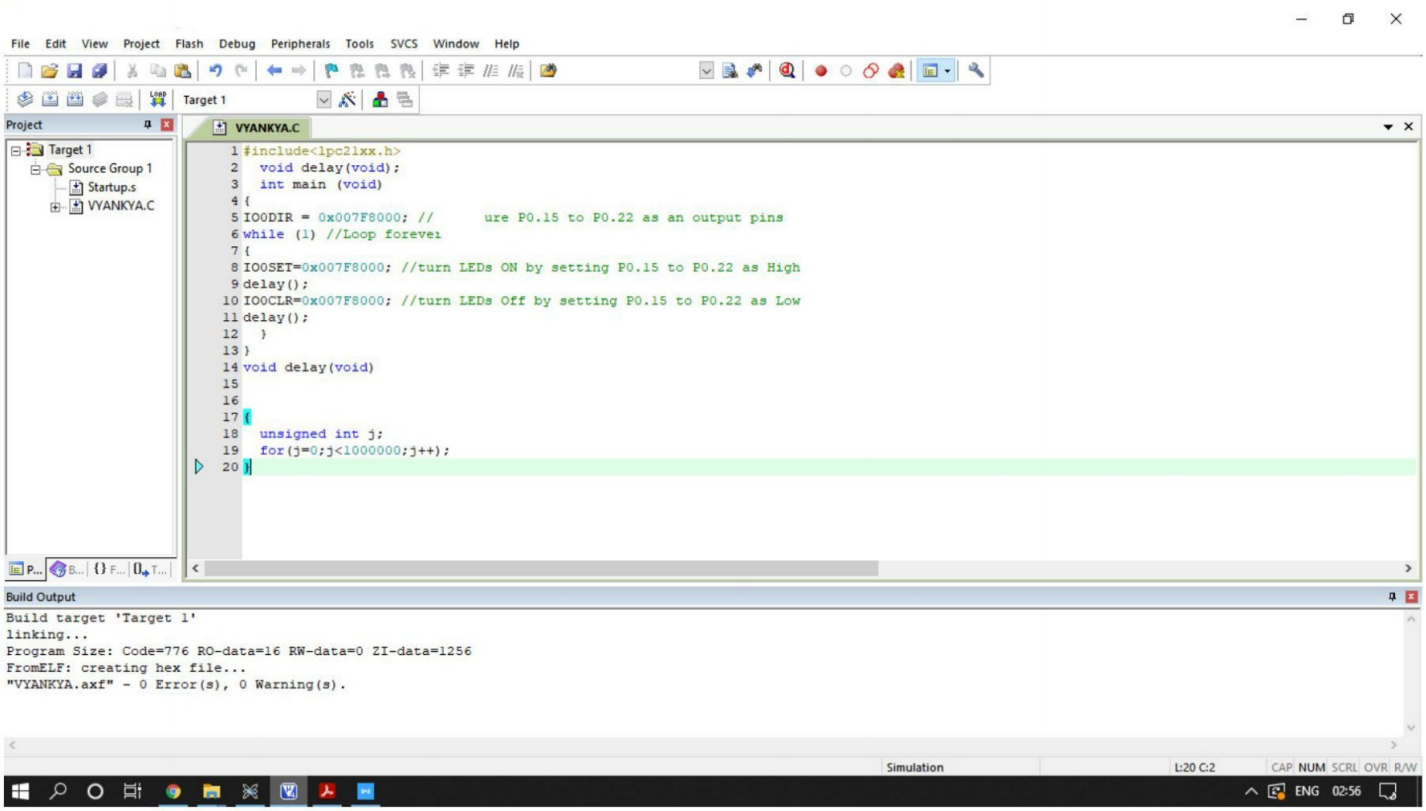


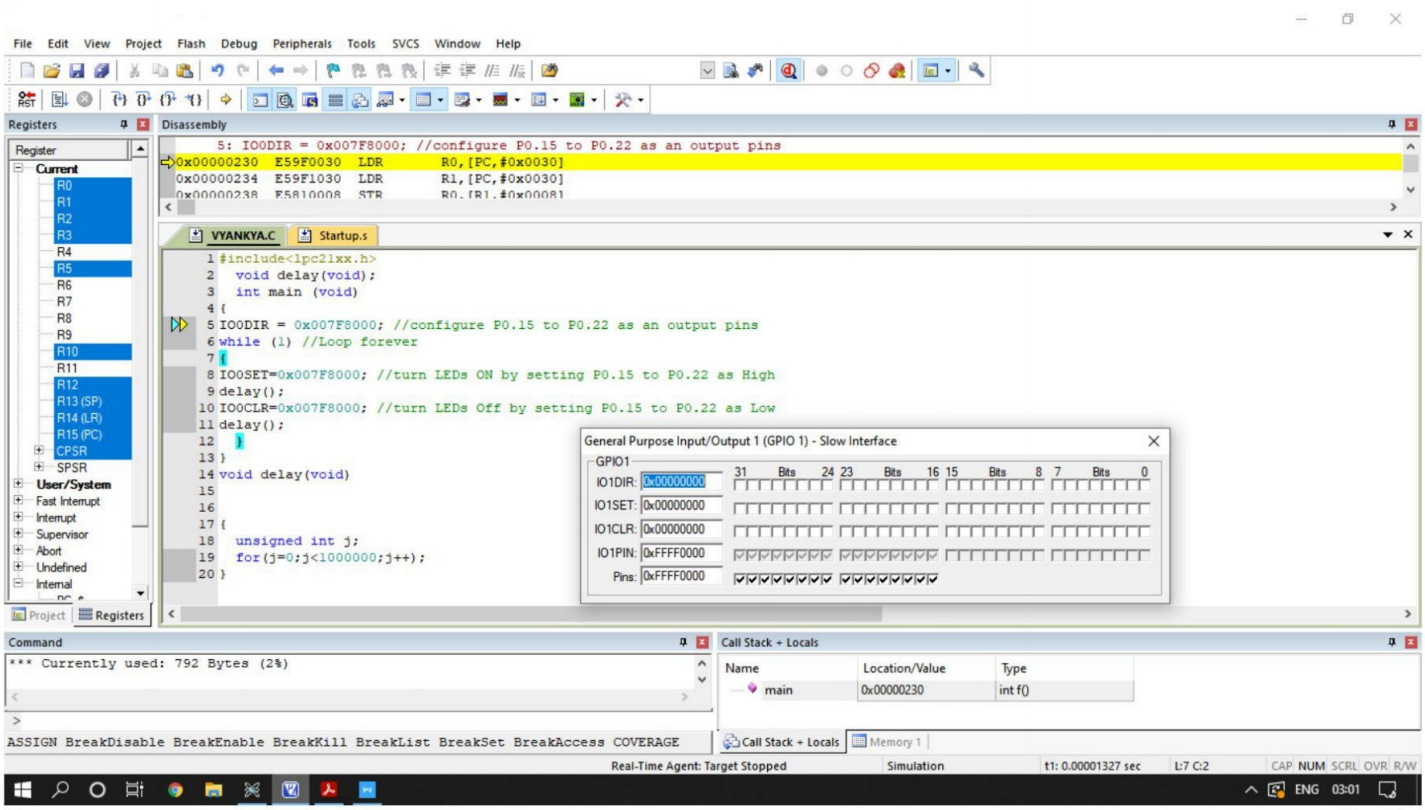
Program:

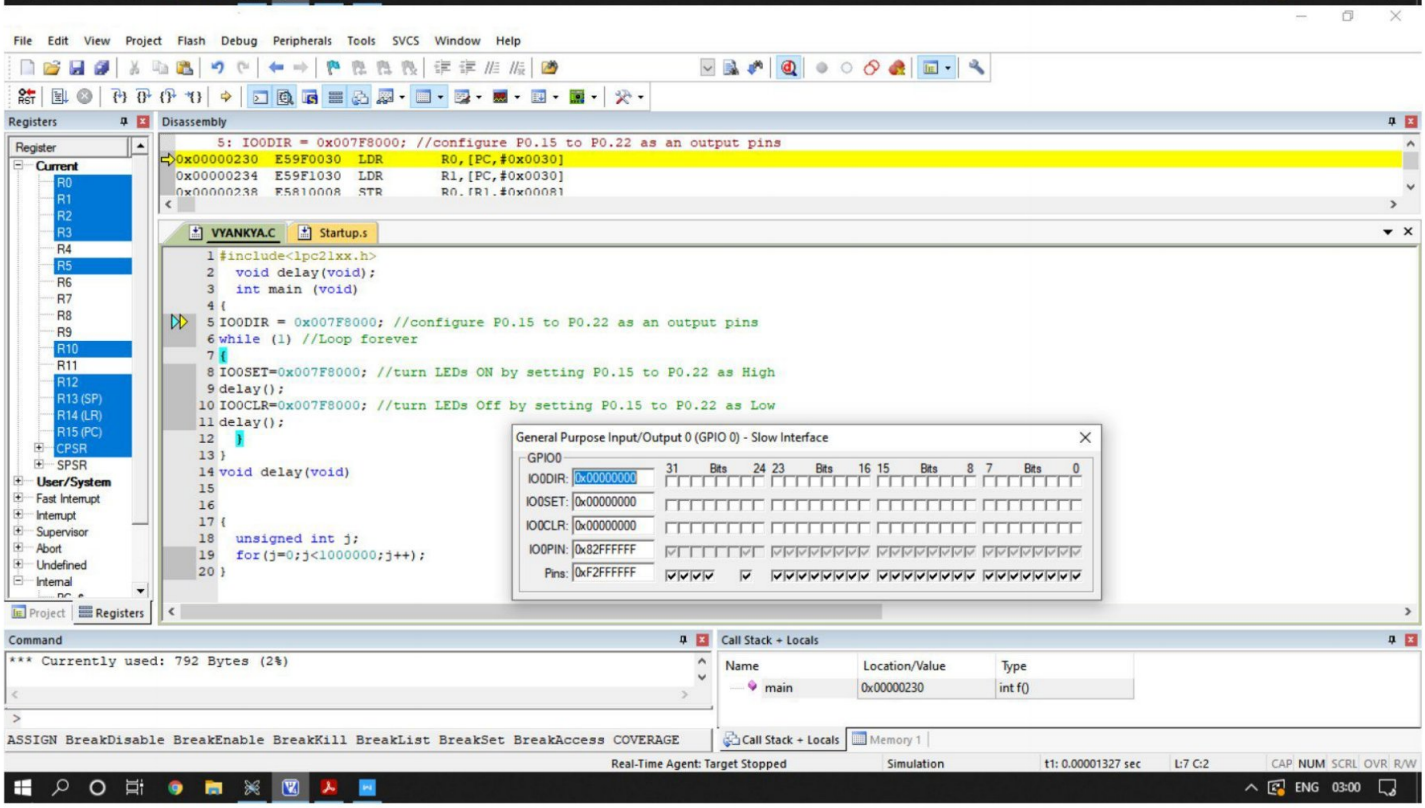
```
#include<lpc21xx.h>
void delay(void);
int main (void)
{
        IO0DIR = 0x007F8000; //configure P0.15 to P0.22 as an output pins
        while (1)                       //Loop forever
        {
                IO0SET=0x007F8000; //turn LEDs ON by setting P0.15 to P0.22 as High
                delay();
                IO0CLR=0x007F8000; //turn LEDs Off by setting P0.15 to P0.22 as Low
                delay();
        }
}
void delay(void)
{
 unsigned int j;
 for(j=0;j<1000000;j++);
}
```

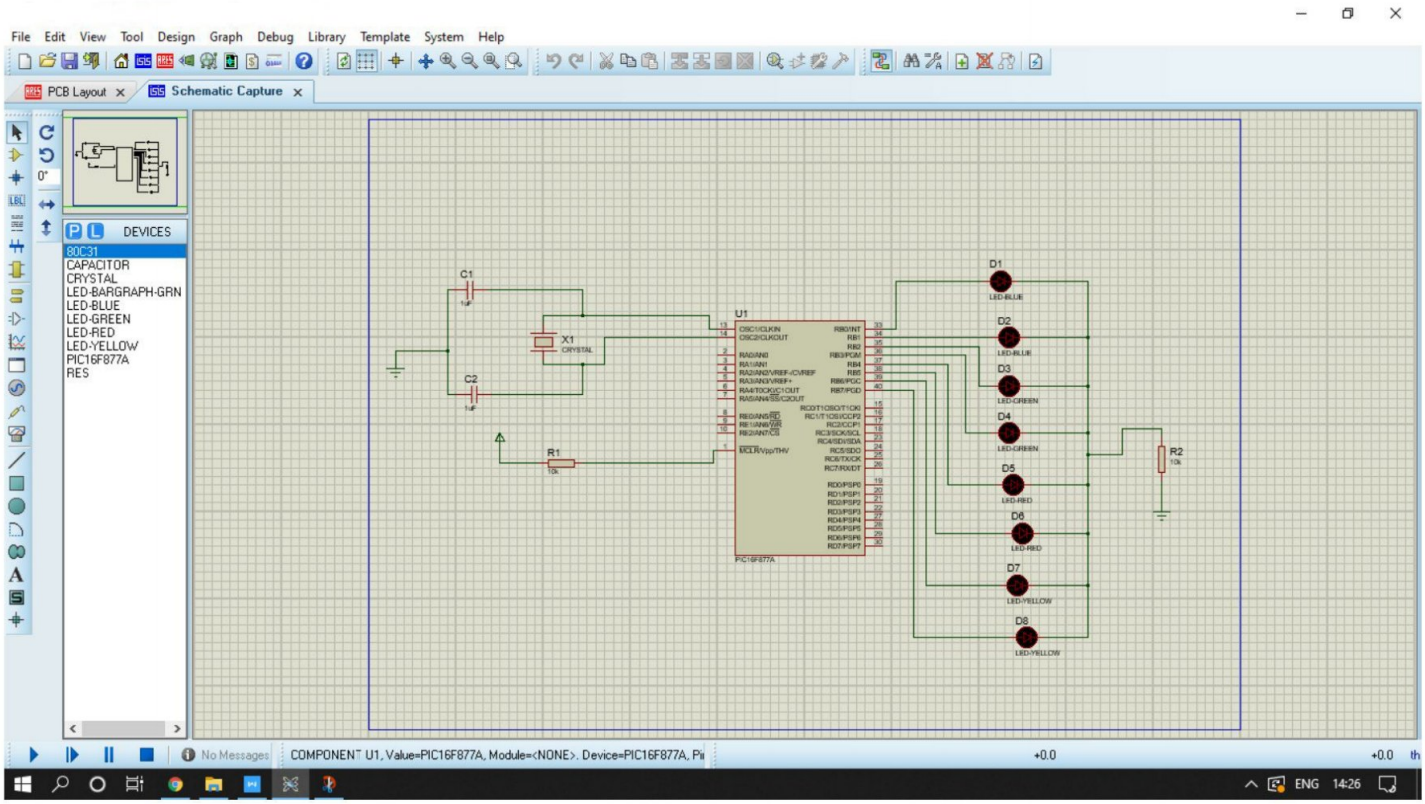
Output:

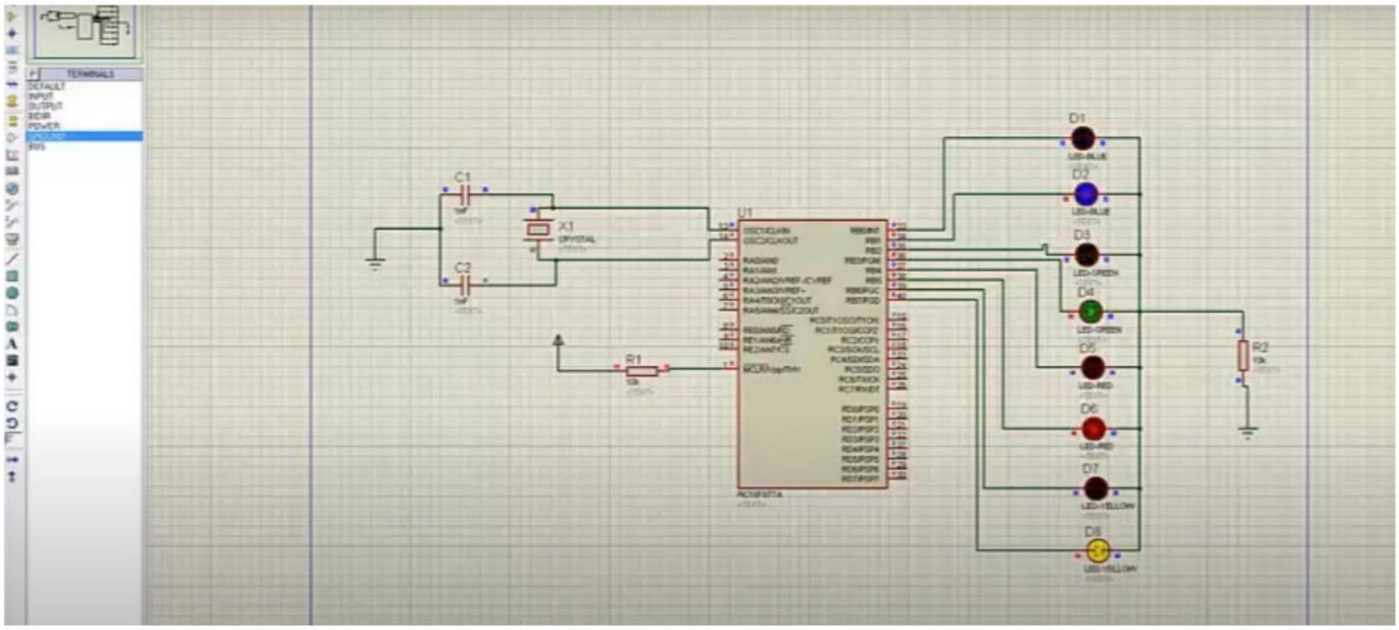
Note: Screenshot of **Keil** window as well as **Proteus** window and also schematic diagram also has to be attached in the output.











Conclusion:

We interfaced LED using Keil and Proteus.

EXPERIMENT NO. 2(B)

Aim: Write a program to Interface seven segment display with LPC 2148.

Apparatus: Keil µ vision 5

Theory:

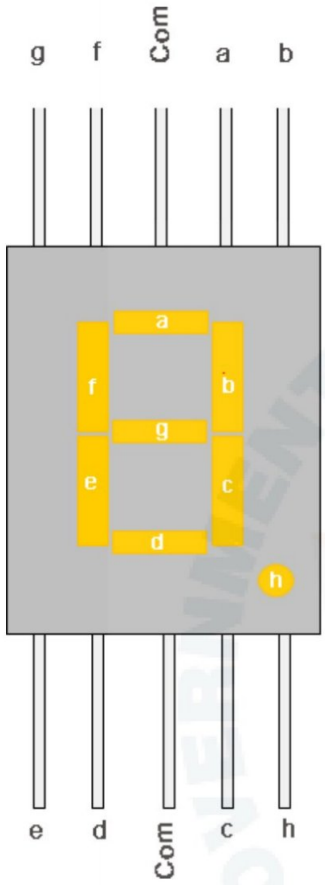
Display is a very important part of any Embedded System Application as it helps users to know the status of the system and also shows the output or any warning message generated by the system. There are many types of displays used in electronics like 7-segment display, LCD display, TFT touch screen display, LED display etc.

7-Segment Display

7 segment displays are among the simplest display units to display the numbers and characters. It is generally used to display numbers and has brighter illumination and simpler construction than dot matrix display. And because of brighter illumination, the output can be viewed from larger distance than LCD. As shown in the above image of a 7-segment display, it consists of 8 LEDs, each LED used to illuminate one segment of unit and the 8thLED used to illuminate DOT in 7 segment display. 8thLED is used when two or more 7-segment modules are used, for example to display (0.1). A single module is used to display single digit or character. To display more than one digit or character, multiple 7-segments are used.

Pins of 7-Segment Display

There are 10 pins, in which 8 pins are used to refer a,b,c,d,e,f,g and h/dp, the two middle pins are common anode/cathode of all he LEDs. These common anode/cathode are internally shorted so we need to connect only one COM pin



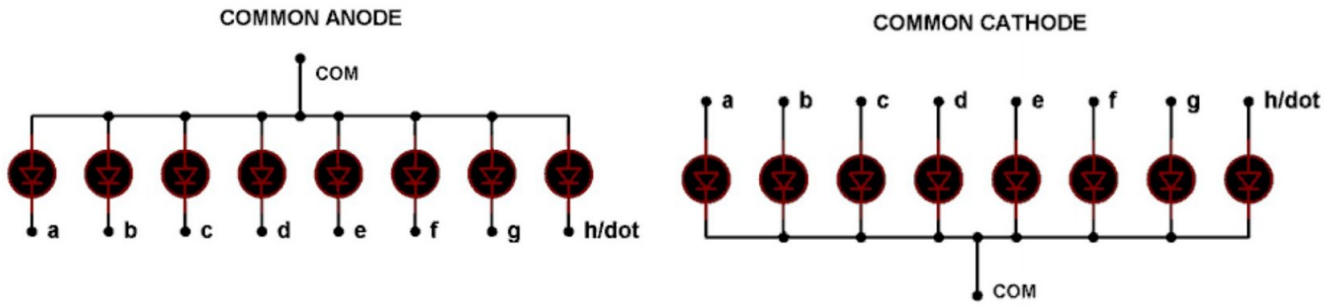
Depending upon connection we classify 7-Segment in two types:

Common Cathode

In this all the Negative terminals (cathode) of all the 8 LEDs are connected together (see diagram below), named as COM. And all the positive terminals are left alone or connected to the microcontroller pins. If we use microcontroller we set logic HIGH to illuminate the particular and set LOW to turn OFF LED.

Common Anode

In this all the positive terminals (Anodes) of all the 8 LEDs are connected together, named as COM. And all the negative thermals are left alone or connected to the microcontroller pins. If we use microcontroller we set logic LOW to illuminate the particular and set logic High to turn OFF LED.



Internal connections of 7 Segment Display

So depending upon the pin value, a particular segment or line of 7 segments can be turned on or Off to display desired number or alphabet. For example to display 0 digit, we must set pins ABCDEF as HIGH and only G as LOW. As ABCDEF LEDs are ON and G is OFF this forms the 0 digit in 7-segment module. (This is for common cathode, for common anode it is opposite).

Below table shows the HEX values and corresponding digit according to LPC2148 pins for common cathode configuration.

Digit	HEX Values for LPC2148	A	B	C	D	E	F	G
0	0xF3	1	1	1	1	1	1	0
1	0x12	0	1	1	0	0	0	0
2	0x163	1	1	0	1	1	0	1
3	0x133	1	1	1	1	0	0	1
4	0x192	0	1	1	0	0	1	1
5	0x1B1	1	0	1	1	0	1	1
6	0x1F1	1	0	1	1	1	1	1
7	0x13	1	1	1	0	0	1	0
8	0x1F3	1	1	1	1	1	1	1
9	0x1B3	1	1	1	1	0	1	1

The table below shows the circuit connections between 7-Segment module & LPC2148

Seven Segment Module Pins	LPC2148 Pins
A	P0.0
B	P0.1
C	P0.4
D	P0.5
E	P0.6
F	P0.7
G	P0.8
Common	GND

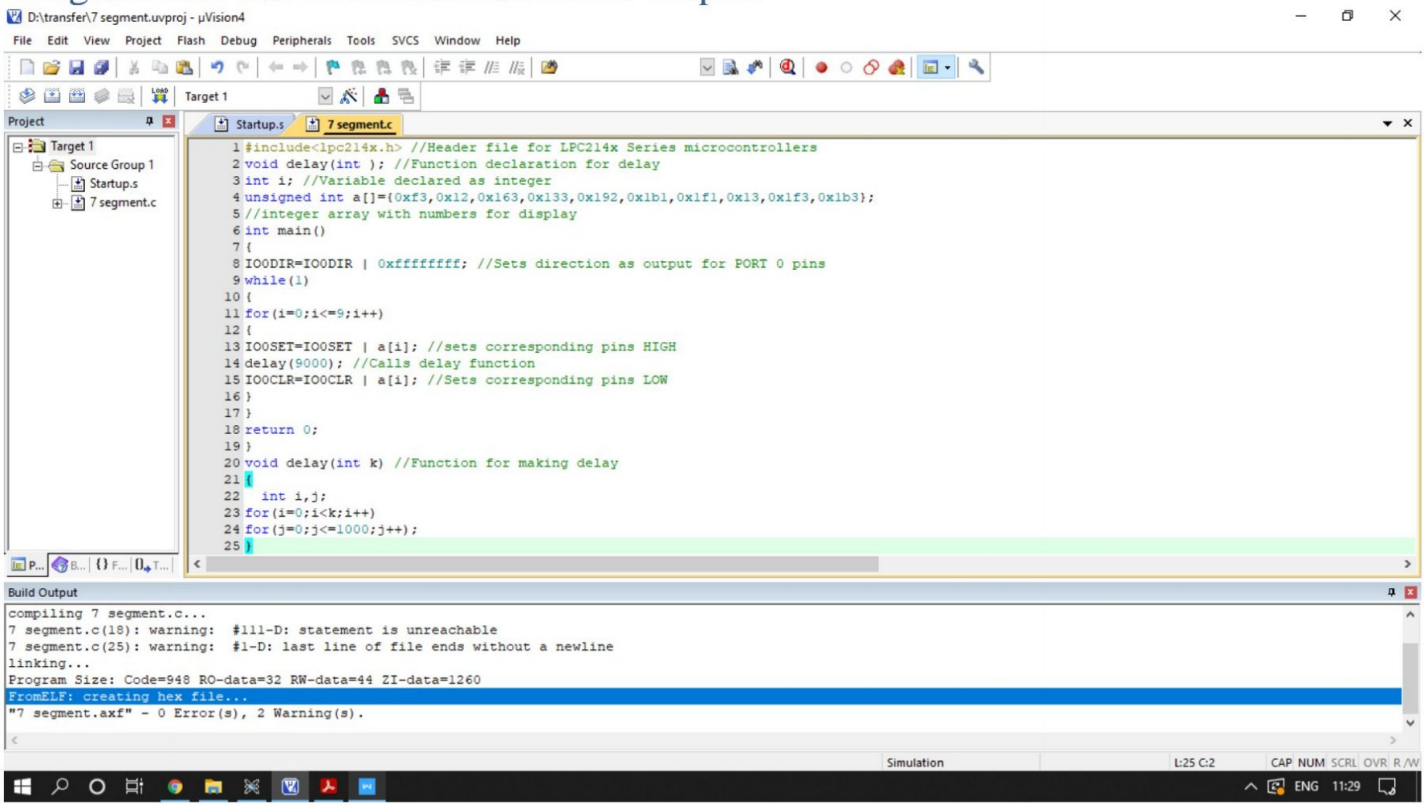
Program:

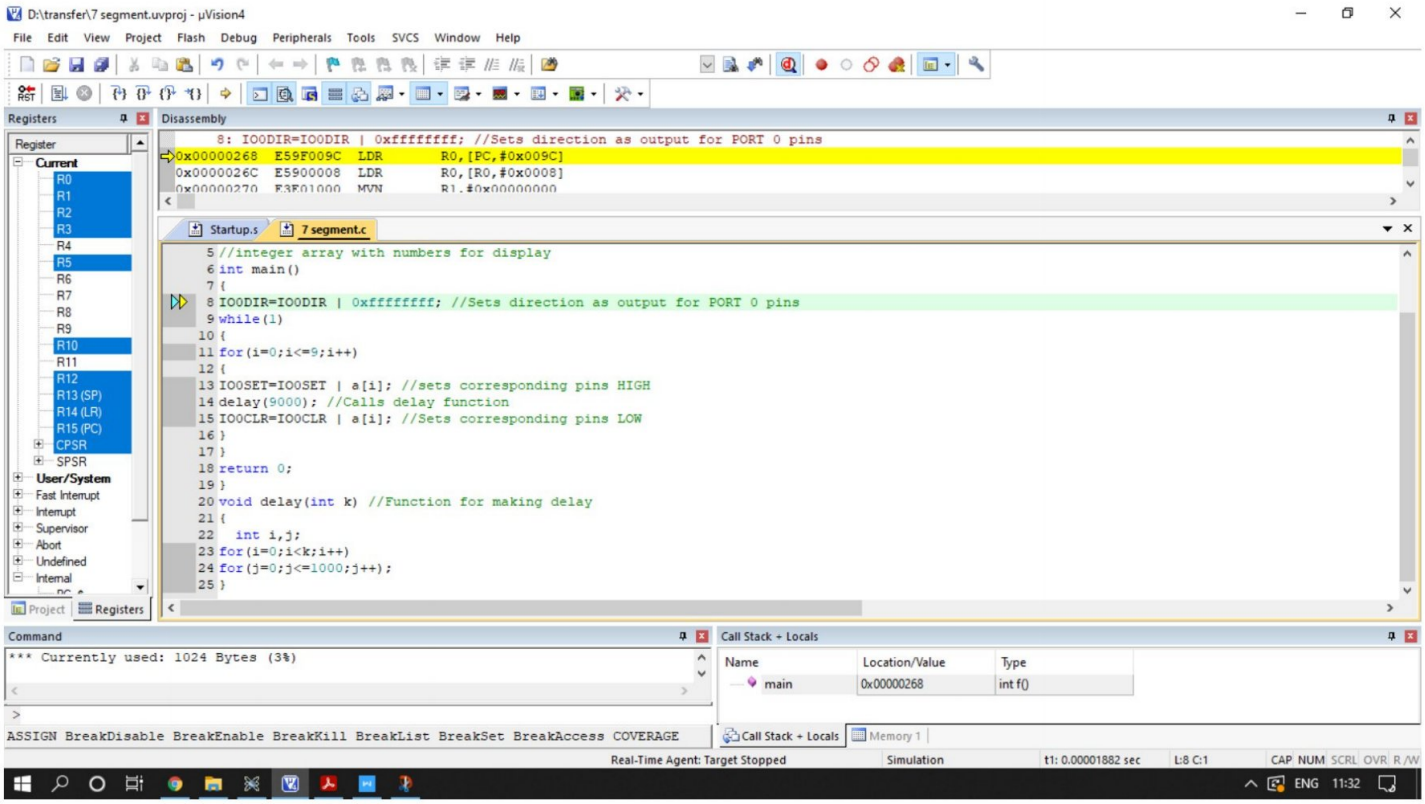
```
#include<lpc214x.h>          //Header file for LPC214x Series microcontrollers
void delay(int );            //Function declaration for delay
int i;                       //Variable declared as integer
unsigned int a[]={0xf3,0x12,0x163,0x133,0x192,0x1b1,0x1f1,0x13,0x1f3,0x1b3};
 //integer array with numbers for display
int main()
{
   IO0DIR=IO0DIR | 0xffffffff;          //Sets direction as output for PORT 0 pins
   while(1)
   {
      for(i=0;i<=9;i++)
      {
         IO0SET=IO0SET | a[i];          //sets corresponding pins HIGH
         delay(9000);                   //Calls delay function
         IO0CLR=IO0CLR | a[i];          //Sets corresponding pins LOW
      }
   }
   return 0;
}
void delay(int k)            //Function for making delay
{
```

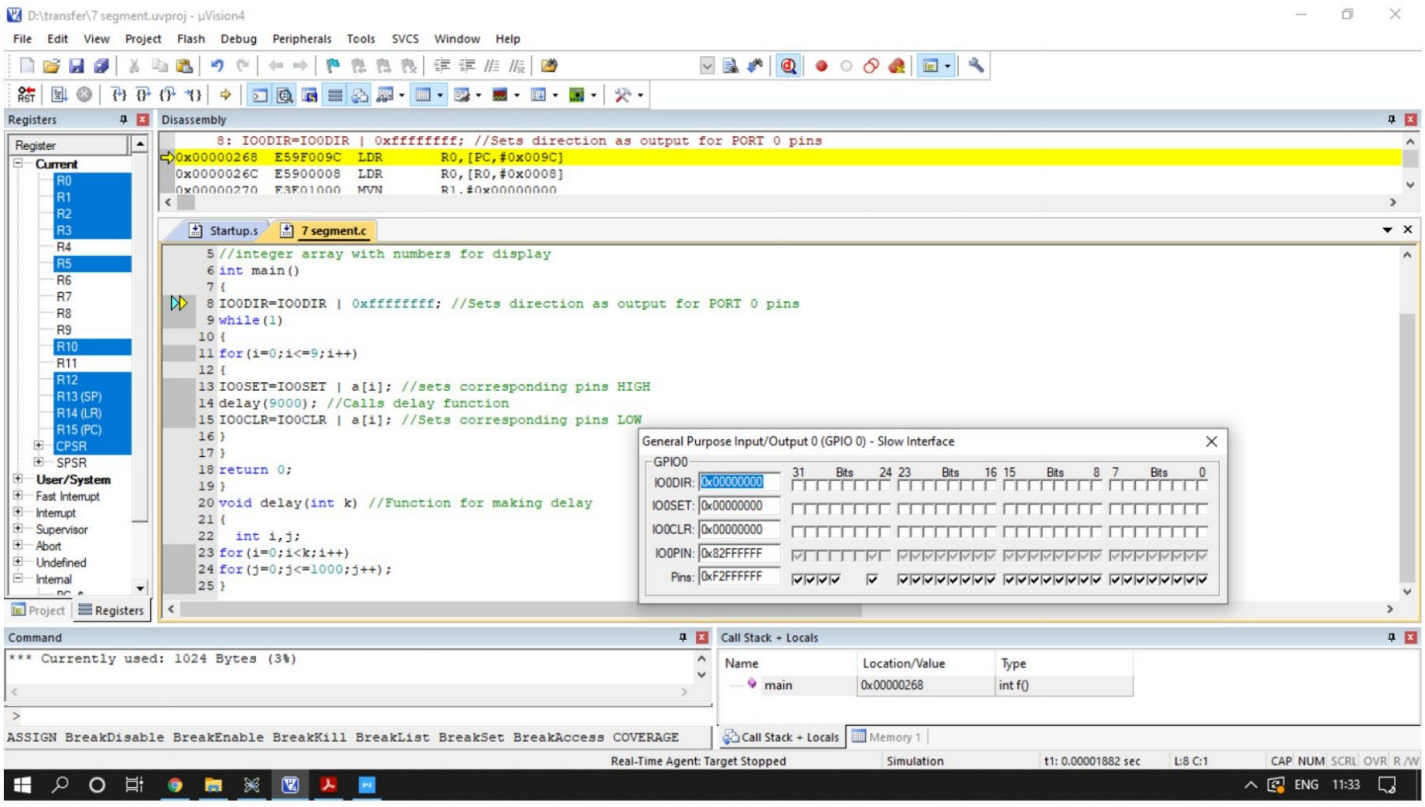
```
    int i,j;
    for(i=0;i<k;i++)
    for(j=0;j<=1000;j++);
}
```

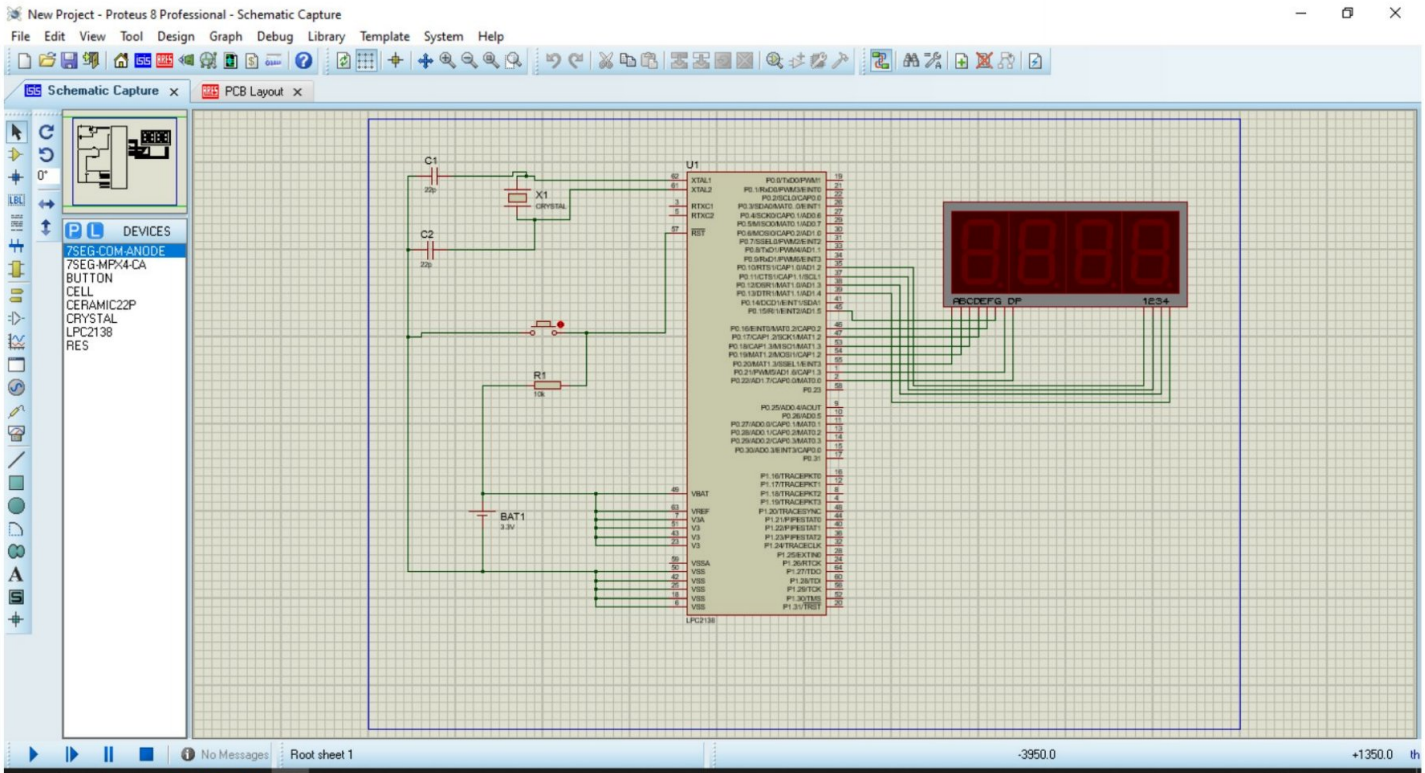
Output:

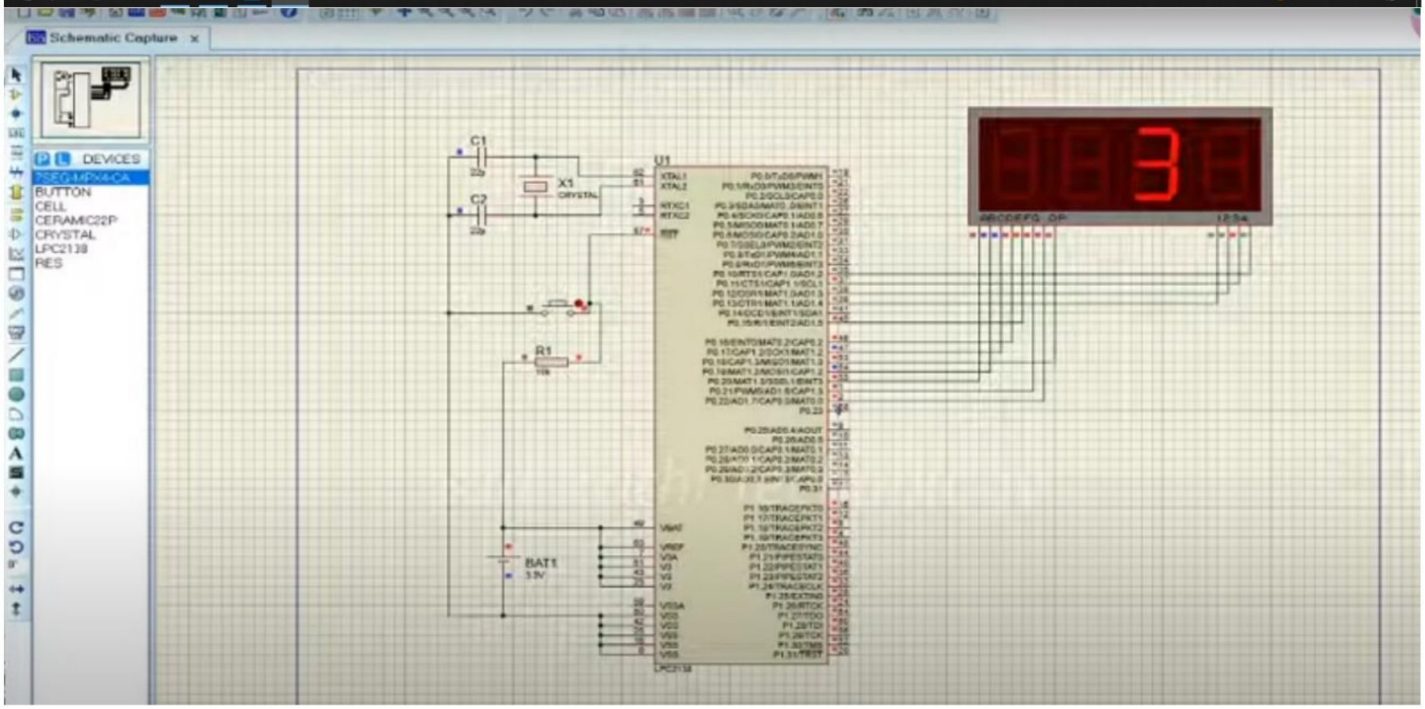
Note: Screenshot of **Keil** window as well as **Proteus** window and also schematic diagram also has to be attached in the output.

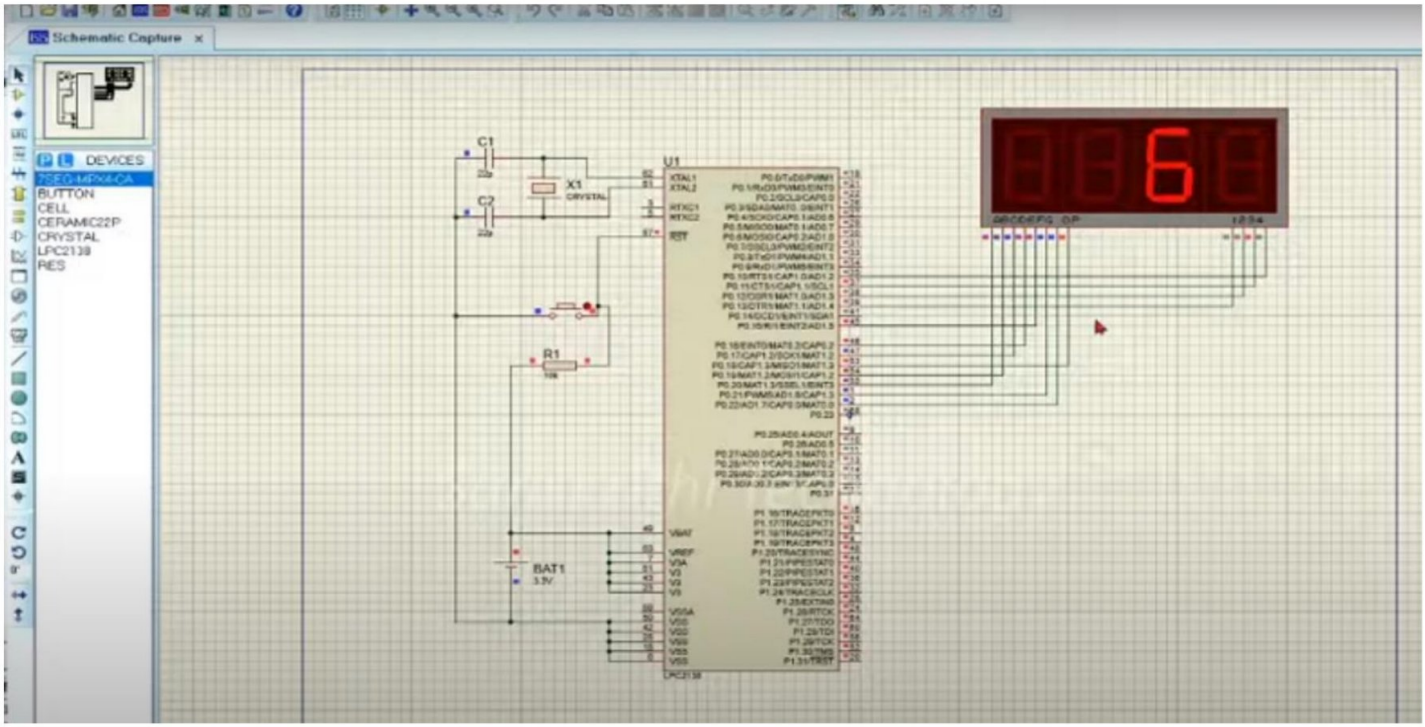












Conclusion:

We interfaced 7 segment display using keil and proteus.

(An Autonomous Institute of Government of Maharashtra)
Government College of Engineering, Aurangabad
शासकीय अभियांत्रिकी महाविद्यालय, औरंगाबाद
In pursuit of Technical Excellence
महाराष्ट्र शासन

Department of Electronics and Telecommunication Engineering

ET3014: Lab Embedded Systems

Name: Ritik Nandanwar

Enrollment No.: BE18F04F033

Class: TY EnTC

Batch: T1

Experiment 2: **(A) Write a program to Interface LEDs with LPC 2148.**

(B) Write a program to Interface seven segment display with LPC2148.

Scanned with CamScanner