Title: Measure 0 to 100Vdc using micro controller

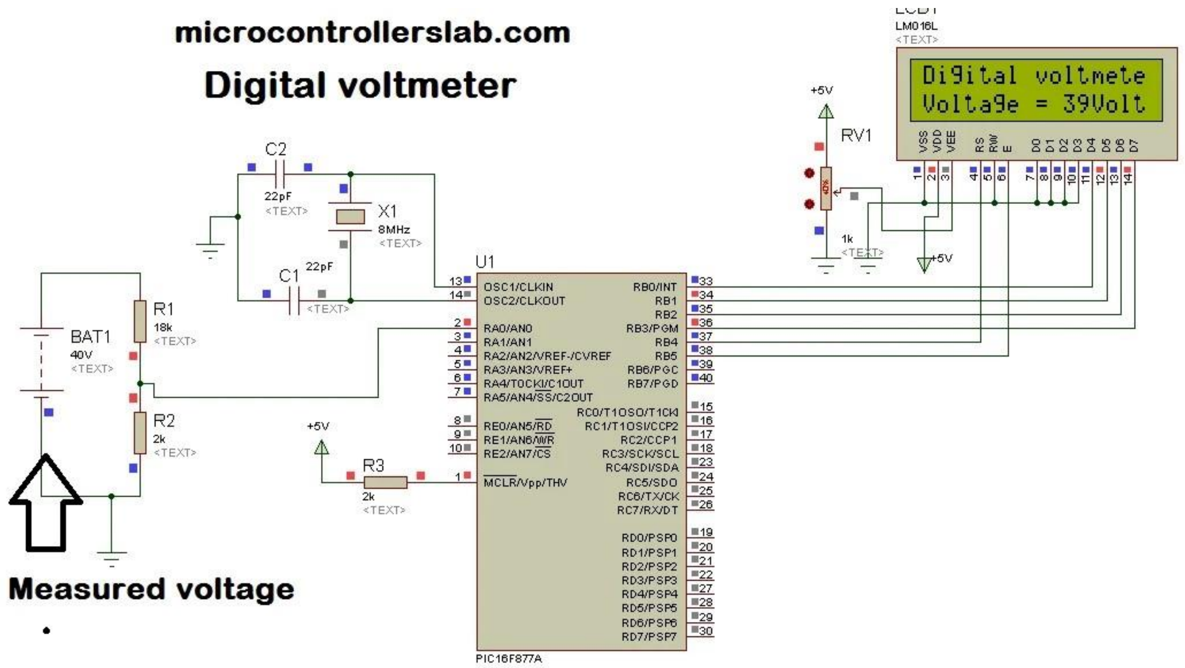
DC Voltmeter Introduction

Digital DC voltmeter is designed to measure DC voltage using the PIC16F877A microcontroller. A voltage divider circuit is used to divide voltage into two parts. To prevent more than 5 volts appearing across the pic microcontroller. Because the microcontroller can not read voltage more than 5 volts directly.

In this project, we used two types of displays namely 16×2 LCD and 4-digit seven-segment display. In the first first section, we will see how to display a value on LCD and in the second section, we will see how to display measured voltage value on a 4-digit seven-segment display.

Digital Voltmeter with LCD display Circuit Diagram

Circuit diagram of digital voltmeter using pic microcontroller and 16×2 LCD is given below. A 40-volt battery is used as a voltage source whose voltage you want to measure. PIC16F877A microcontroller can not directly read 40 volts. The voltage divider circuit using a resistor is used to **step down dc voltage** appearing across analog to digital converter pin of PIC16F877A microcontroller.



Title: Measure 0 to 100Vdc using micro controller

To measure 0 to 100V DC using a microcontroller like the ESP32, you can use a voltage divider circuit to step down the voltage to a range that the microcontroller's ADC (Analog-to-Digital Converter) can handle. The ESP32's ADC typically operates at a maximum input voltage of 3.3V.

Here's a step-by-step guide on how to do this:

1. **Design the Voltage Divider:**
   - A voltage divider uses two resistors to scale down the input voltage to a lower voltage. The voltage divider formula is: Vout=Vin×R2R1+R2V_{out} = V_{in} \times \frac{R2}{R1 + R2}Vout=Vin×R1+R2R2
   - For example, to scale down 100V to 3.3V: 3.3V=100V×R2R1+R23.3V = 100V \times \frac{R2}{R1 + R2}3.3V=100V×R1+R2R2 Solving for R2R1+R2\frac{R2}{R1 + R2}R1+R2R2: R2R1+R2=3.3100=0.033\frac{R2}{R1 + R2} = \frac{3.3}{100} = 0.033R1+R2R2=1003.3=0.033
2. **Choose Resistor Values:**
   - Let's choose R2=3.3kΩR2 = 3.3k\OmegaR2=3.3kΩ. Then, 3.3kΩR1+3.3kΩ=0.033\frac{3.3k\Omega}{R1 + 3.3k\Omega} = 0.033R1+3.3kΩ3.3kΩ =0.033 Solving for R1R1R1: R1=3.3kΩ0.033−3.3kΩ≈96.7kΩR1 = \frac{3.3k\Omega}{0.033} - 3.3k\Omega \approx 96.7k\OmegaR1=0.0333.3kΩ −3.3kΩ≈96.7kΩ
   - You can choose standard resistor values close to these calculated values. For instance, R1=100kΩR1 = 100k\OmegaR1=100kΩ and R2=3.3kΩR2 = 3.3k\OmegaR2=3.3kΩ.
3. **Connect the Voltage Divider:**
   - Connect the high voltage (0 to 100V) to the input of the voltage divider.
   - Connect the junction of the two resistors to an ADC pin of the ESP32.
   - Ensure that the ground of the high voltage source is common with the ESP32 ground.
4. **Protect the Microcontroller:**
   - To protect the ADC pin from over-voltage, you can add a Zener diode (e.g., 3.3V) across the ADC input and ground.
   - You can also add a small capacitor (e.g., 100nF) across the ADC input to filter out noise.
5. **Measure and Calculate the Voltage:**
   - In your ESP32 code, read the ADC value and convert it back to the original voltage:

     ```
     cpp
     int adcValue = analogRead(ADC_PIN);
     float voltage = (adcValue / 4095.0) * 3.3; // Assuming 12-bit
     ADC resolution
     float inputVoltage = voltage * ((R1 + R2) / R2);
     ```

   - Adjust the calculation if your resistor values are different.

Title: Measure 0 to 100Vdc using micro controller

Here's a schematic representation:

```
 High Voltage (0-100V)

        |
      R1 (100kΩ)
        |
----> ADC Pin (ESP32)
        |
      R2 (3.3kΩ)
        |
      GND
```

Note:

- Ensure resistors have a high enough power rating to handle the voltage and current.
- Safety is crucial when working with high voltages. Make sure to take necessary precautions and follow best practices.

This method will allow you to safely measure a high voltage range using your microcontroller.

implement same using pic controller

Implementing the voltage measurement using a PIC microcontroller involves similar steps to those outlined for the ESP32. Here's how you can adapt the process:

Step-by-Step Implementation for PIC Microcontroller:

1. **Choose a PIC Microcontroller:**
   - Select a PIC microcontroller with an ADC module. For example, PIC16F877A.
2. **Design the Voltage Divider:**
   - Follow the same calculation for the voltage divider as before:
     - Choose R1=100kΩR1 = 100k\OmegaR1=100kΩ and R2=3.3kΩR2 = 3.3k\OmegaR2=3.3kΩ.
3. **Connect the Voltage Divider:**
   - Connect the high voltage (0 to 100V) to the input of the voltage divider.
   - Connect the junction of the two resistors to an ADC pin of the PIC.
   - Ensure the ground of the high voltage source is common with the PIC ground.
4. **Protect the Microcontroller:**
   - Use a Zener diode (e.g., 3.3V) across the ADC input and ground to protect against over-voltage.
   - Add a small capacitor (e.g., 100nF) across the ADC input to filter out noise.
5. **Write the Code:**

   Here is an example code snippet for measuring the voltage using the PIC16F877A in MPLAB X IDE with XC8 compiler:

Title: Measure 0 to 100Vdc using micro controller

```
C program

#include <xc.h>
#define _XTAL_FREQ 20000000  // Define oscillator frequency for delay

void ADC_Init(){
    ADCON0 = 0x41;  // Turn ON ADC and select Fosc/8 as the
conversion clock
```

6. **Write the Code:**

   Here is an example code snippet for measuring the voltage using the PIC16F877A in MPLAB X IDE with XC8 compiler:

```
c

#include <xc.h>
#define _XTAL_FREQ 20000000  // Define oscillator frequency for delay

void ADC_Init(){
    ADCON0 = 0x41;  // Turn ON ADC and select Fosc/8 as the
conversion clock
ADCON1 = 0x80;  // Configure ADC voltage reference and result format
(right justified)
}

unsigned int ADC_Read(unsigned char channel){
    if(channel > 7) return 0;  // ADC has 8 channels, 0-7
    ADCON0 &= 0xC5;  // Clear existing channel selection bits
    ADCON0 |= channel << 3;  // Set new channel
    __delay_ms(2);  // Acquisition time to charge the hold capacitor
    GO_nDONE = 1;  // Start ADC conversion
    while(GO_nDONE);  // Wait for conversion to complete
    return ((ADRESH << 8) + ADRESL);  // Return result
}

void main(){
    unsigned int adc_value;
    float voltage, input_voltage;

    TRISA = 0xFF;  // Configure PORTA as input
    ADC_Init();  // Initialize ADC

    while(1){
        adc_value = ADC_Read(0);  // Read ADC value from channel 0
        voltage = (adc_value * 5.0) / 1023.0;  // Assuming Vref+ is
5V and 10-bit ADC resolution
        input_voltage = voltage * ((100.0 + 3.3) / 3.3);  // Convert
to input voltage

        // Now you can use input_voltage as needed
        __delay_ms(500);  // Delay for a while
    }
}
```

Title: Measure 0 to 100Vdc using micro controller

Explanation of the Code:

1. **Initialization:**
   - `ADC_Init()`: Initializes the ADC module.
     - `ADCON0`: Configures ADCON0 register to turn on the ADC and set the conversion clock.
     - `ADCON1`: Configures ADC voltage reference and result format.
2. **Reading ADC Value:**
   - `ADC_Read()`: Reads the ADC value from the specified channel.
     - `ADCON0` configuration ensures correct channel selection.
     - `GO_nDONE`: Starts ADC conversion and waits for completion.
3. **Voltage Calculation:**
   - Convert the ADC value to voltage (`voltage`).
   - Calculate the actual input voltage using the voltage divider ratio (`input_voltage`).
4. **Loop:**
   - Continuously read and calculate the voltage in the `while` loop.

Schematic Representation:

```
scss
High Voltage (0-100V)
        |
      R1 (100kΩ)
        |
      ----> ADC Pin (PIC)
        |
      R2 (3.3kΩ)
        |
      GND
```

Note:

- Adjust the code and configurations according to the specific PIC microcontroller and your application requirements.
- Ensure resistors have an adequate power rating and follow all safety protocols when working with high voltages.

This implementation will allow you to measure 0 to 100V DC using a PIC microcontroller.

Name: Kothi Sai varun

23013_009