17.368 Data Conversion and Lab
Lab 5
Fall 2008

Analog to Digital Conversion Using the PIC16F684 Microcontroller

Objectives

1) To demonstrate the following:

    a. Analog to Digital Conversion using the PIC microcontroller
    b. Compare the PIC16F684 ADC function to that of the ADC0804 ADC

Materials

□	1	Pre-programmed PIC16F684
□	1	Breadboard
□	1	Oscilloscope
□	1	Voltmeter
□	1	DC Power Supply
□	1	LM7805 Voltage Regulator
□	1	100 pF Capacitor (101)
□	1	150 pF Capacitor (151)
□	1	1000 picofarad capacitor (102) … note: AKA 1 nanofarad or 0.001 microfarad
□	1	0.01 microfarad capacitor (103) – power circuit
□	3	0.1 microfarad capacitor (104) – power circuit and ADC0804 circuit
□	1	0.33 microfarad capacitor (334) – power circuit
□	1	10 μF Capacitor (Tantalum)
□	11	470 ohm resistors (yellow violet brown)
□	1	1 k ohm resister (brown black red)
□	8	1.3 kΩ Resistors (brown orange red) Note … can also use 470 ohm
□	1	10 kΩ Resistors (brown black orange)
□	1	10k ohm potentiometer
□	1	LED – power circuit
□	20	LEDs or two 10 LED Bargraph Displays
□	2	10 LED Bargraph Display or 20 LEDs

WARNINGS AND PRECAUTIONS

1) Never remove the PIC16F684 from an energized circuit

2) Do not construct circuits while energized

3) Follow electrical safety precautions

Background Information

Before we get into the specifics of the PIC16F684 Analog to Digital Converter (ADC) converter lets focus on the fundamentals of A/D conversions.

First … an analog signal is continuous in amplitude & time within certain limits, i.e., it changes smoothly without interruptions. An example is a sinusoidal signal.

A Digital signal is discrete in amplitude and time … i.e., it can only take certain specific values within certain limits at specific time intervals. When numbers are assigned to these steps (usually binary numbers) the result is a digital signal. An example is a square wave, a 1-Bit digital signal with its high level being a binary '1' and its low level being a binary '0'.

An Analog-to-Digital converter (ADC) is an electronic circuit that changes or converts a continuous analog signal into a digital signal without altering its critical content. An Analog-to-Digital converter samples an analog waveform at uniform time intervals and assigns a digital value to each sample.

Sampling is the process of analyzing the continuous analog signal with measurements taken at discrete and standard intervals. In conjunction with sampling … the device needs to be able to "hold" the signal for a finite amount of time. During the "hold" time, the ADC will perform its function of converting the signal from analog to digital. The "hold" is performed via a storage capacitor. Up to the time the "hold" is commanded, the capacitor is tracking/sampling the analog signal.

The PIC16F684 Analog-to-Digital converter (A/D) allows conversion of an analog input signal to a 10-bit binary representation of that signal. The PIC16F684 has eight analog inputs, multiplexed into one sample and hold circuit. The output of the sample and hold is connected to the input of the converter. The converter generates a binary result via successive approximation and stores the result in a 10-bit register. The voltage reference used in the conversion is software selectable to either VDD or a voltage applied by the VREF pin.

In this course we will provide a pre-programmed PIC16F684 microcontroller that will execute analog to digital conversion. The actual program is included in this lab for those who may want to further investigate the functionality of the software, however, it is not the intent of this lab to write or modify the program. The purpose of this lab is to perform a comparison between ADC of a specialty chip to that of a microcontroller.

To better understand the workings of the microcontroller, the following is provided.

There are three registers available to control the functionality of the A/D module:

ANSEL (Register 9-1)
ADCON0 (Register 9-2)
ADCON1 (Register 9-3)

The A/D conversion can be supplied in two formats, either left or right shifted. The ADFM bit (ADCON0<7>) controls the output format.

Steps to follow for analog to digital conversion:

1. Configure the A/D module
2. Configure A/D interrupts (if desired)
3. Wait the required acquisition time
4. Start conversion
5. Wait for A/D conversion to complete
6. Read A/D Result register pair
7. For the next conversion … go to step 1 or step 2 as required

This lab will use the PIC16F684 ADC function. The results will be verified by comparing the actual input to the LED converted value along with a comparison with the ADC0804.

Pre-Lab Preparation

1. Download the component Data Sheets available on the course web page
2. Construct the required circuit on your breadboard

Procedure

Experiment 1. ANALOG TO DIGITAL CONVERSION USING THE PIC16F684

a. Build the voltage regulator circuit shown in figure 1 on your breadboard. **DO NOT INCLUDE JP2.** Some guidelines for your construction follows:

1. Locate your voltage regulator circuit in a corner of your board; group the components together to minimize space.

2. Apply 9 volts to the V_{IN} connection.

3. Using the voltmeter and an oscilloscope, verify that the +5V_REG port reads +5 volts and the output is clean and free of interference.

4. Power down the circuit.

b. Construct the circuit shown in Figure 2. This will be a visual aid in determining if voltage is present (when the LED is turned on).

1. Apply 9 volts to V_{IN} connection and verify that the LED lights.

2. Power down the circuit.

c. Construct the circuit shown in Figure 3.

d. Construct the circuit shown in Figure 4. Make the connection to the PIC16F684 (RA0 which is pin 13). This circuit will provide an analog input voltage to both the microcontroller and the stand alone ADC (we will construct the circuit later in the lab). The analog voltage is a DC voltage which will be used to simulate a slow changing analog signal.

e. The PIC16F684 has been pre-programmed with the Code provided in Figure 5. b. The Code provided will execute as follows:

1. Performs Analog to Digital Conversion using the PIC16F684 and the circuitry shown in Figure 3.

2. Figure 3 LEDs will display the digital result of the ADC operation. The Most Significant Bit (MSB) is represented by LED D7. This corresponds to 2^{10}. The two Least Significant Bits (LSB) will not be displayed; therefore, D0 represents the 2^2 bit. **We are ignoring the two least significant bits.**

3. Potentiometer RP1 is connected to RA0 and will be used to vary the voltage to the ADC input of the PIC microcontroller (Figure 4).

4. Examine the code provided along with the schematic shown in Figure 4. You may want to try to understand how the Code was developed, what features of the PIC16F684 are turned on/off as well as what the expected voltages will be seen by the ADC.

f. Turn RP1 counter clock wise (CCW) until it hits the stops. All LEDs should be extinguished at this point.

g. Slowly turn RP1 clock wise (CW). You should see the LEDs start to turn on/off. The binary representation of the LEDs will increase until all the

LEDs are on when RP1 hits the other stop.

h. Turn RP1 CCW until it hits the stops again. All LEDs should be extinguished at this point.

i. Measure the voltage from RA0 (pin 13) to ground or at RP1 pin 2.

j. Record the voltage in Table 1.

k. If needed, rotate RP1 CW until your voltmeter reads 0.0 volts (NOTE: FULL CCW and 0.0 may be the same position). Record the state of each LED, the actual voltage on your voltmeter, and calculate the voltage by using the LED display.

l. Repeat for all the values listed in Table 1.

TABLE 1

Voltage (Volts)	LED State ("1" or "0")								Actual Voltage	Calculated Voltage
	D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**		
FULL CCW										
0.0										
0.5										
1.0										
1.5										
2.0										
2.5										
3.0										
3.5										
4.0										
4.5										
5.0										
FULL CW										

m. **QUESTION:** How accurate was the ADC as compared to the actual voltages applied to Pin 13 or RP1 pin 2?

Experiment 2. ADC COMPARISION BETWEEN PIC16F684 AND ADC0804

a. Construct the ADC circuit shown in Figure 6. This is the same circuit we constructed in Labs 3 and 4 utilizing the ADC0804 (this is also Figure 9 of the ADC0804 Data Sheet).

b. Connect the output of the circuit shown in Figure 4 to both RA0 of the PIC16F684 (should already be connected) and to pin 6 (V_{IN}) of the ADC0804. We are now applying the same analog signal to both circuits. This will allow us to compare the conversions of both devices.

c. Recall that Figure 3 LEDs will display the digital result of the ADC operation. The Most Significant Bit (MSB) is represented by LED D7. This corresponds to 2^{10}. The two Least Significant Bits (LSB) will not be displayed; therefore, D0 represents the 2^2 bit. **We are ignoring the two least significant bits.**

d. The ADC0804 is only an eight bit ADC, therefore LED D7 (MSB) corresponds to 2^8 and D0 represents the 2^0 bit (LSB). **KEEP THIS IN MIND WHEN COMPARING THE TWO ADC CONVERTERS.**

e. Turn RP1 counter clock wise (CCW) until it hits the stops. All LEDs in both circuits should be extinguished at this point.

f. Slowly turn RP1 clock wise (CW). You should see the LEDs start to turn on/off. The binary representation of the LEDs will increase until all the LEDs are on when RP1 hits the other stop.

g. Turn RP1 CCW until it hits the stops again. All LEDs should be extinguished at this point.

h. Measure the input analog voltage at either RA0 (pin 13) of the PIC16F684 or pin 6 of the ADC0804 to ground or at RP1 pin 2.

i. Construct 2 more copies of Table 1, one to record the results of the PIC16F684 ADC and one for the results of the ADC0804. Record the voltage.

j. If needed, rotate RP1 CW until your voltmeter reads 0.0 volts (NOTE: FULL CCW and 0.0 may be the same position). Record the state of each LED, the actual voltage on your voltmeter, and calculate the voltages for both converters by using the LED displays.

k. Repeat for all the values listed in Table 1.

l. How do the results of the two converters compare? What are the advantages and disadvantages of both converters?

Summary:

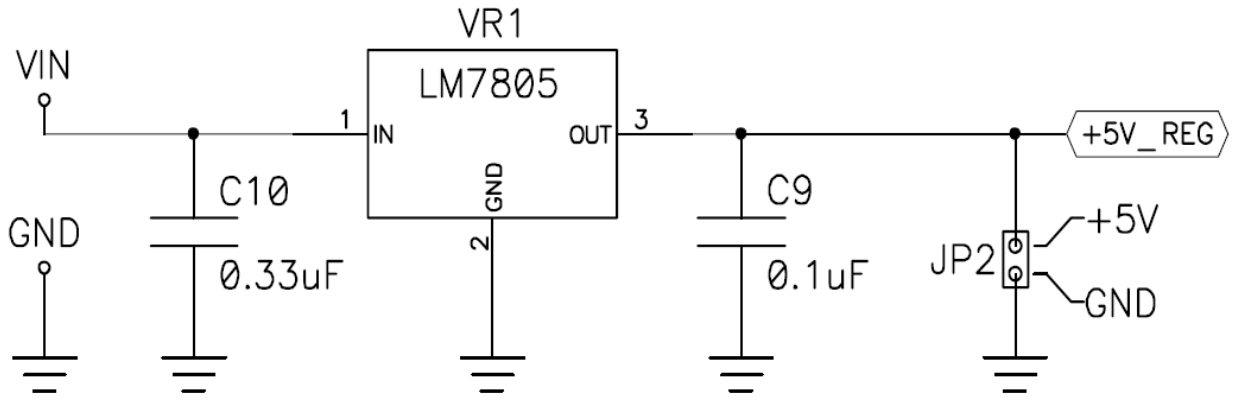
The microcontroller can be used for more than just digital input/output as demonstrated by this lab. The PIC16F684 has eight analog to digital converters available for use. The conversion performed by the device provides an accurate conversion of the analog value. Ease of use is also a plus.

In real world applications we see that sensors provide analog outputs. Without the ability to convert these signals to a digital format would make processing these signals more complex.

Questions:

1. None.

Figure 1 – Voltage Regulator Circuit



7805 Pin Connections - Top View

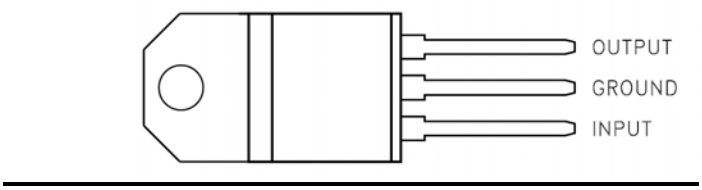
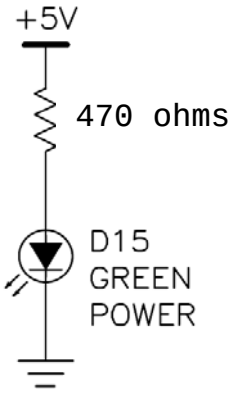


Figure 2 - Power Indication Circuit



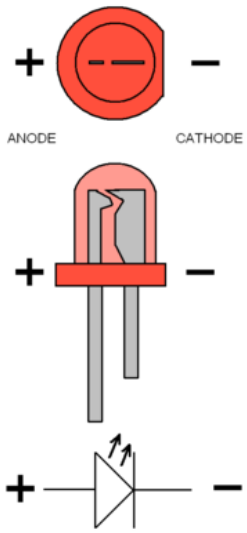


Figure 3 – PIC16F684 Circuit

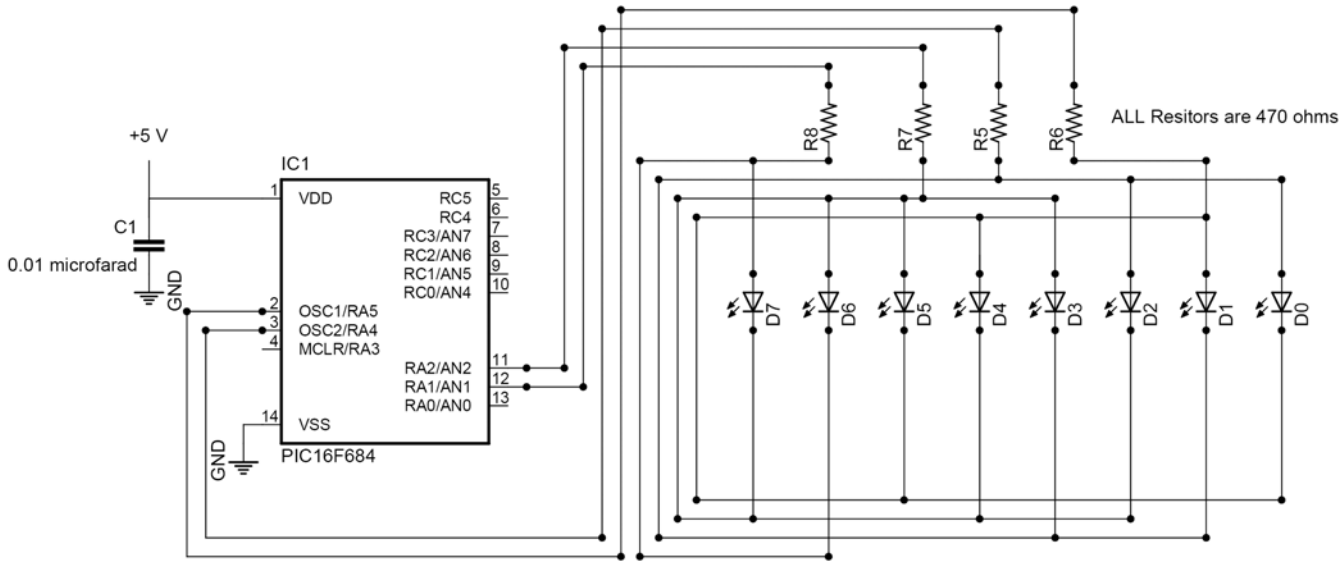


Figure 4 –ADC Input Circuit

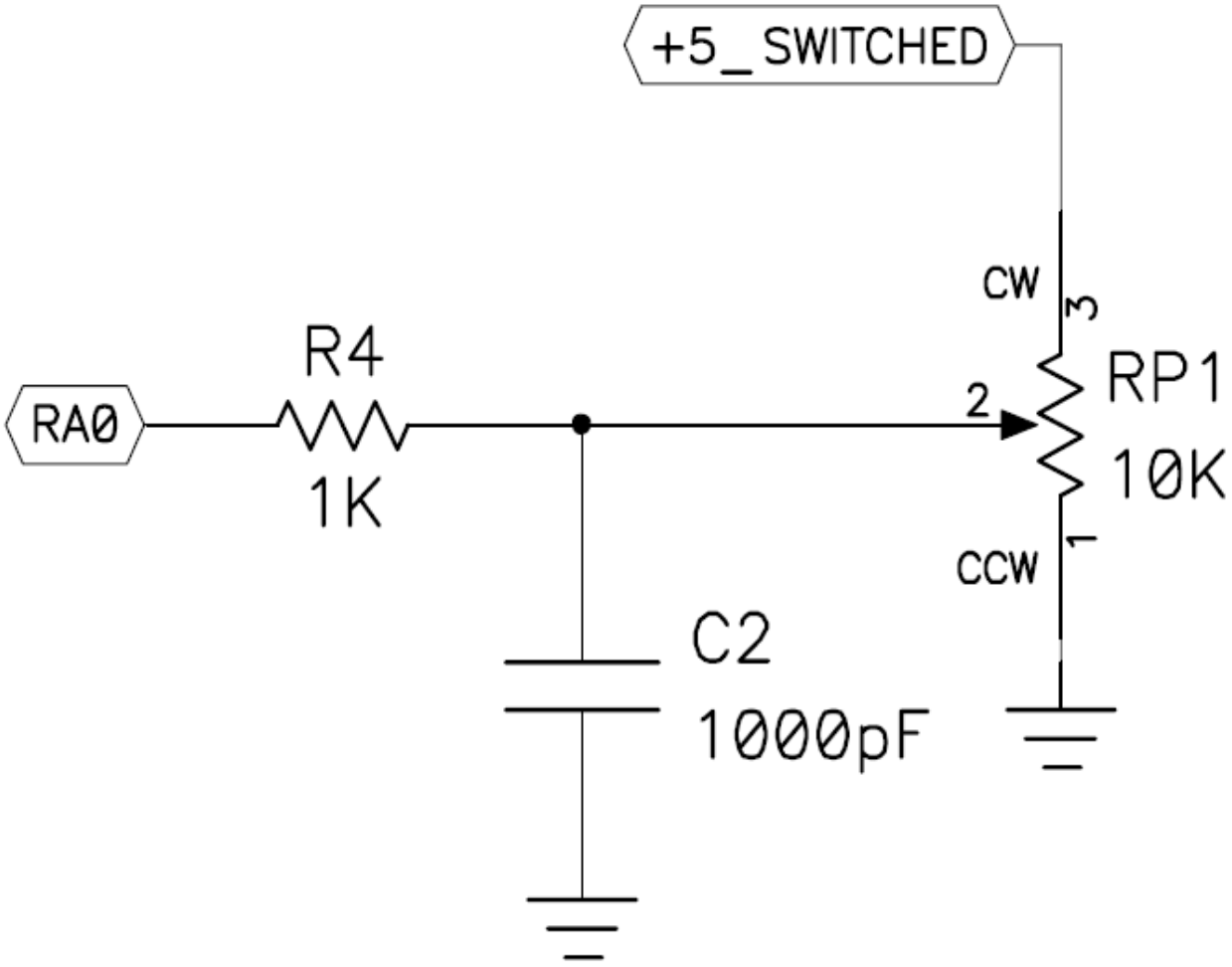


Figure 5 ADC Pre-Programmed Code

```
#include <pic.h>


/*****************************************************
* Function: main
*
* Description:  D0 - D7 on PICkit 1 will Display the results of the ADC
*
* Notes:
*
* RA0 - Input from RP1
*
*
*
* Returns:  This routine contains an infinite loop
*
*****************************************************/

/* Configuration Word */

__CONFIG(INTIO & WDTDIS & PWRTEN & MCLRDIS & UNPROTECT \
  & UNPROTECT & BORDIS & IESODIS & FCMDIS);


void PORTA_init(void);
void ADC_Disp(void);
void Delay_LED_On(void);

int ADC_Value = 0;

const char PORTA_Value[8] = {
      0b010000,   // D0
      0b100000,   // D1
      0b010000,   // D2
      0b000100,   // D3
      0b100000,   // D4
      0b000100,   // D5
      0b000100,   // D6
      0b000010};  // D7

const char TRISA_Value[8] = {
      0b001111,   // D0
      0b001111,   // D1
      0b101011,   // D2
      0b101011,   // D3
      0b011011,   // D4
      0b011011,   // D5
      0b111001,   // D6
      b111001};   // D7
```

Continued next page

Continued from previous page

```
main()
{
    PORTA_init();

    ANSEL = 1;                  //  Just RA0 is an Analog Input
    TRISA0 = 1;                 //  Corresponding TRIS bit is set as input

    ADCON0 = 0b00000001;        //  Turn on the ADC
                                //   Bit 7          - Left Justified Sample
                                //   Bit 6          - Use VDD
                                //   Bit 4:2        - Channel 0
                                //   Bit 1          - Do not Start
                                //   Bit 0          - Turn on ADC

    ADCON1 = 0b00010000;        //  Select the Clock as Fosc/8


    ADC_Disp();
    GODONE = 1;                 // Start A/D Conversion


    while(1 == 1)               //  Loop Forever
    {

        if (GODONE == 0)                // Is A/D Conversion complete?
            {     ADC_Disp();           // Display A/D Conversion Results
                  ADC_Value = ADRESH;   // Get new A/D value
                  GODONE = 1;           // Start the next A/D Conversion
              }
          else                          // A/D Conversion still in progress
                  ADC_Disp();

      }
}

/******** END OF main ROUTINE **************************/

/***************************************************
* Function: PORT_init
*
* Description:  Initializes PORTA to a known condition
*
* Notes:    None
*
* Returns:  None
*
***************************************************/
void PORTA_init(void)
{
      PORTA = 0;                //  All PORTA Pins are low
      CMCON0 = 7;               //  Turn off Comparators
      ANSEL = 0;                //  Turn off ADC

      return;

}
/******** END OF PORTA_init *********
```

Continued next page

Continued from previous page

```
/***************************
* Function: ADC_Disp
*
* Description:  Displays the value of A/D Conversion on D0 - D7
*
* Notes:
*
*
*
* Returns:  None
*
*****************************************************/
void ADC_Disp(void)
{

      int i;

      for (i = 0; i < 8; i++ )
      {                              // Loop through Each of the 8 LEDS

        Delay_LED_On();              // Allows time for individual LEDs to light

            if ((ADC_Value & (1 << i)) == 0)
                PORTA = 0;
            else
                PORTA = PORTA_Value[i];
                TRISA = TRISA_Value[i];
        }  //

      return;

}
/******** END OF ADC_Disp ************************/

/*****************************************************
* Function: delay_LED_On
*
* Description:  Causes a delay in program execution
*
* Notes:
*
* Delay was determined through trial and error
*
* Returns:  None
*
*****************************************************/
void Delay_LED_On(void)
{
      int j;

      for (j = 0; j < 60; j++);           //  Display "On" Loop

      return;

}
/******** END OF Delay_LED_On ************************/
```

Figure 6 ADC0804 Circuit

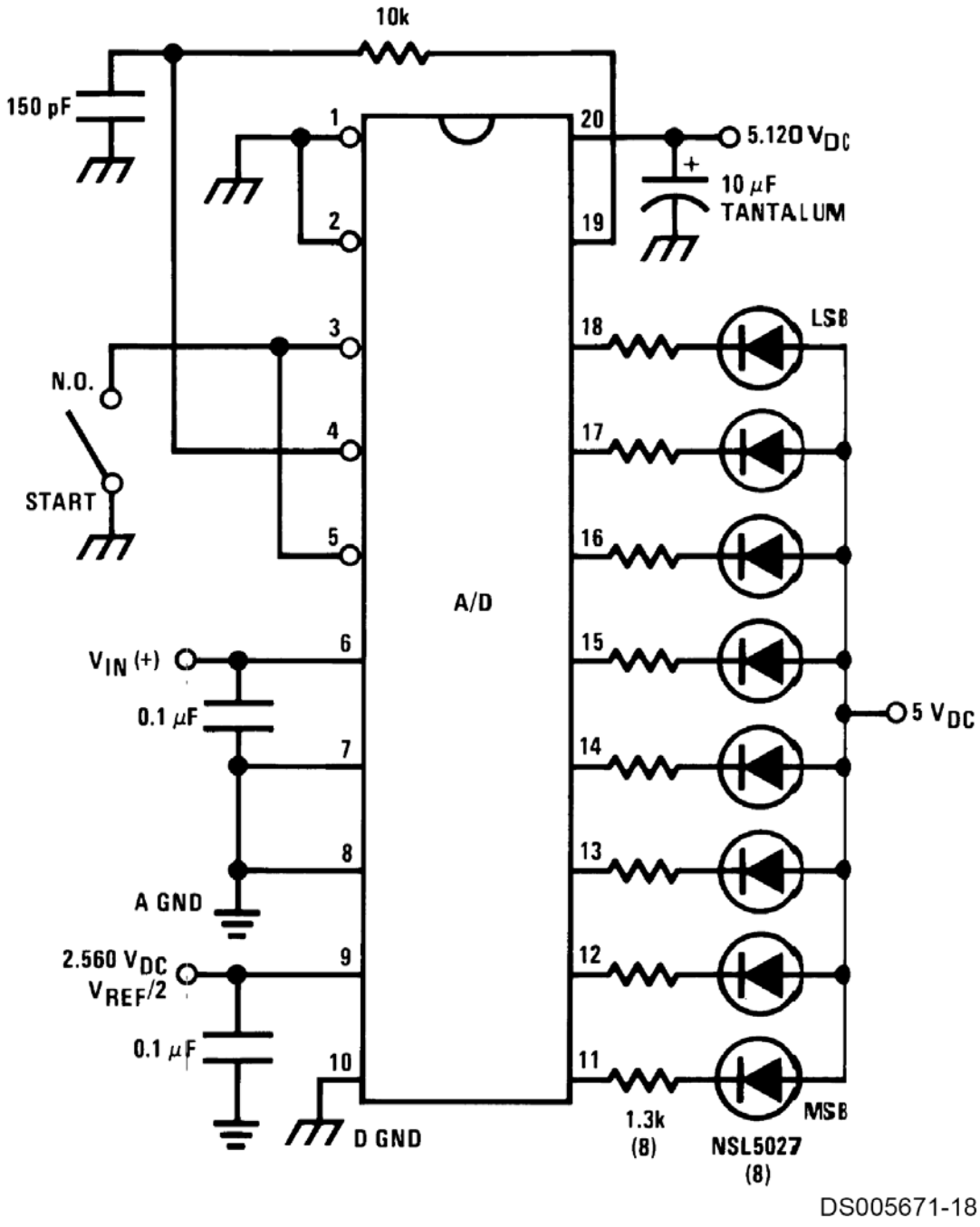


FIGURE 9. Basic A/D Tester