AIM: To implement a Voltmeter using PIC 18F452

Tools: Keil uVision 4, ISIS Proteus 3

Theory: This experiment describes how to make a simple digital voltmeter (DVM) using a PIC 18F452 microcontroller. The range of this DVM is 0-5V, but we can easily increase or decrease the range of input voltage as our requirements using the voltage scaling method described. The PIC micro-controller reads the input voltage through one of the 13 analog channels and convert it to a 10-bit digital number using the internal ADC. Doing some math with ADC conversion this number can be converted to the actual measured voltage. The voltage is displayed in a LCD.

This powerful 10 MIPS (100 nanosecond instruction execution) yet easy-to-program (only 77 single word instructions) CMOS FLASH-based 8-bit microcontroller packs Microchip's powerful PIC architecture into an 40- or 44-pin package and is upwards compatible with the PIC16C5X, PIC12CXXX, PIC16CXX and PIC17CXX devices and thus providing a seamless migration path of software code to higher levels of hardware integration. The PIC18F452 features a 'C' compiler friendly development environment, 256 bytes of EEPROM, Self-programming, an ICD, 2 capture/compare/PWM functions, 8 channels of 10-bit Analog-to-Digital (A/D) converter, the synchronous serial port can be configured as either 3-wire Serial Peripheral Interface (SPI™) or the 2-wire Inter-Integrated Circuit (I²C™) bus and Addressable Universal Asynchronous Receiver Transmitter (AUSART). All of these features make it ideal for manufacturing equipment, instrumentation and monitoring, data acquisition, power conditioning, environmental monitoring, telecom and consumer audio/video applications.

Algorithm :

1) Feed the input to RA0 through a pot or other network.
2) We can use voltage scaling technique to widen the range of measurement.
3) PIC 18 converts this analog signal into 10 bit digital output
4) Corresponding output is displayed on the LCD.
5) We can reset the system by pressing reset button.

Code: // **C Program**

```
#include <p18f452.h>
void delay(void);
void init_lcd(void);
void lcd_ready(void);
void put_nibble(void);
#define lcd_port PORTB
#define lcdbusy PORTBbits.RB7
#define  lcdrs PORTCbits.RC2
#define  lcdrw PORTCbits.RC1
#define  lcden PORTCbits.RC0
unsigned char rdbuf1,rdbuf2;
unsigned char dptr,a,b,digits[7];
long int dvsor[]={0x186a0,0x2710,0x3e8,0x64,0x0a};
long int adcbuf,temp;

unsigned char lcd_cmd[]={0x28,0x01,0x06,0x0E,0x82};
unsigned char msg1[]={"VOLTMETER:$"};
void delay(void)
                {
                        unsigned char i,j;
                         for(i=0;i<255;i++)
                          for (j=0;j<255;j++);
                }
void init_lcd(void)
        {
                delay();
                lcdrs=0;
                lcdrw=0;
                lcden=1;
                Nop();
                lcden=0;
        }
void lcd_ready(void)
        {
                lcd_port=0xff;
                lcdrw=1;
wait:  lcden=1;
                Nop();
                rdbuf2=lcd_port;
                lcden=0;
                Nop();
```

```
                lcden=1;
                Nop();
                rdbuf1=lcd_port;
                lcden=0;
                Nop();
                rdbuf2 &= 0x80;
                if (rdbuf2==0x80) goto wait;
                return;
        }
void put_nibble(void)
        {

                lcdrw=0;
                lcd_port=rdbuf1;
                lcden=1;
                Nop();
                lcden=0;
                rdbuf1=rdbuf1<<4;
                lcd_port=rdbuf1;
                lcden=1;
                Nop();
                lcden=0;
                return;
        }
void main(void)
{

        TRISB=0;
        TRISC=0;
        lcden=0;
        lcd_port=0x30;
        init_lcd();
        lcd_port=0x30;
        init_lcd();
        lcd_port=0x20;
        init_lcd();
        lcdrs=0;
        for (a=0;a<5;a++)
        {
                lcd_ready();
                rdbuf1=lcd_cmd[a];
                put_nibble();
        }

        dptr=0;
        do
        {
```

```
                lcdrs=0;
                lcd_ready();
                lcdrs=1;
                rdbuf1=msg1[dptr];
                put_nibble();
                dptr++;
        }while(msg1[dptr]!='$');
        ADCON0=0X81;
        ADCON1=0X80;
        while(1)
        {
                for(a=0;a<4;a++) delay();
                digits[6]=' ';
                lcdrs=0;
                rdbuf1=0xc2;
                put_nibble();
                ADCON0 |=4;
wait4done:      a=ADCON0&4;
                if(a==4)goto wait4done;
                adcbuf=0;
                adcbuf=(long int)ADRESH<<8;
                adcbuf |=(long int)ADRESL;
                adcbuf *=(long int)488.8;
                for(dptr=0;dptr<5;dptr++)
                {
                        temp=adcbuf/dvsor[dptr];
                        digits[dptr]=(unsigned char)temp;
                        digits[dptr]+=0x30;
                        adcbuf%=dvsor[dptr];
                }
                digits[dptr]=(unsigned char)adcbuf+0x30;
                for(dptr=0;dptr<7;dptr++)
                {
                        lcdrs=0;
                        lcd_ready();
                        lcdrs=1;
                        if(dptr==1)
                        {
                                rdbuf1='.';
                                put_nibble();
                        }
                        rdbuf1=digits[dptr];
                        put_nibble();
                }
                lcdrs=0;
                lcd_ready();
                lcdrs=1;
```

```
                rdbuf1='V';
                put_nibble();
            }
        }
```

Conclusion: Thus, a Voltmeter using PIC 18F452 was simulated using Proteus with the help of Keil. The results were visually verified using the Run feature in Proteus.