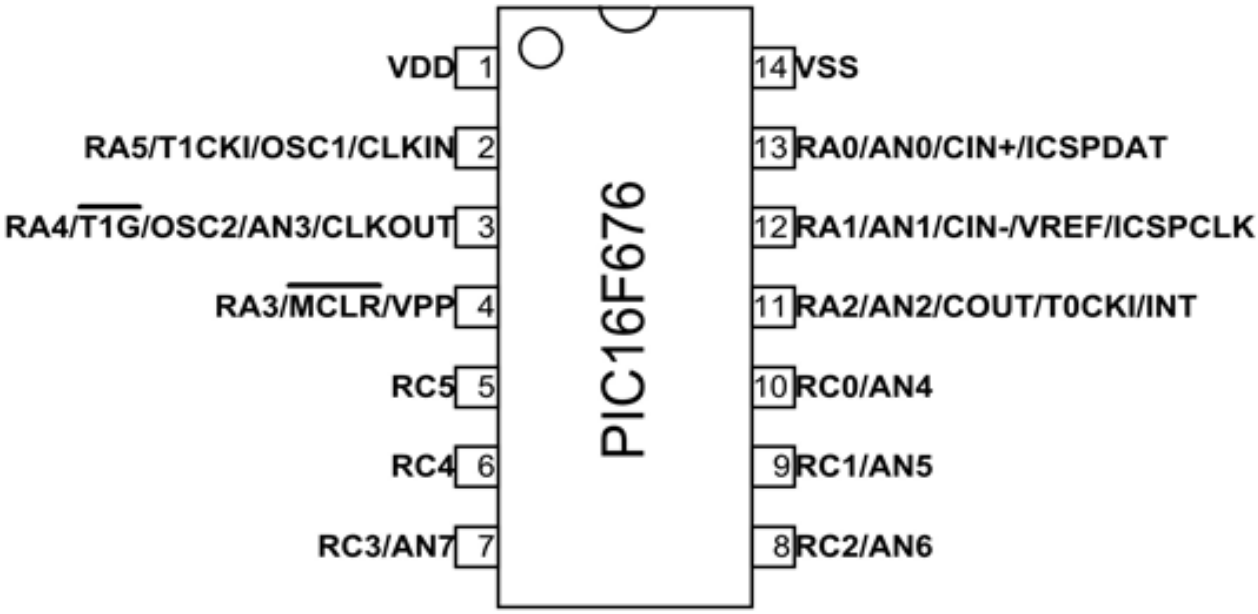
Name: Vikas Rajendra Sangannavar

Reg_no: 23026_642

Topic: Implementation of voltmeter ranges from 0 to 100 Vdc using Microcontroller.

Micro Controller: PIC16F676



Calculations for 100V → 5V:

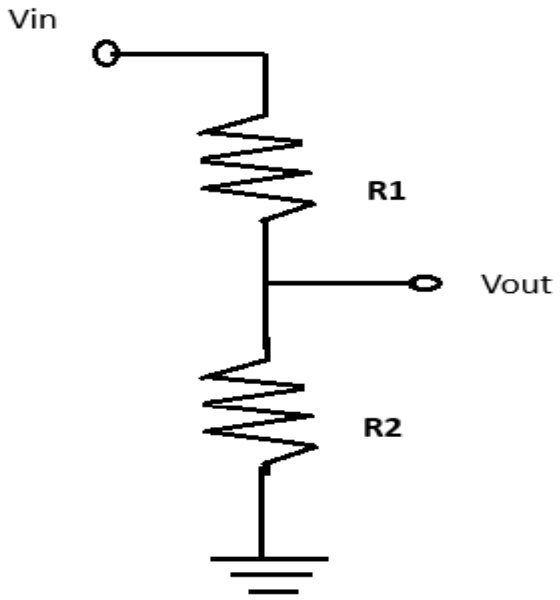


Fig 1.1 step down voltage

To reduce the voltage from 100 to 5 volts as the above fig 1.1 Vin = 100 volts as input and Vout = 5 volts as Output

Hence

Vout = (R1 / R1 + R2) x Vin

By sub the values

5 = (R1 / R1 + R2) x 100

5 = (100R1 / R1 + R2) ---- I

By solving equation I we get

R1 + R2 = 20R1

R2 = 20R1 – R1

R2 = 19R1 --- II

From the above equation we get to know that

R2 resistor have the value as 1K ohms

R1 resistor have the value as 19k ohms

Note : market we cant get the **19k resistor** as we are using simulation we can modify the resistor value as we want in the application.

Circuit Diagram:

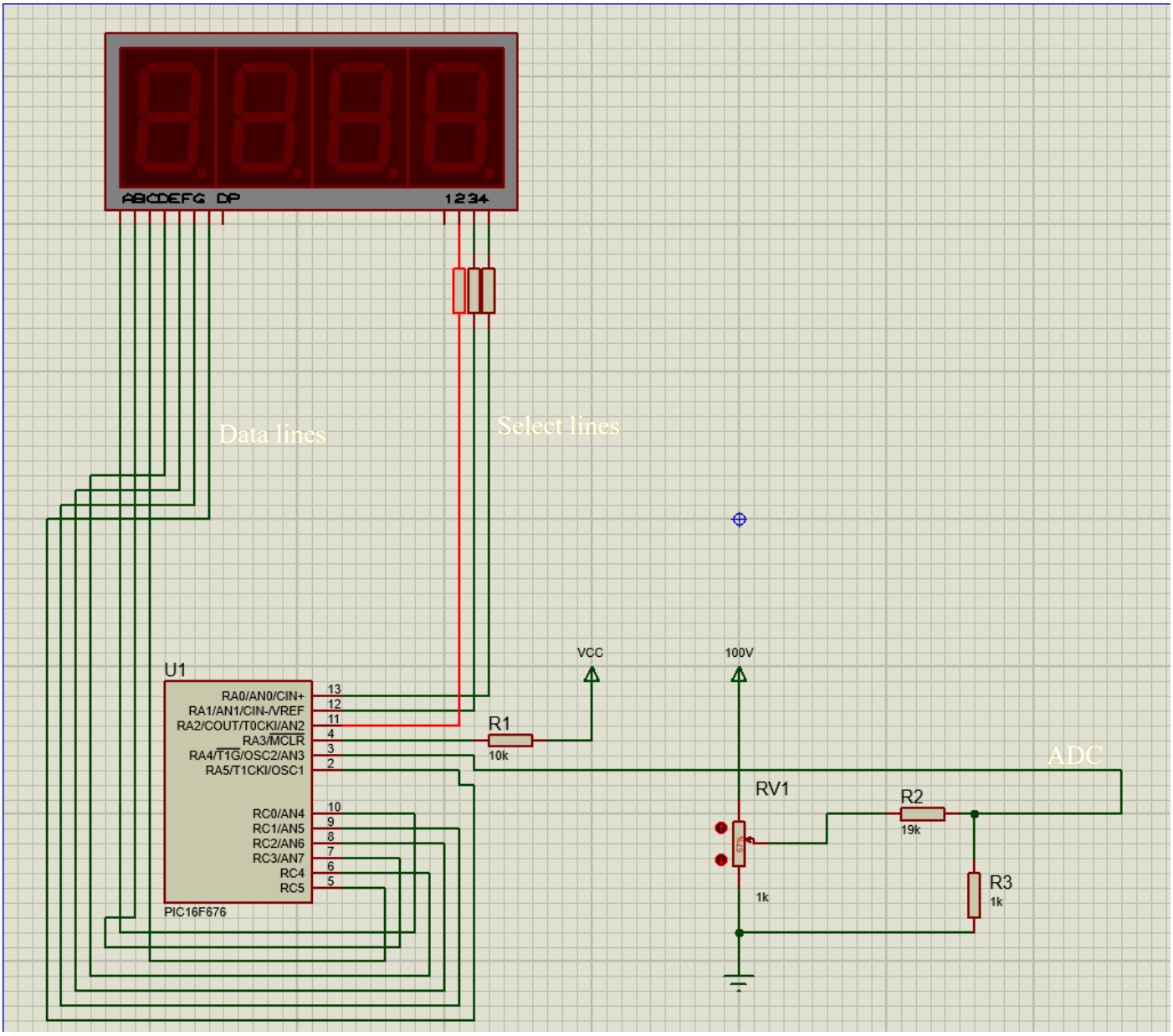


Fig 1.2 Circuit Diagram.

Flowchart:

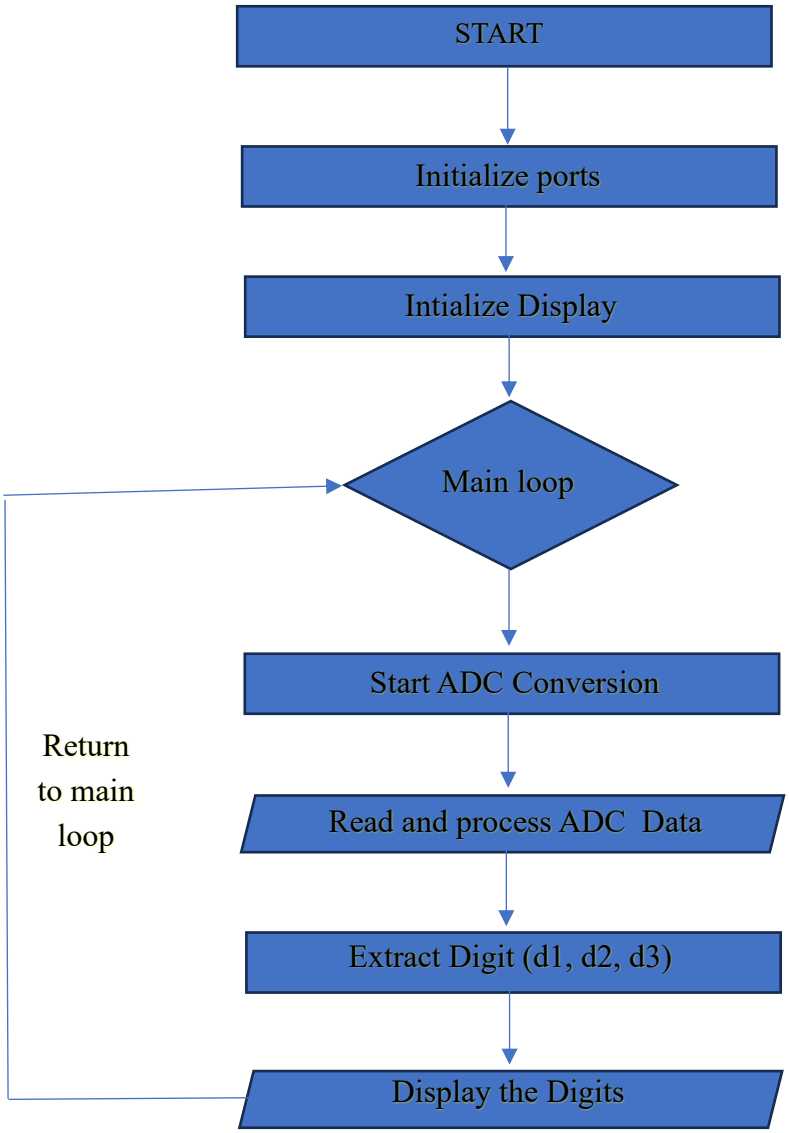


Fig 1.3 Flowchart.

CODE:

```
#include <xc.h>
```

- **These definitions map each digit (0 - 9) and a blank state to the corresponding segment code.**

```
#define DIGIT_0_Indx   0x40
#define DIGIT_1_Indx   0x79
#define DIGIT_2_Indx   0x24
#define DIGIT_3_Indx   0x30
#define DIGIT_4_Indx   0x19
#define DIGIT_5_Indx   0x12
#define DIGIT_6_Indx   0x02
#define DIGIT_7_Indx   0x78
#define DIGIT_8_Indx   0x00
#define DIGIT_9_Indx   0x10
#define BLANK_Indx     0x7F
```

- **An array 'SevenSegment' stores the segment codes for easy access using digit indices.**

```
uint8_t SevenSegment[] = { DIGIT_0_Indx, DIGIT_1_Indx, DIGIT_2_Indx,
        DIGIT_3_Indx, DIGIT_4_Indx, DIGIT_5_Indx,
        DIGIT_6_Indx, DIGIT_7_Indx, DIGIT_8_Indx,
        DIGIT_9_Indx };
```

- **Function Prototypes**

```
void Initialize_Display( void );
void Display_Value( uint8_t code );
```

- **This function sets up the microcontrollers comparator, analog select, ADC control register, and calls 'Intialize_Display' to set up the 7-segment display.**

```
void init_config()
{
    CMCON = 0x07;
    ANSEL = 0x08;
    TRISA |= 0x10; // RA4/AN3 as Input
    ADCON0 = 0x8D; // Right Justified, VDD, AN3 Selected
    ADCON1 = 0x10;
```

```
    Initialize_Display ();
}
```

- Enters an infinite loop where starts an ADC conversion, waits for the conversion to complete, reads the ADC result from ADRESH and ADRESL and process it, Extracts the Digits from the ADC result and Display each digit on the 7-segment display

```
void main()
{
   init_config(); //initialize the ports

  while(1)
  {
     int adc_data = 0;
     ADCON0 = 0x8D;
     ADCON0bits.GO_DONE = 1;  // Start Conversion
     while( ADCON0bits.GO_DONE ); // Wait Here
     adc_data = ADRESH<<8;
     adc_data |= ADRESL;
     adc_data = adc_data & 0x3FF;
     adc_data = adc_data/10;
     uint8_t d1, d2, d3;
     d1 = adc_data % 10;
     d2 = (adc_data/10) % 10;
     d3 = (adc_data/100) % 10;

     PORTA &= 0xF8;
     PORTA |= 0x04;
     Display_Value (SevenSegment[d1]);

     PORTA &= 0xF8;
     PORTA |= 0x02;
     Display_Value (SevenSegment[d2]);

     PORTA &= 0xF8;
```

```
      PORTA |= 0x01;
      Display_Value (SevenSegment[d3]);
   }
}
void Initialize_Display( void )
{
 TRISA &= 0xD8; // RA0,RA1,RA2, RA5 as Output
 TRISC = 0x00;
 PORTA &= 0xD8;
 PORTC = 0x00;
}
```

- This function takes a segment code and sets the appropriate pins on PORTC and PORTA to display the corresponding digit on 7 segment display.

```
void Display_Value( uint8_t code )
{
 /*
  * a -> RC0 (bit-0)
  * b -> RC3 (bit-1)
  * c -> RC5 (bit-2)
  * d -> RC4 (bit-3)
  * e -> RC2 (bit-4)
  * f -> RC1 (bit-5)
  * g -> RA5 (bit-6)
  */
 PORTC = 0;
 PORTA &= 0xDF;
 if ( code == BLANK_Indx )
 {
  PORTC = 0xFF;
  PORTA |= 0x20;
 }
 else
 {
```

```
  // RC0
  if( code & 0x01 )
   PORTC |= 0x01;
  // RC3
  if( code & 0x02 )
   PORTC |= 0x08;
  // RC5
  if( code & 0x04 )
   PORTC |= 0x20;
  // RC4
  if( code & 0x08 )
   PORTC |= 0x10;
  // RC2
  if( code & 0x10 )
   PORTC |= 0x04;
  // RC1
  if( code & 0x20 )
   PORTC |= 0x02;
  // RA5
  if( code & 0x40 )
   PORTA |= 0x20;
 }
}
```