

1. INTRODUCTION

This report describes hardware and software design for a Digital Voltage Meter (DVM). The design of DVM is based on PIC microcontroller, specifically

- The input voltage is from 0V to 30V.
- DVM should automatically select one of these ranges: 0V-10V, 10V-20V, 20V-30V.
- The measured voltage should be displayed on character LCD.
- The measured voltage should also be displayed on PC using USART at 9600bps.
- DVM should display a warning message if the input voltage is above 30V.

2. HARDWARE DESIGN

The block diagram of DVM is shown below

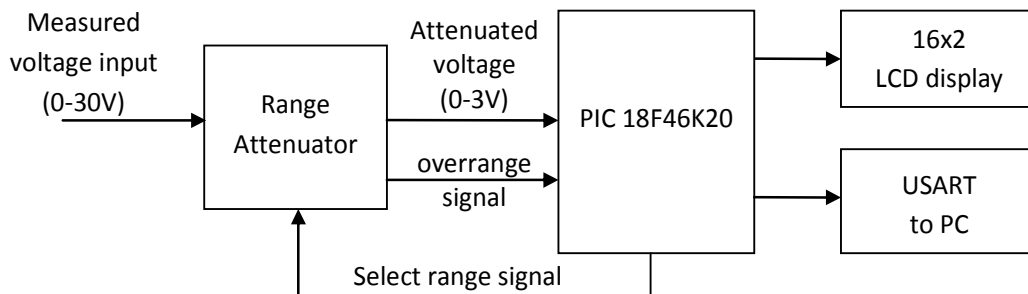


Figure 1 Block diagram of DVM

We are going to use PIC18F46K20 as the controller of DVM. Whilst the input is from 0V-30V, the maximum ADC input voltage of the microcontroller is only 3.3V. So we need a circuit to scale the input voltage to 0V-3.3V range (we will use 3V as maximum scaled input voltage to calculate resistor values). Figure 2 is the schematic of analog switch and resistor bridges of three ranges to scale the input voltage. Resistor values can be calculated as following

Range 0:

$$\frac{R_4}{R_4 + 100K} = \frac{3V}{9.999V} \rightarrow R_4 = 43K$$

Range 1:

$$\frac{R_2}{R_2 + 100K} = \frac{3V}{19.999V} \rightarrow R_2 = 17.6K$$

Range 2:

$$\frac{R_2}{R_2+100K} = \frac{3V}{29.999V} \rightarrow R_2 = 11.1K$$

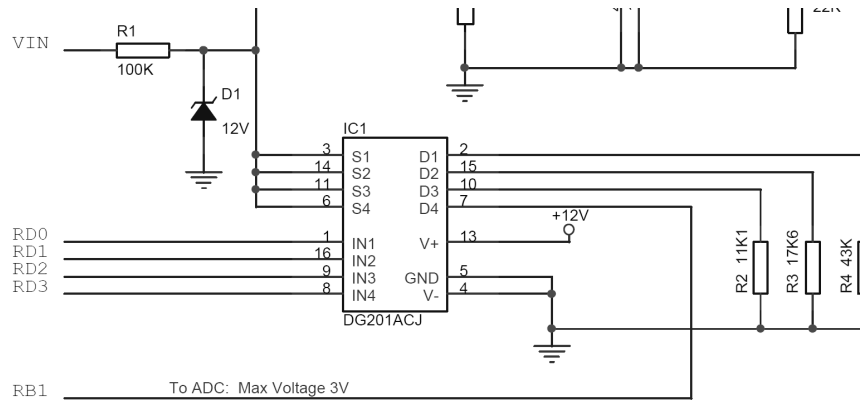


Figure 2 Attenuator

DG201 is a CMOS Quad SPST Analog Switch chip. Because the maximum input of DG201 should be kept lower than $28V_{p-p}$ (with $\pm 15V$ power supply) [1] and our input voltage is lower than 10V, a 12V zener diode is used at the V_{IN} . Switches are controlled by $IN0 - IN3$ inputs. Ranges will be selected by the microcontroller using $RD0 - RD3$ pins.

To help the microcontroller determines which range is correct, a comparator should be used to monitor the input voltage. The circuit is shown in Figure 3 below.

LM311 is a voltage comparator [2]. If v^+ (pin 2) is larger than v^- (pin 3) the output of LM311 goes high (+5V). If v^+ is smaller than v^- the output will be low (GND). At the output of the comparator (pin 7) is a bridge circuit (22K, 10K). This circuit will keep the output at 3.3V when the output goes high

$$\frac{22K}{22K+10K} \times 5V = 3.4V \text{ (Approximately high level of the microcontroller)}$$

The scaled input voltage is connected to v^+ and v^- is connected to a variable resistor R_5 . To ensure the scaled input voltage is not larger than 3V (as shown above), we must adjust R_5 at v^- pin at 3V.

There is a mistake in the schematic which is given by Mr. Bell (the output of the comparator is connected to $RD4$ which is connected to LED already). Instead, the output of the comparator should be connected to $RB0$ pin. The microcontroller will continuously read this pin. If $RB0$ goes high, it means the input voltage is above current range. The microcontroller should select the next range. If current range is the last range, the microcontroller should display a warning message.

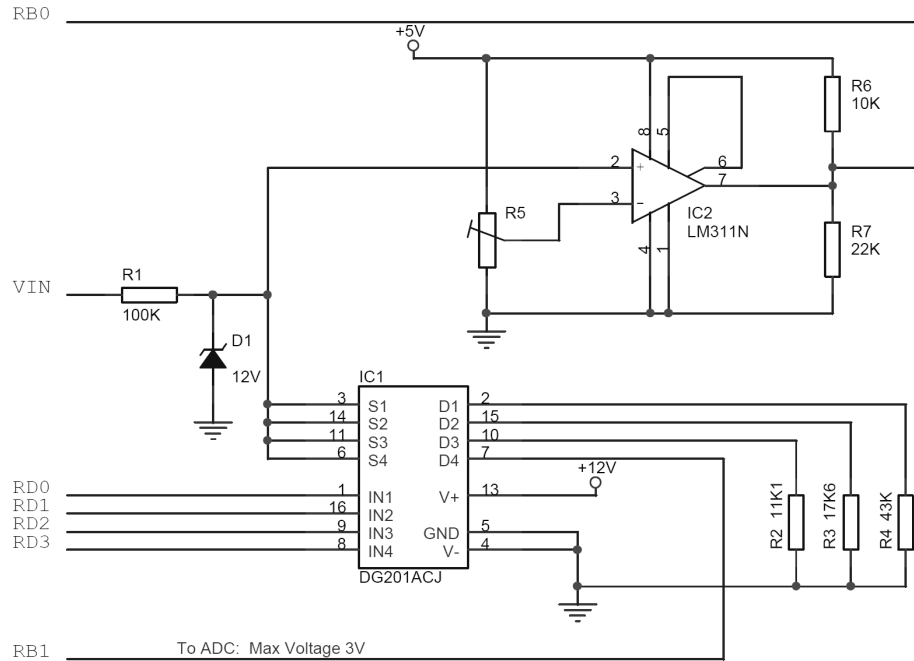


Figure 3 Attenuator and comparator

The third part of the schematic is LCD display. Because the difference of voltages between LCD and the microcontroller [3], we use 4504N chip to convert between 3.3V and 5V. To reduce the number of pins, LCD will be configured in 4-bit mode [3]. Therefore, we use DB4-DB7 as data bus; E is enable signal and RS is register select signal. There is another mistake in the schematic which is given by Mr. Bell. We should use RA0-RA3 (not port E) as data bus of LCD as shown below

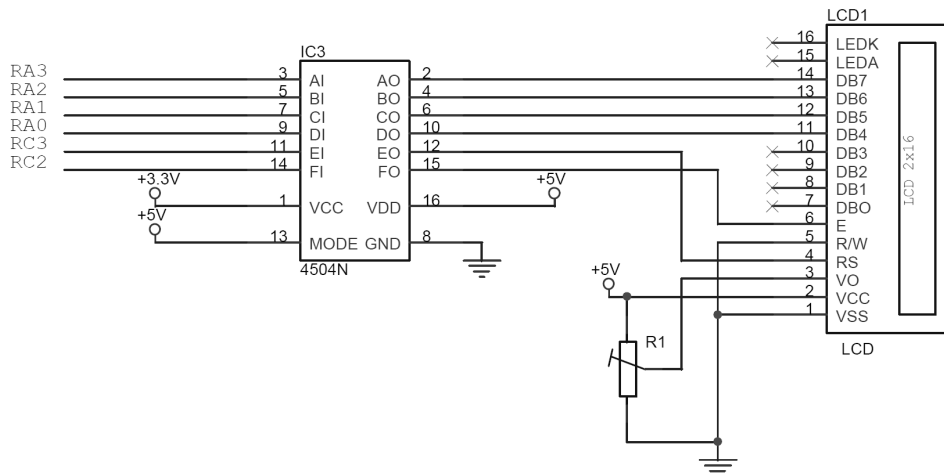


Figure 4 LCD Interfacing

The variable resistor R_8 is used to adjust LED backlight. The R/W signal is constantly connected to GND, so LCD would not be read. We use RC3 and RC2 to control E and RS signals of LCD.

The fourth part of the schematic is PC interface using RS232. This is a popular and simple way to exchange data between microcontrollers and PC [4]. Because the difference of standards, we will use MAX232 [6] to convert between LVTTTL [5] and RS232 [4]. The schematic is shown below

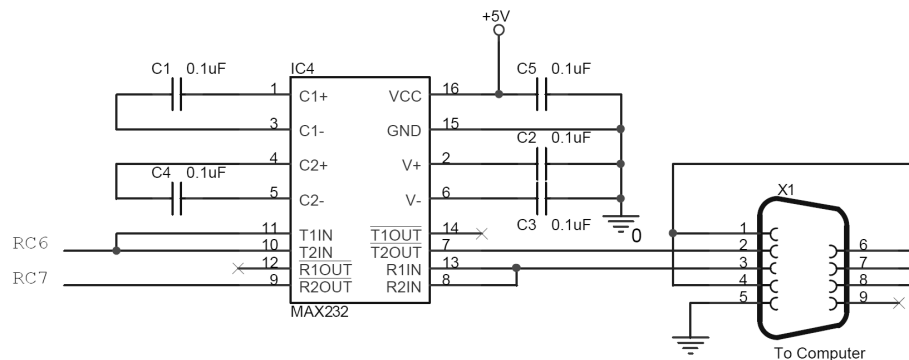


Figure 5 RS232 Interfacing

Serial interface only use two pins, one for transmitting and one for receiving. These pins are connected to USART pins (RC6 and RC7) on the microcontrollers [5].

3. SOFTWARE DESIGN

MPLAB is a development tool suite for PIC microcontrollers. We are using PIC18F46K20 for the project. This is an 8-bit microcontroller with nanoWatt technology. PIC18F46K20 includes 64KB Flash for program memory, 3936 bytes SRAM and 1024 bytes EEPROM for data memory, 14 10-bit ADC channels and one EUSART [5].

Below are the steps to control DVM

- Initialize port pins, ADC, LCD
- Disable ADC input by the main switch, select range 0
- Using a loop
 - If it is over range, select the next range
 - If it is in range, convert analog input to digital, translate binary value into characters and display on LCD
 - If voltage value is lower than current range, decrease range by one

First, we should initialize for peripherals. To do that, we look at RD3-RD0 pins (used for ranges) and RB0 pin (used for over range). Because these pins are bidirectional, we must configure them first using TRISD and TRISB registers.

```

TRISD = 0; // all are outputs
TRISBbits.TRISB0 = 1; // RB0 is input
LATDbits.LATD0 = 0; LATDbits.LATD1 = 1; LATDbits.LATD2 = 1;
LATDbits.LATD3 = 1; // select range 0, disable main switch
    
```

Configure LCD can take many steps

- Set pins' direction: RA3-RA0, RC3-RC2 are outputs

```
TRISC = 0; LATC = 0;
TRISA = 0; LATA = 0;
```

- Set control signals

```
LCD_RS = 0; // select control register
LCD_E = 0; // disable
```

- Reset LCD and select 4-bit mode: wait for 15ms, write 0x03 to control register, wait for 5ms, write 0x03 to control register, wait for 1ms, write 0x03 to control register, wait for 1ms, write 0x02 to control register for 4-bit mode.
- Initialize LCD:

```
lcd_write(0x28); // function set instruction [3]
// 4-bit, 2-line, 5x8 dot
lcd_write(0x0F); // display on/off control instruction [3]
// on, display cursor, blink
lcd_clear();
lcd_write(0x06); // entry mode set instruction [3]
// increment, shift right
```

We will know how to control LCD later. Below is how to configure ADC, note that we are using AN10 (RB1) as analog input, so we must set RB1 as an input pin, turn on AN10 by ANSELH and ADCON0 registers. Then, ADC can be configured by using ADCON0 and ADCON1 registers [5]

```
ANSEL = 0; ANSELH = 0; // turn off all analog inputs
ANSELHbits.ANS10 = 1; // select AN10
ADCON1 = 0; // voltage references are VDD and VSS
ADCON2 = 0b00111000; // left justified, 20 TAD, Fosc/2
ADCON0 = 0b00101001; // select AN10 and turn on ADC
```

Now, we will use a loop to select correct range and start a conversion. The pseudo code can be shown below

```
while (1)
{
    if (in range)
    {
        enable_main_switch;
        start_ADC_conversion;
        convert_binary_to_string;
        display_LCD;
    }
    else if (over range)
    {
        disable_main_switch;
        if (range is 3) display_warning_message;
        else increase_range;
    }
}
```

```
    }  
    else if (below range) decrease_range;  
}
```

To enable or disable the main switch, we use following instruction

```
LATDbits.LATD3 = 0;    // enable the main switch  
LATDbits.LATD3 = 1;    // disable the main switch
```

In the program, we call ADC_Convert() to start a conversion. This function will do following steps:

- Clear GO_DONE bit in ADCON0 register to start a conversion.
- When the ADC conversion is completed, GO_DONE will go high. So we should wait for this bit.
- ADC value is in ADRESH register. We should note that the result is 10-bit. However, we only use 8 MSB bits. These are in ADRESH. Two remain LSB bits are in ADRESL [5].

```
unsigned char ADC_Convert(void)  
{  
    ADCON0bits.GO_DONE = 1;           // start conversion  
    while (ADCON0bits.GO_DONE == 1); // wait for it to complete  
    return ADRESH;                    // return high byte of result  
}
```

To display the value on LCD, we should convert the result to ASCII characters first. Because we are using 3.3V and GND as references and the maximum input voltage is set at 3V, we can convert ADC conversion result into displayed voltage value by using the following equation

$$\text{displayed value} = \frac{(\text{ADC result}) \times 1000 \times (\text{current range} + 1)}{231}$$

In this equation, the result is divided by 231 because this is the ADC conversion result if the input voltage is 3V ($\frac{3V}{3.3V} \times 255 = 231.8$). Because the input voltage is scaled from 10V, we should multiple $1000 \times (\text{current range} + 1)$. We use 1000 to increase the resolution of the result.

Now the displayed value is the binary-based input voltage. We will convert this into ASCII characters by dividing 1000, 100, 10 respectively. Below is code to do that

```
thousand = vol/1000; thousand += 0x30;  
vol = vol%1000;  
hundred = vol/100; hundred += 0x30;  
vol = vol%100;  
ten = vol/10; ten += 0x30;  
unit = vol%10; unit += 0x30;
```

The vol variable is the displayed value in binary. The thousand, hundred, ten and unit are ASCII characters will be used to display on LCD. We also can use vol variable to select a correct range by comparing it with 1000, 2000 which are 10V, 20V respectively.

```
if ((vol < 1000) && (CurrentRange != 0))
    select_range_0(); // <10V and current_range is not 0
else if ((vol > 1000) && (vol < 2000) && (CurrentRange != 1))
    select_range_1(); // 10V-20V and current range is not 1
else select_range_2(); // >20V, select range 2
```

Finally, we can display voltage on LCD by calling lcd_putch() function as below

```
lcd_putch(thousand);
lcd_putch(hundred);
lcd_putch('.');
lcd_putch(ten);
lcd_putch(unit);
```

4. RESULTS

Below is the circuit and result displayed on LCD

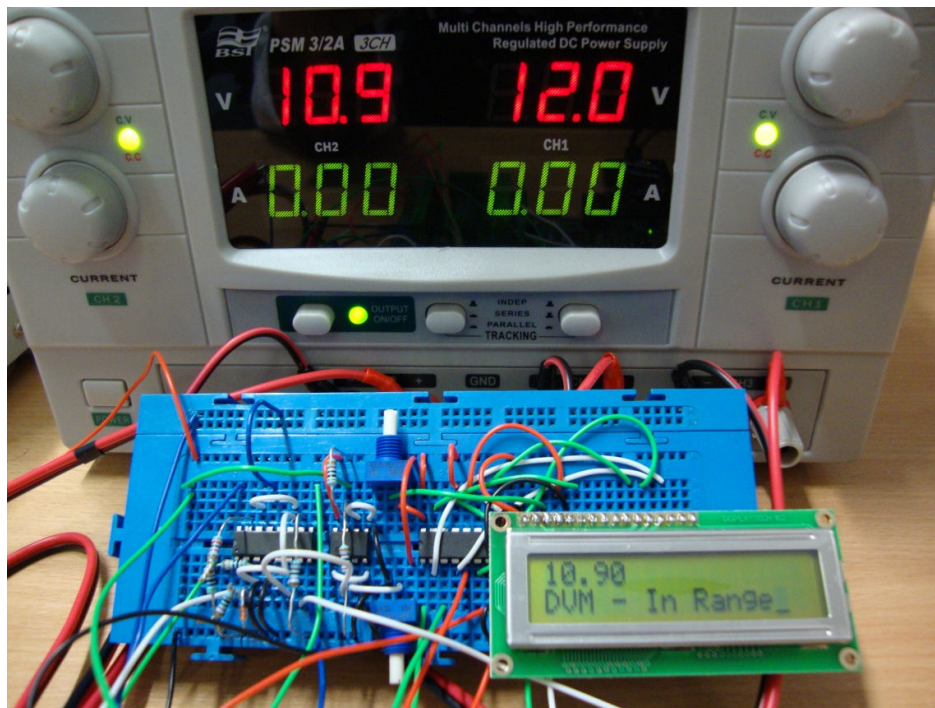


Figure 6 Results

5. REFERENCES

- [1] DG201 datasheet, Harris Semiconductor
- [2] LM311 datasheet, National

[3] HD44780U (LCD) datasheet, Hitachi

[4] Networking and Internetworking with Microcontrollers, Fred Eady, Newnes, 2004

[5] PIC18F46K20 datasheet, Microchip

[6] MAX232 datasheet, Texas Instruments