ECE 3720

Analog to Digital Converter

Section 2

Brandon Wider

Date Performed: April / 10 / 2020

Lab 10

ABSTRACT:

Use an analog to digital converter to read the resistance of a force sensitive resistor shown through the LEDs outputted in a binary.

INTRODUCTION:

The purpose of this lab is to show how to use the analog to digital converter by showing that the analog values can be translated to a digital signal processed by the digital logic of the circuit.

EXPERIMENTAL PROCEDURES:

1) Configure the ADC register
2) Set the data direction with the TRISxbits
3) Disable interrupts
4) Set up edge polarity
5) Create a while loop to continuously run the program when the user presses the force sensitive resistor, which acts as the interrupt of this experiment
6) Create and __ISR function to write the output of the 8 LEDs
7) Enable the interrupt, the force sensitive resistor.
8) Wire up the circuit diagram
9) Run and test the diagram and code

RESULTS and DISCUSSION:

If done in the lab, the 8 LEDs light up accordingly to how much pressure we place on the force sensitive resistor. When the pressure is increased, the value the LED displayed in binary also increases. An issue I had was figuring out how to add the code for the force sensitive resistor. I little research was required, and some changes were made in areas like setting the values of each variable for each step and calling in the right bits for each step.

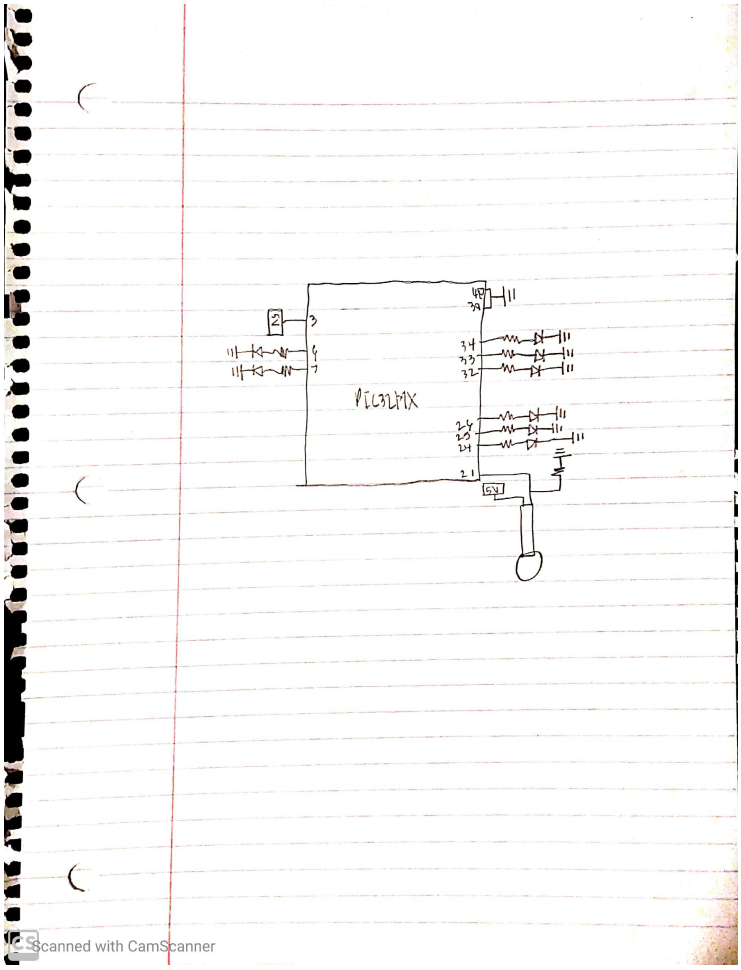
CONCLUSSION:

The analog to digital converters reliance on pressure can be used in many forms for digital components such as running a motor, increasing/decreasing the volume and/or the frequency of a speaker, and any other electronic devices.

REFERENCES:

Clemson University's ECE 372 Lab 10 Powerpoint.

FIGURES AND TABLES:



CODE:

```
#include<plib.h>

void __ISR(23) AN(void)
{
    LATC = ((ADC1BUF0 + ADC1BUF1 + ADC1BUF2 + ADC1BUF3 +
ADC1BUF4)/5)>>2;
    IFS0bits.AD1IF = 0;
}

main()
{
    INTEnableSystemMultiVectoredInt();
```

```
TRISAbits.TRISA0 = 1;
TRISAbits.TRISA1 = 1;
TRISBbits.TRISB0 = 1; //Input

// Configure analog port pins
ANSELAbits.ANSA0 = 1;
ANSELAbits.ANSA1 = 1;
ANSELB = 1;

TRISC= 0; //Output
AD1CON1bits.ON = 0;
IEC0bits.AD1IE = 0;

// Select Analog Inputs
AD1CHSbits.CH0NA = 1;
AD1CHSbits.CH0SA = 2;

// Select the format of the ADC
AD1CON1bits.FORM = 0b000;

// Sample Clock Source
AD1CON1bits.SSRC = 0b111;

// Voltage Reference Source
AD1CON2bits.VCFG = 0b011;

// Scan Mode using CSCNA
AD1CON2bits.CSCNA = 0;

// Set number of conversions per interrupt
AD1CON2bits.SMPI = 0b100;

// Set buffer Fill mode
AD1CON2bits.BUFM = 0;

// Select the MUX
AD1CON2bits.ALTS = 0;

// Select the ADC clock source
AD1CON3bits.ADRC = 0;

// Select the sample time
AD1CON3bits.SAMC = 12;

// Select the ADC clock prescaler
AD1CON3bits.ADCS = 6;
```

```
    // Configure the ADC interrupt
    IFS0bits.AD1IF = 0;
    IPC5bits.AD1IP = 1;
    IEC0bits.AD1IE = 1;

    // Start the conversion sequence
    AD1CON1bits.ASAM = 1;

    // Turn the ACD module on
    AD1CON1bits.ON = 1;
    while(1) {
    }

}
```