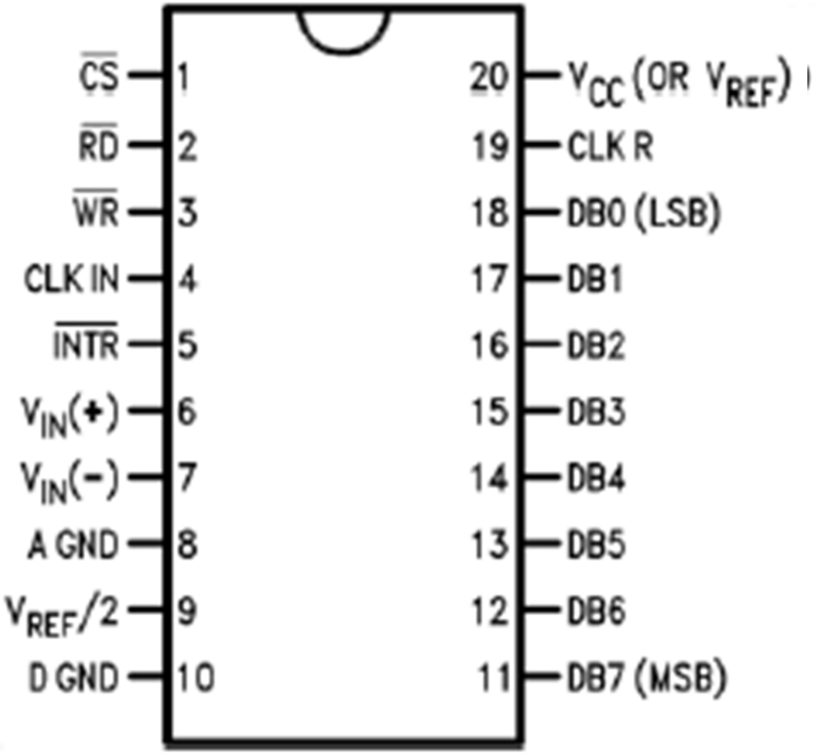
Microprocessors and Microcontrollers (EE-231)

Lab-16

Objective

- Interfacing ADC 0804 to Microcontroller

The ADC0804

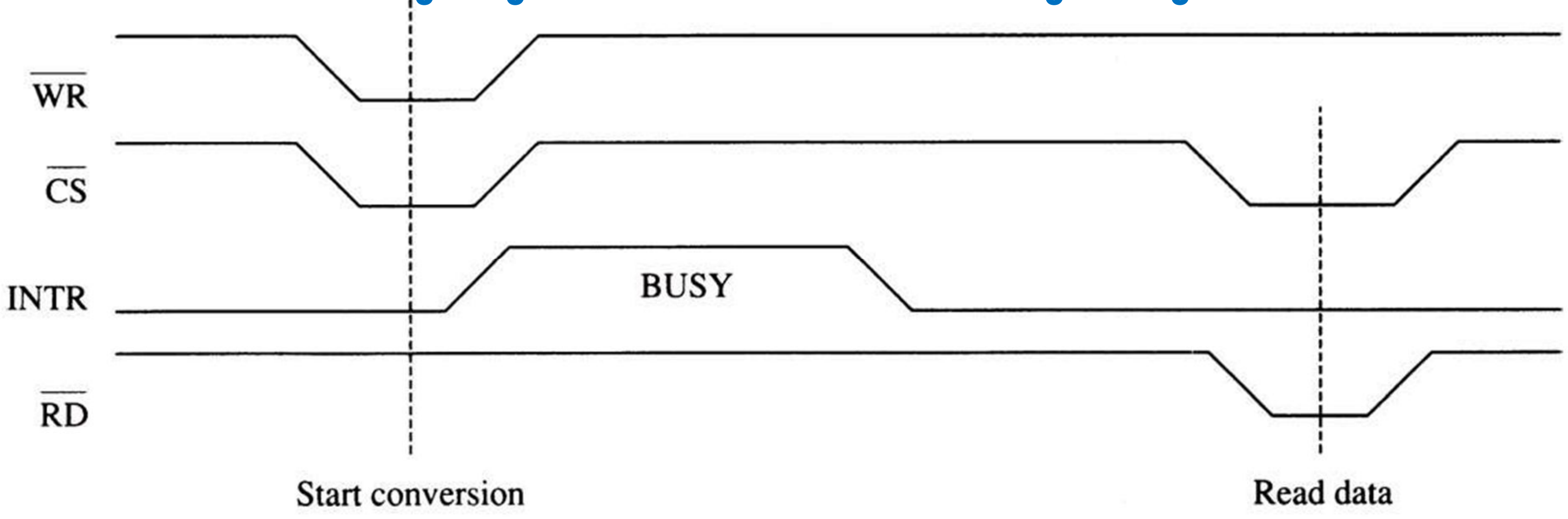


$\overline{CS}$ 1
$\overline{RD}$ 2
$\overline{WR}$ 3
CLK IN 4
$\overline{INTR}$ 5
$V_{IN}(+)$ 6
$V_{IN}(-)$ 7
A GND 8
$V_{REF}/2$ 9
D GND 10
20 V_{CC} (OR V_{REF})
19 CLK R
18 DB0 (LSB)
17 DB1
16 DB2
15 DB3
14 DB4
13 DB5
12 DB6
11 DB7 (MSB)

The ADC0804 Analog-to-Digital Converter

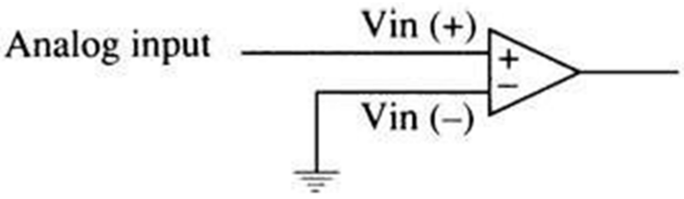
- To operate the converter, the WR pin is pulsed with CS grounded to start the conversion process.
- If a time delay is used that allows at least 100 μs of time, there is no need to test INTR pin.
- Another option is to connect the INTR pin to an interrupt input, so when the conversion is complete, an interrupt occurs.

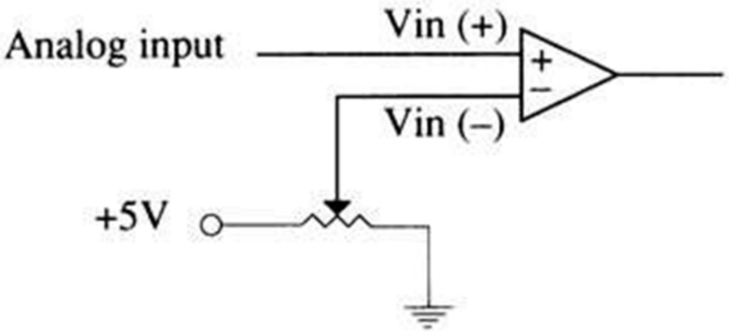
The timing diagram for the ADC0804 analog-to-digital converter



The Analog Input Signal

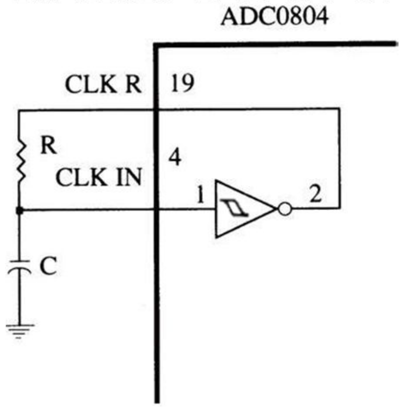
- Before ADC0804 has two analog inputs:
  - VIN(+) and VIN(–)
- These differential inputs are connected to an operational amplifier to produce a signal for the internal analog-to-digital converter.





Generating the Clock Signal

- ADC has an internal clock generator. We just need to connect a resistor and capacitor.
- ‘Or’
- It can be an external clock applied to CLK IN pin or can be generated with an RC circuit.
  - permissible range of clock frequencies is 100 KHz - 1460 KHz.
  - desirable to use a frequency as close as possible to 1460 KHz so conversion time is minimized
  - here **Fc= 1/(1.1 x RC)**
- If generated with an RC circuit. CLK IN and CLK R pins are connected to an RC circuit



Programming the ADC0804

Polling Method:

1. Make CS = 0 and send a low-to-high pulse to pin WR to
2. start conversion.
3. Keep monitoring the INTR pin using

   while(INTR==1);

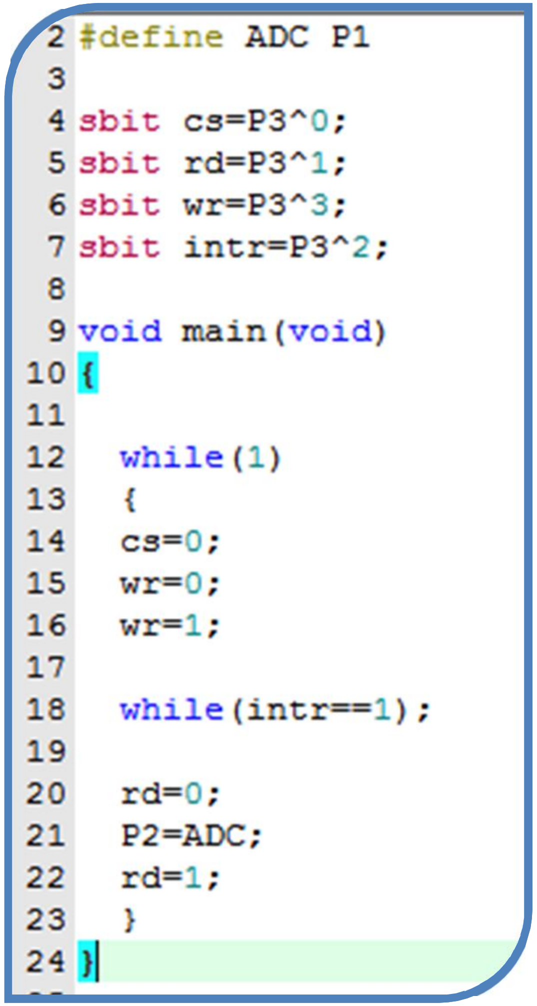
1. If INTR is low, the conversion is finished. If the INTR is high, keep polling until it goes low.
2. After the INTR has become low, we make CS = 0 and send a high-to-low pulse to the RD pin to get the data out of the
3. ADC804. Then send this to any of MC pins.

1. Interrupt Method:
2. Connect INTR pin of ADC to 0804 to INT0 or INT1 of 8051 use it as interrupt. Read the value of ADC in ISR.

Todays Task

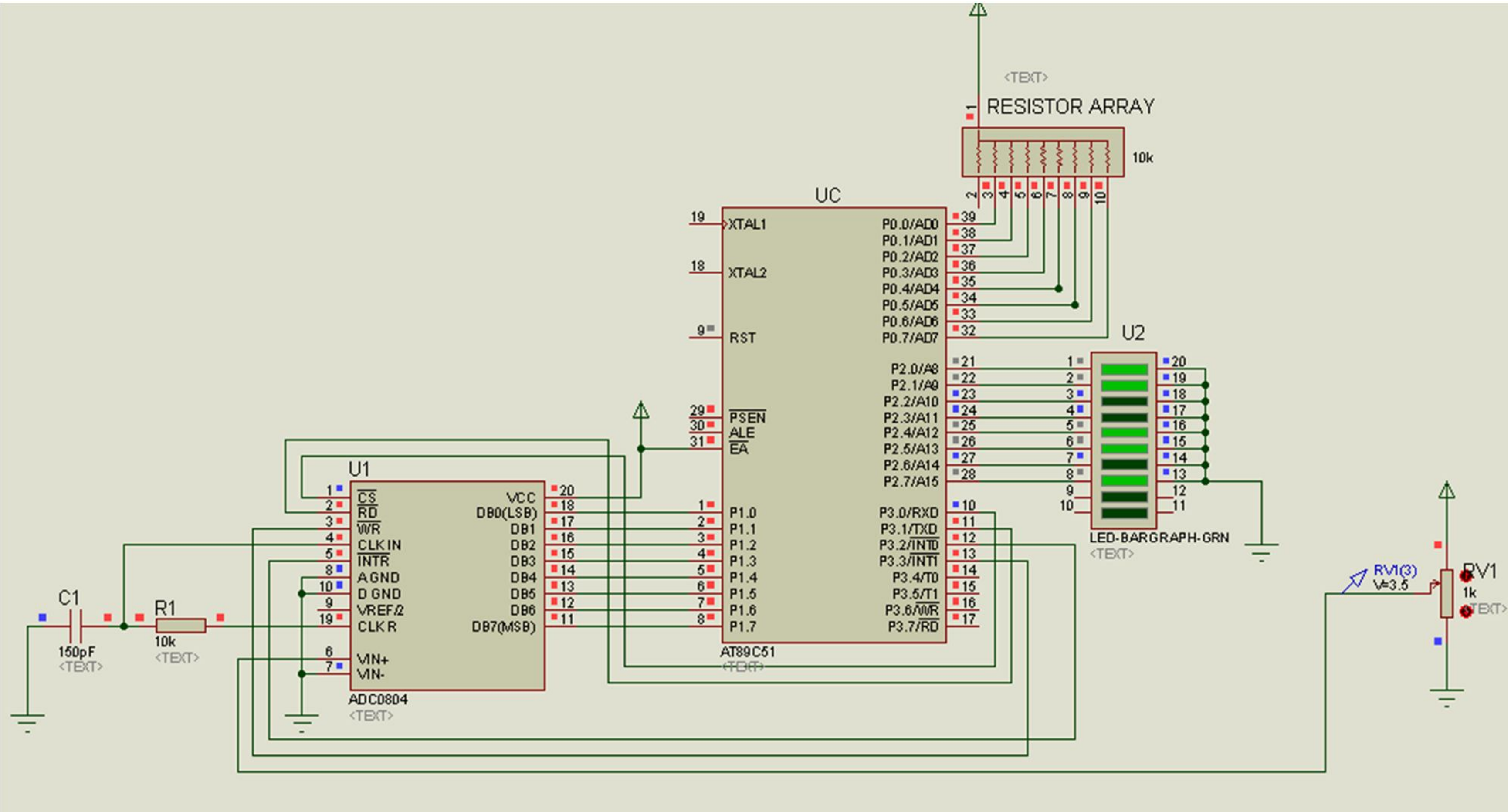
Read the Digital Output from an ADC and display its Value on Bar LEDs. Use a potentiometer to give an analog signal to ADC from 0-5V.

Task Code



```
#define ADC P1

sbit cs=P3^0;
sbit rd=P3^1;
sbit wr=P3^3;
sbit intr=P3^2;

void main(void)
{

  while(1)
  {
  cs=0;
  wr=0;
  wr=1;

  while(intr==1);

  rd=0;
  P2=ADC;
  rd=1;
  }
}
```

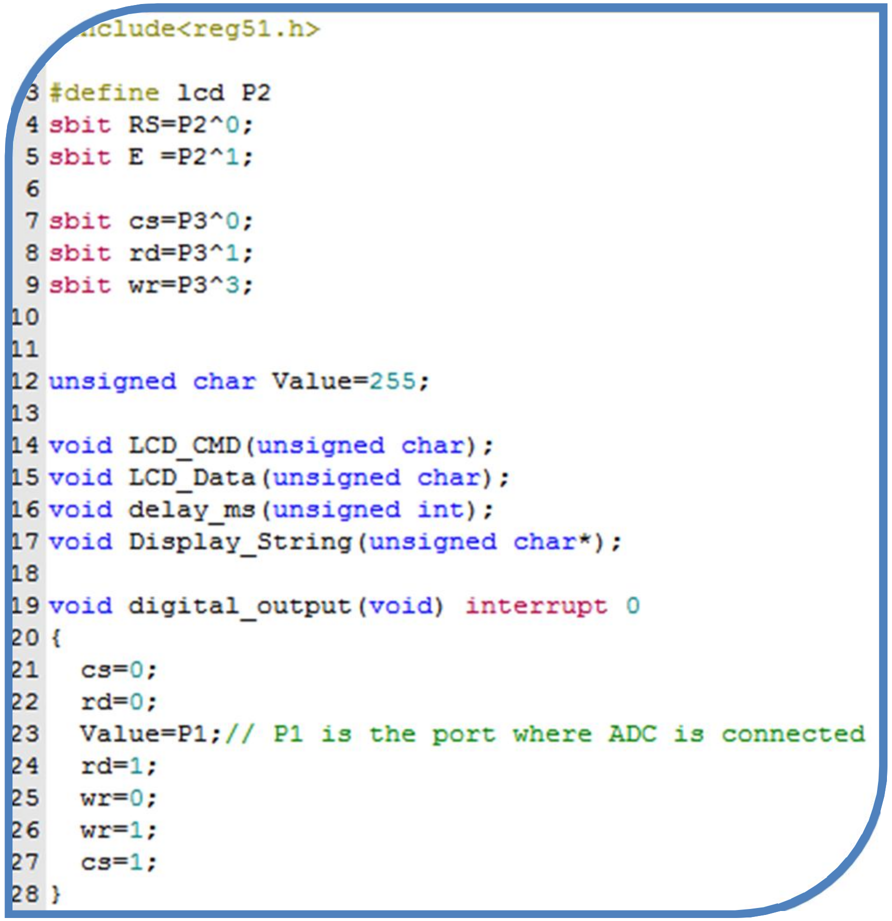
Proteus Simulation



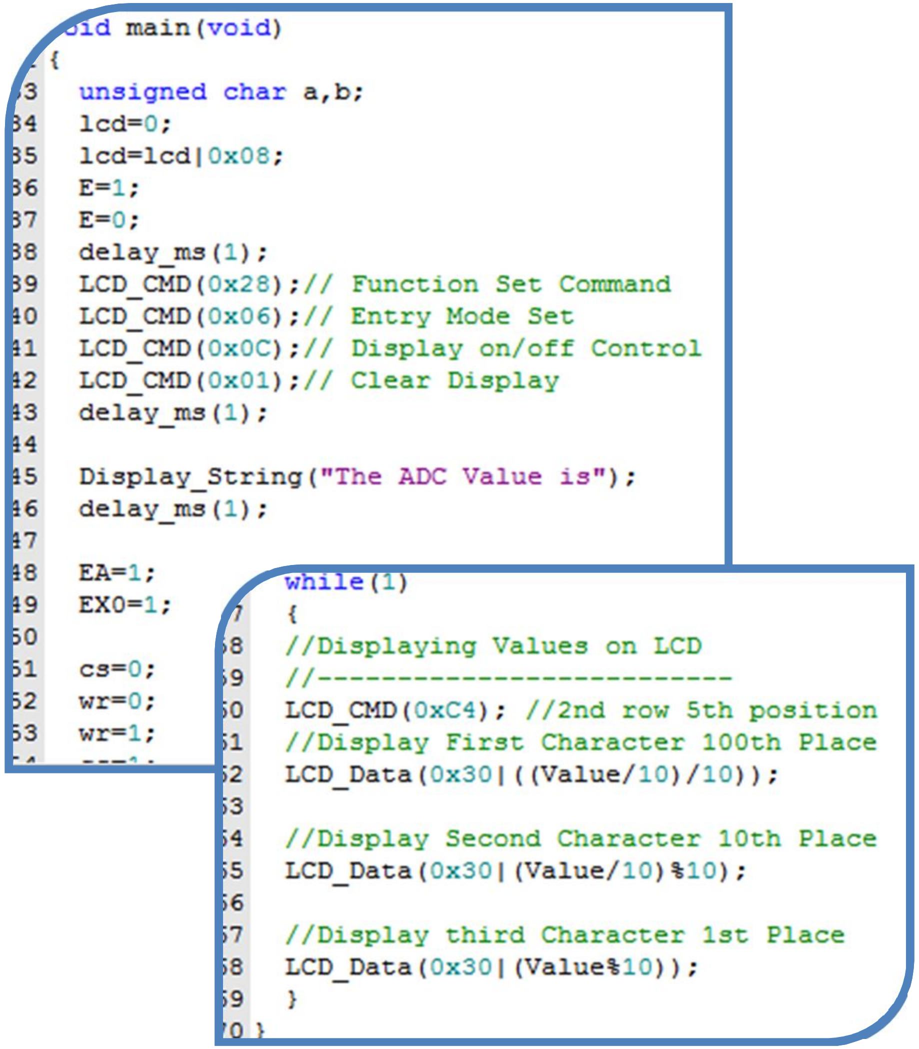
Todays Task2

Convert the digital value obtained in the last task to decimal and display it on LCD.

Task Code



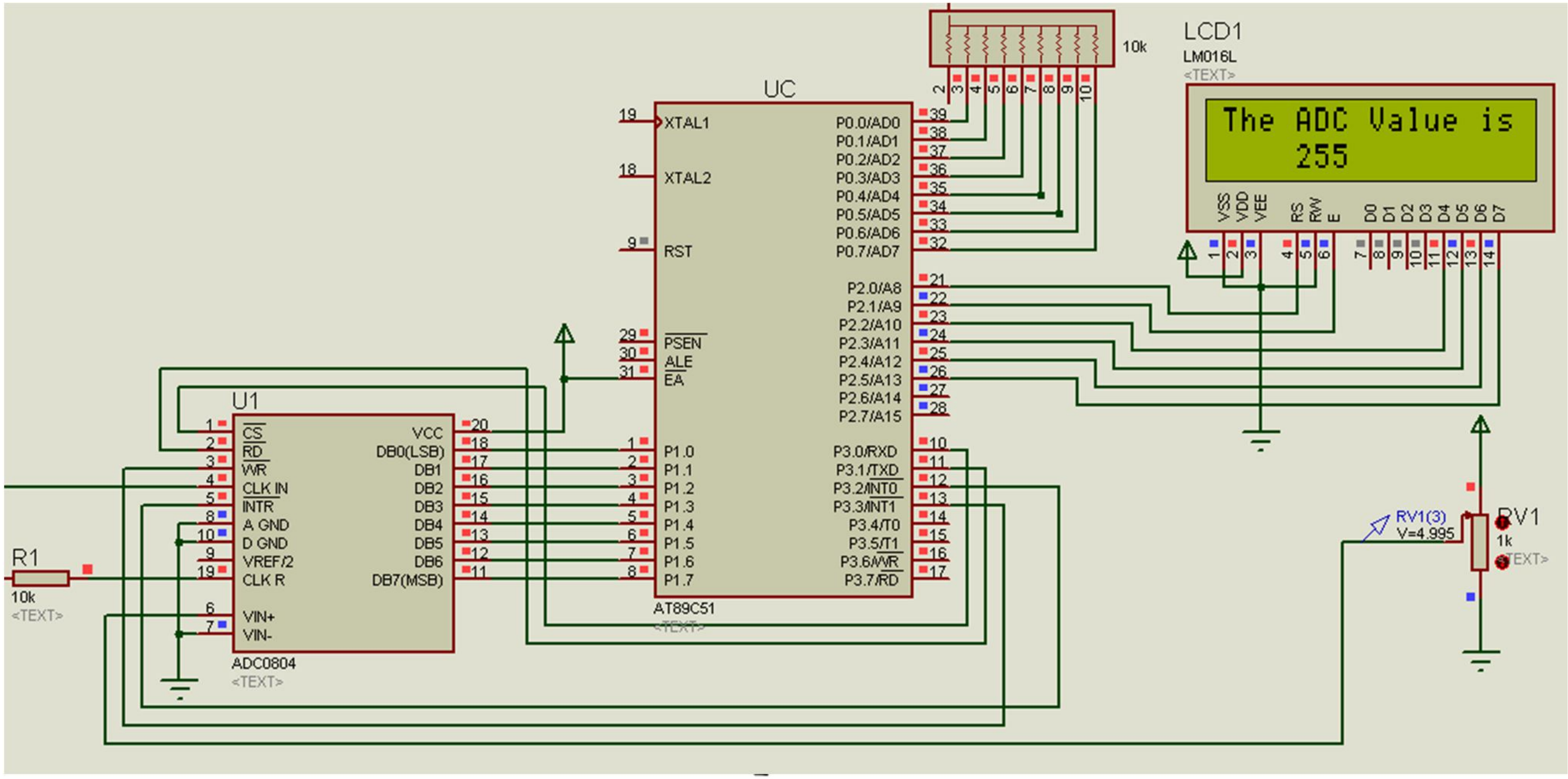
```
nclude<reg51.h>

#define lcd P2
sbit RS=P2^0;
sbit E =P2^1;

sbit cs=P3^0;
sbit rd=P3^1;
sbit wr=P3^3;


unsigned char Value=255;

void LCD_CMD(unsigned char);
void LCD_Data(unsigned char);
void delay_ms(unsigned int);
void Display_String(unsigned char*);

void digital_output(void) interrupt 0
{
  cs=0;
  rd=0;
  Value=P1;// P1 is the port where ADC is connected
  rd=1;
  wr=0;
  wr=1;
  cs=1;
}
```



```
oid main(void)
{
  unsigned char a,b;
  lcd=0;
  lcd=lcd|0x08;
  E=1;
  E=0;
  delay_ms(1);
  LCD_CMD(0x28);// Function Set Command
  LCD_CMD(0x06);// Entry Mode Set
  LCD_CMD(0x0C);// Display on/off Control
  LCD_CMD(0x01);// Clear Display
  delay_ms(1);

  Display_String("The ADC Value is");
  delay_ms(1);

  EA=1;
  EX0=1;

  cs=0;
  wr=0;
  wr=1;
```

```
  while(1)
  {
  //Displaying Values on LCD
  //---------------------------
  LCD_CMD(0xC4); //2nd row 5th position
  //Display First Character 100th Place
  LCD_Data(0x30|((Value/10)/10));

  //Display Second Character 10th Place
  LCD_Data(0x30|(Value/10)%10);

  //Display third Character 1st Place
  LCD_Data(0x30|(Value%10));
  }
}
```

Proteus Simulation



```
The ADC Value is
   255
```