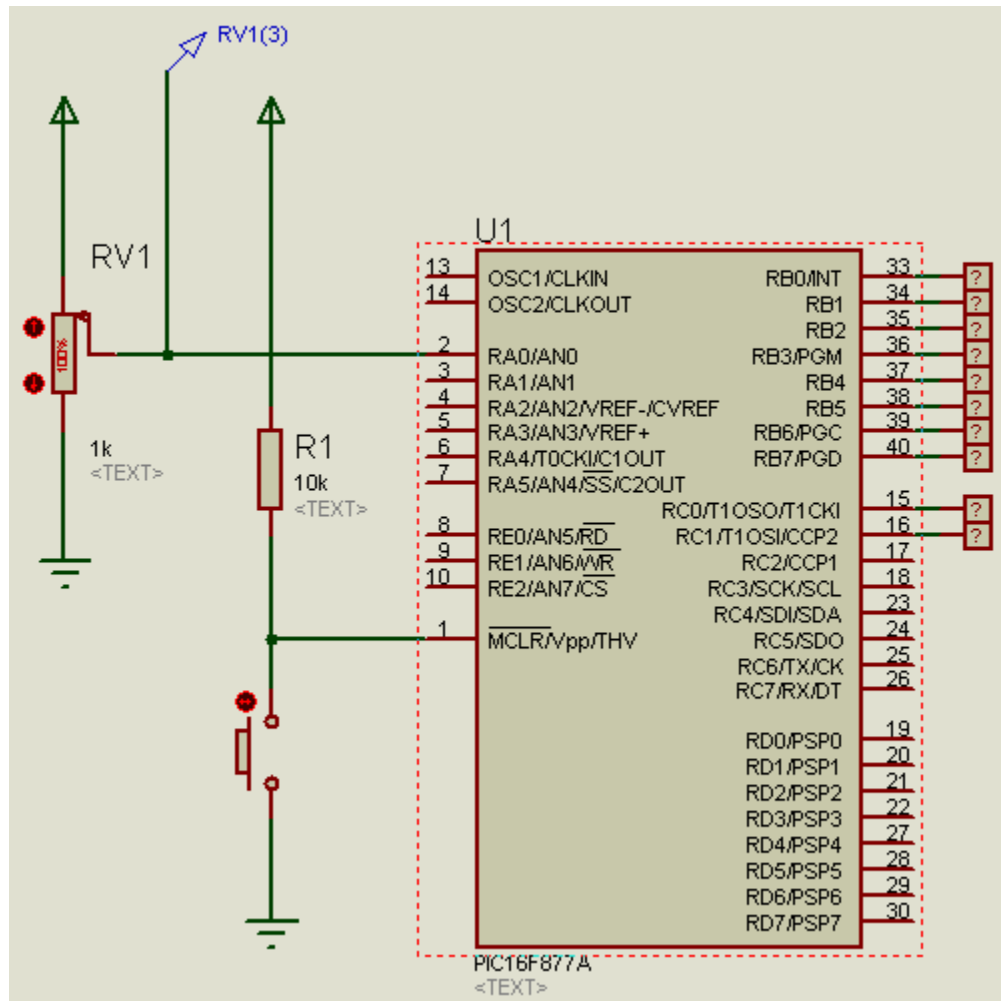


In Campus Activity 2
ECPE402
Nov 16, 2021 | 0100P – 0330P

Construct the circuit below in Proteus.



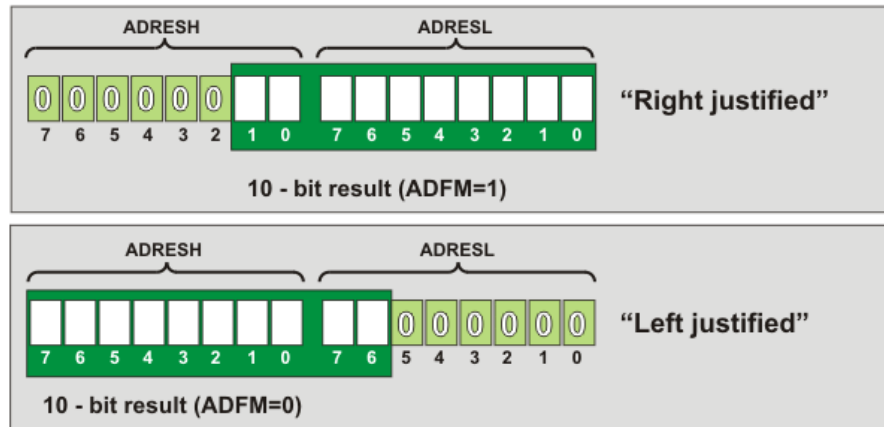
Note:

- The oscillator to be used is 8 MHz
- Use POT-HG component for the potentiometer
- Use voltage probe mode to measure the voltage across the potentiometer

For the detailed discussion of ADCON0 and ADCON1 of PIC16F877A, read page 127 of the datasheet.

A/D Result Format (ADCON0)

The digital result of the 10-bit ADC module is stored in the registers ADRESH and ADRESL with two formats namely left justified and right justified as shown below.



Source: <https://www.mikroe.com/ebooks/pic-microcontrollers-programming-in-assembly/analog-modules>

Create a project named my_adc in MikroC and encode the program below

```
unsigned int digital_value;

void init_adc_module()
{
    /*
        ADCS1:ADCS0 (bit 7:6) = 00, clock conversion is Fosc/
                                if ADCS2 = 0 of ADON1
        CHS2:CHS0 (bit 5:3) = 000, only AN0 is selected
        GO/DONE' (bit 2) = 0, A/D conversion not in progress
        ADON (bit 0) = 0, A/D converter module is shut-off and consumes no
        operating current

        Default: ADON0 = 0x00
    */
    ADON0 = 0x00;

    /*
        ADFM (bit 7) = 1, right justified
        ADCS2 (bit 6) = 0, clock conversion is Fosc/2
                                if ADCS1:ADCS0 = 00 of ADON0
        PCFG3:PCFG0 (bit 3:0) = 1110, only AN0 is will be configured
                                as an analog input pin
    */
    ADON1 = 0x8E;
}
```

```

void init_port()
{
    TRISA.B0 = 0;      // RA0/AN0 = input pin
    TRISB.B0 = 1;      // RB0 = input pin

    /*
        Digital value of the analog input
        RD1:RD0 = Bit 9:8
        RC7:RC0 = Bit 7:0
    */

    TRISB = 0x00;      // PORTB = output port
    TRISC.B0 = 0;      // RC0 = output pin
    TRISC.B1 = 0;      // RC1 = output pin

    PORTB = 0x00;
    PORTC = 0x00;
}

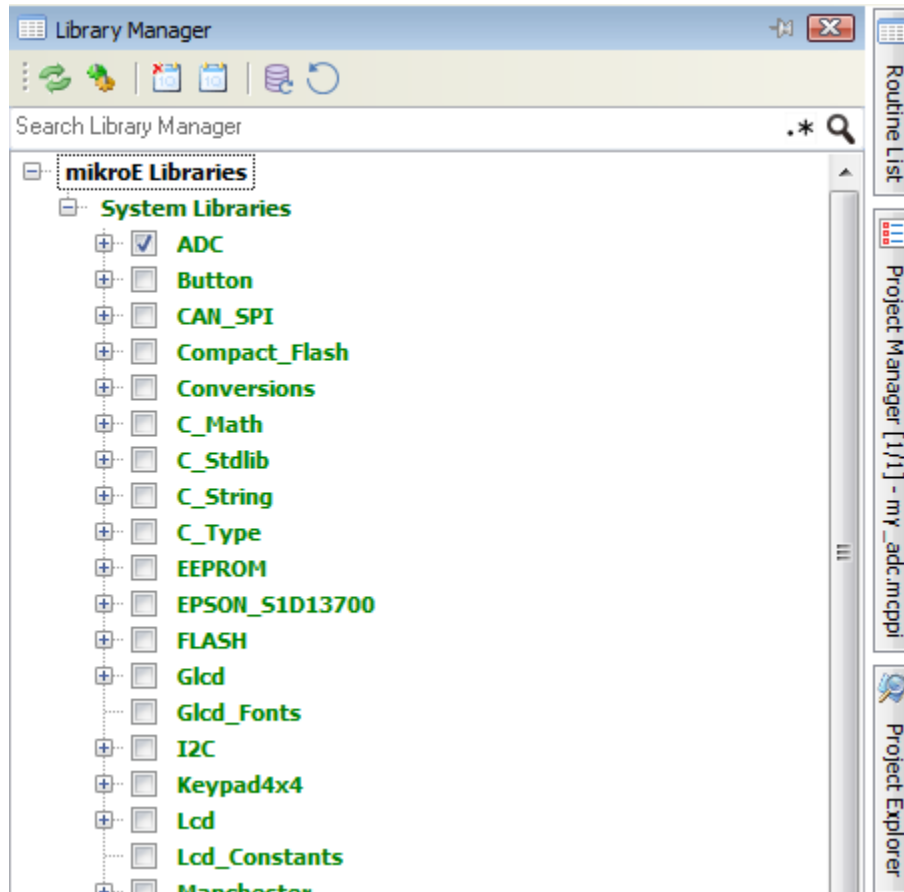
void main()
{
    init_port();
    init_adc_module();

    while (1)
    {
        /*
            read channel 0 of the ADC module
            to read channel 1, use ADC_Read(1)
        */
        digital_value = ADC_Read(0);    // read channel 0, if

        PORTB = digital_value;
        /*
            shifting the 10-bit value 8 bits position to the right
            so that bit 9 and bit 8 is positioned to bit 1 and bit 0
            respectively
        */
        PORTC = digital_value >> 8;
    }
}

```

Make sure to check the ADC library of MikroC in order to use the function `ADC_Read()` as shown below.



Configure the project by clicking Project -> Edit Project and follow the configuration below

The image shows the MikroC configuration window. On the left, there is a list of configuration options, each with a dropdown menu. The 'Oscillator Selection' is set to 'XT oscillator'. The 'Watchdog Timer', 'Power-up Timer', 'Brown-out Reset', 'Low-Voltage (Single-Supply) In-Circuit Serial Programming', 'Data EEPROM Memory Code Protection', 'Flash Program Memory Write', 'In-Circuit Debugger Mode', and 'Flash Program Memory Code Protection' are all set to 'Disabled'. On the right, the 'MCU and Oscillator' section shows 'MCU Name' as 'P16F877A' and 'MCU Clock Frequency [MHz]' as '8.000000'. Below this, the 'Build Type' is set to 'Release' (with 'ICD Debug' as an option) and 'Heap Size' is 0. The 'Configuration Registers' section shows 'CONFIG : \$2007 : 0x2F49'. On the far right, there are buttons for 'Load Scheme', 'Save Scheme', and 'Default'. At the bottom right, there is a button for 'General Output Settings ...'.

Compile the program.

Simulate the circuit in Proteus by adjusting the potentiometer using the arrow up and arrow down symbols. While adjusting the resistance, observe the voltage and its digital equivalent in Port B and Port C.