

Lab08 Part C - Analog to Digital Conversion (adapter from Basic Analog and Digital - Parallax)

Build Your Own Digital DC Voltmeter

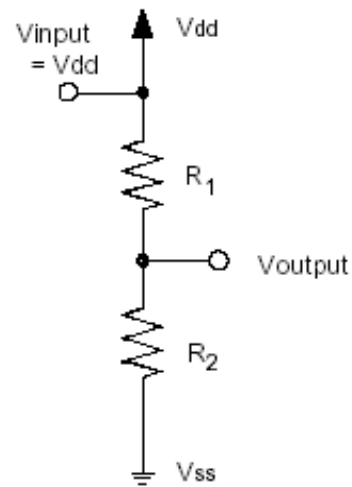
A digital DC voltmeter (DC DVM) is a handy tool for measuring voltage between two contact points. In this experiment, we will build a DVM for measuring DC voltage in the 0 to 5 Volt range. A common use for a DC DVM is testing the voltage (potential) between the two terminals on a battery.

A digital voltmeter is so named because it displays its measurements with digits. The digits 0 through 9 and a decimal point are used to display the voltage measurements as decimal values. The digits 0 and 1 could be used. It would still be a "digital" voltmeter, but it would have binary display instead of a decimal display. Making sense out of each measurement would be time consuming. Since our DVM processes its measurements in binary, we'll start with a binary display and then modify it to the more conventional and easy to read decimal display.

Although information about analog voltage can be processed efficiently with binary devices, the voltage has to be sampled and described using binary numbers first. The ADC0831 is a common integrated circuit that does this job. It describes the analog information with binary numbers for devices that process binary information, such as the BASIC Stamp.

In this experiment, we will make a DVM using the BASIC Stamp together with the ADC0831 integrated circuit. A thermistor (i.e., temperature sensor) will be wired to an inverter opamp which will provide signal gain to the ADC.

A voltage divider is shown in Figure 1. If R_1 and R_2 are equal V_{output} will be exactly half of V_{dd} ($V_{dd} = +5VDC$). So $V_{output} = 2.5$ volts. If R_2 decreases, then V_{output} decreases and if R_2 increases, V_{output} increases.



The ADC0831 Integrated Circuit - An 8-bit Analog to Digital Converter The ADC0831 is an integrated circuit referred to as an 8-bit analog to digital converter (A/D converter) with synchronous serial output. Let's look at what each of these terms mean:

Figure 1

§ An integrated circuit (IC) is a circuit with microscopic components implanted on the surface of a silicon wafer. The Analog and Digital Parts Kit has three chips used in these experiments. Each chip is a black casing with eight pins. The black casing houses and protects an integrated circuit.

§ An A/D converter measures an analog voltage sample and returns a binary number that describes the sample.

§ 8-bit is the number of binary digits the ADC0831 uses to describe the analog voltage it samples. 8-bit is also the resolution of the A/D converter. You can count from 0 to 255 (decimal) using an 8-bit binary number. This means that the ADC0831 can approximate the voltage it measures as one of 256 levels. A higher resolution converter, such as 12-bit, would break the same voltage range into 4096 levels because you can count from 0 to 4095 with 12 binary bits.

§ With synchronous / serial communication with the ADC0831, the BS2 sends a clock signal to time the sending of each serial output bit.

The BASIC Stamp will be programmed to read and store the 8-serial-bits transmitted by the ADC0831. We'll also program the BASIC Stamp to display the decimal equivalent of the binary output. The BASIC Stamp must also be programmed to send binary control signals to make the ADC0831 do its job.

Figure 2 shows a pinout map of the ADC0831. Each pin has a number and a label. The number is important for getting the wires connected to the right pins when constructing your circuit. The labels indicate the function of each pin.

The notation for the ADC0831's inputs and outputs works as follows:

Vin(+) is the analog input, and D0 is the serial output.

VREF and Vin(-) are used to bias the IC. Bias: A method of applying specific voltage levels at certain

places in a circuit to calibrate or tune it.

Vss or GND are used for supplying power to the IC.

Vcc or +5VDC

/CS stands for active low chip select

CLK stands for clock

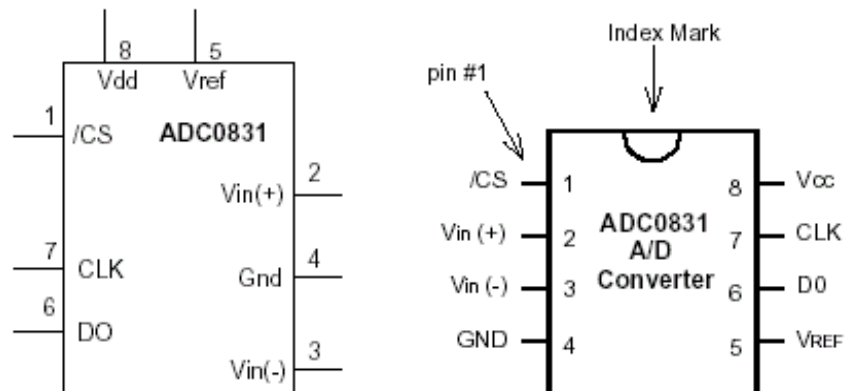


Figure 2

To prime the ADC0831 for taking a measurement, the /CS pin has to receive a signal from the BASIC Stamp that starts high, then goes low. This signal has to stay low for the duration of the conversion. Then the CLK input must receive a single clock pulse to signify that the conversion should start at the next clock pulse. For this IC, a clock pulse starts low, goes high, then goes low again. It takes 8 more clock pulses to complete the conversion. Each time a clock pulse is received by the CLK input, another of the serial bits is sent by the D0 output.

Construction

1. ADC Circuit - Figure 3, 4, 5, 6 & 7 shows the schematic for this experiment. This is a fairly simple circuit to build, so let's try it without the breadboard example. Hopefully you're getting the hang of the list of connections described by a schematic. Remember, when working with the connections to an IC, use the index mark on the chip along with the pin map to figure out the pin numbers.

- a. Pin 1 on the ADC0831 is connected to pin P0 on the BASIC Stamp.
- b. Pins 3 and 4 on the ADC0831 are connected to Vss (i.e., ground).
- c. Pins 5 and 8 on the ADC0831 are connected to +5VDC.
- d. Pins 7 and 6 on the ADC0831 are connected to BASIC Stamp pins P1 and P2 respectively.

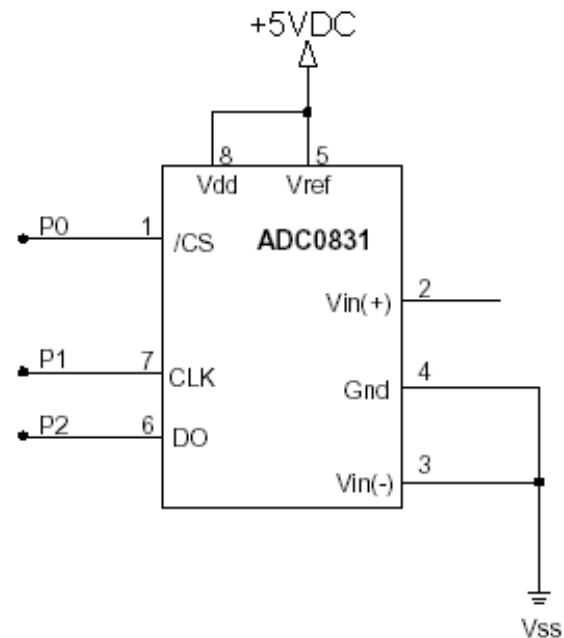


Figure 3

2. Thermistor Circuit - This circuit buffers and amplifies the change in temperature (i.e., change in resistance) of the thermistor. The thermistor changes nonlinearly and its resistance decreases as temperature increases (i.e., negative thermal coefficient resistor).

- a. One resistor in the 10K ohm array is used as the R1 of a voltage divider with the 10K thermistor replacing R2.
- b. The resulting signal goes to a buffer opamp to boost the current.
- c. The signal from the buffer opamp goes into an inverter opamp with a gain of about 18 (i.e., $(47K + 47K)/5.1K = 18.43$). 47K = yellow, violet, orange, 5.1K - green, brown, red.
- d. The amplified signal then goes into the analog to digital chip (i.e., ADC0831 - pin 2 - Vin).

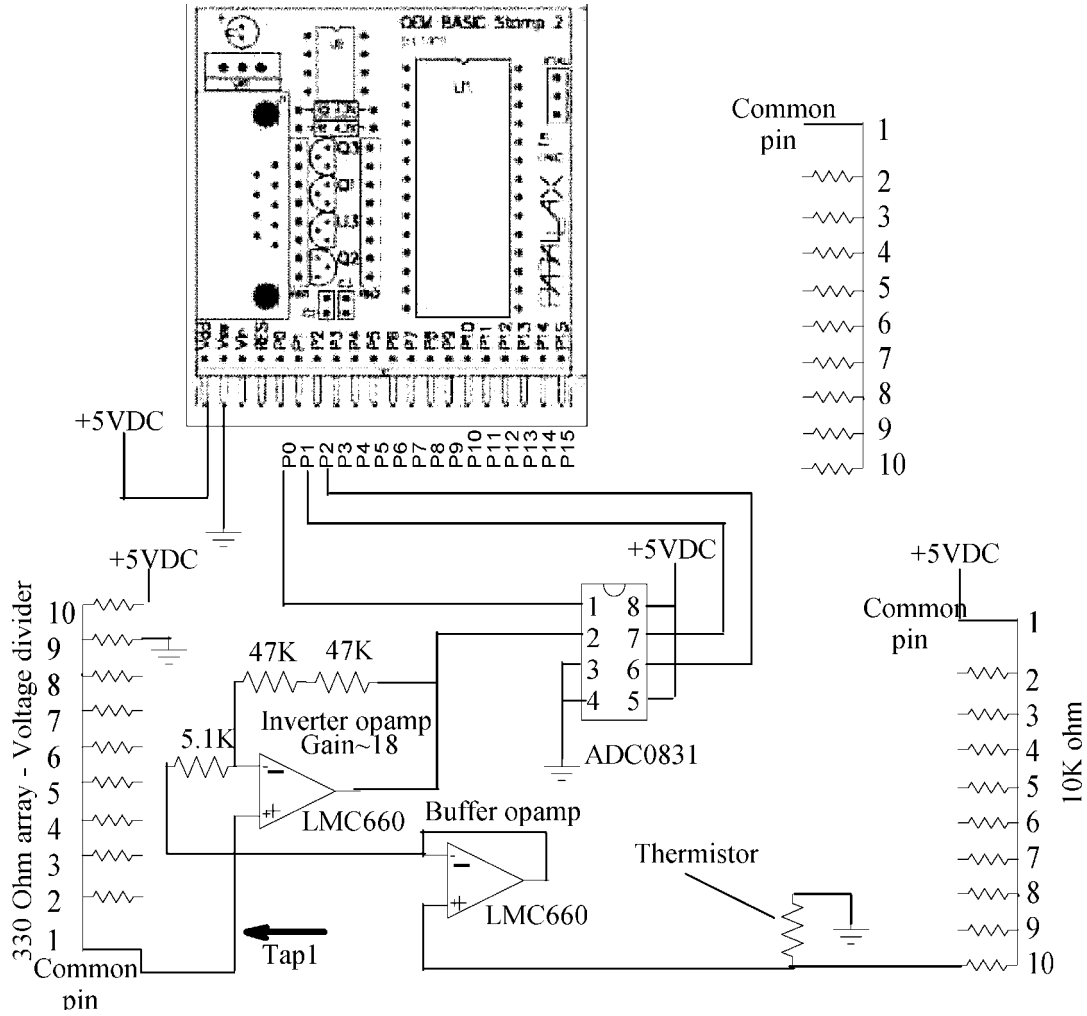


Figure 4

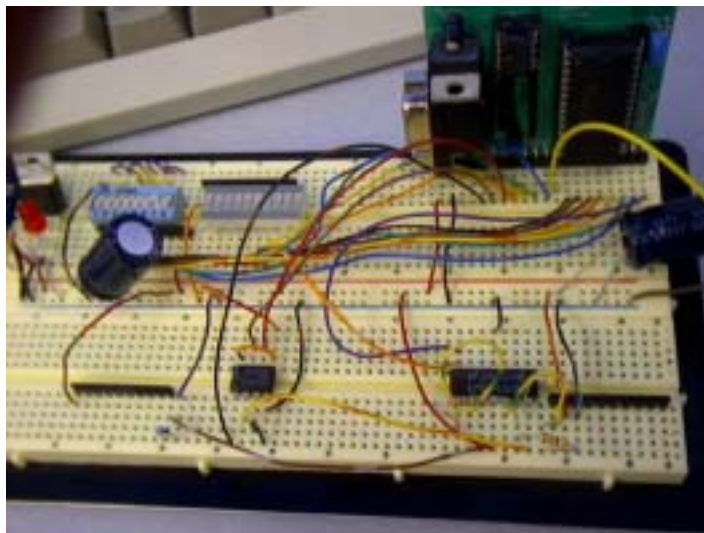


Figure 5

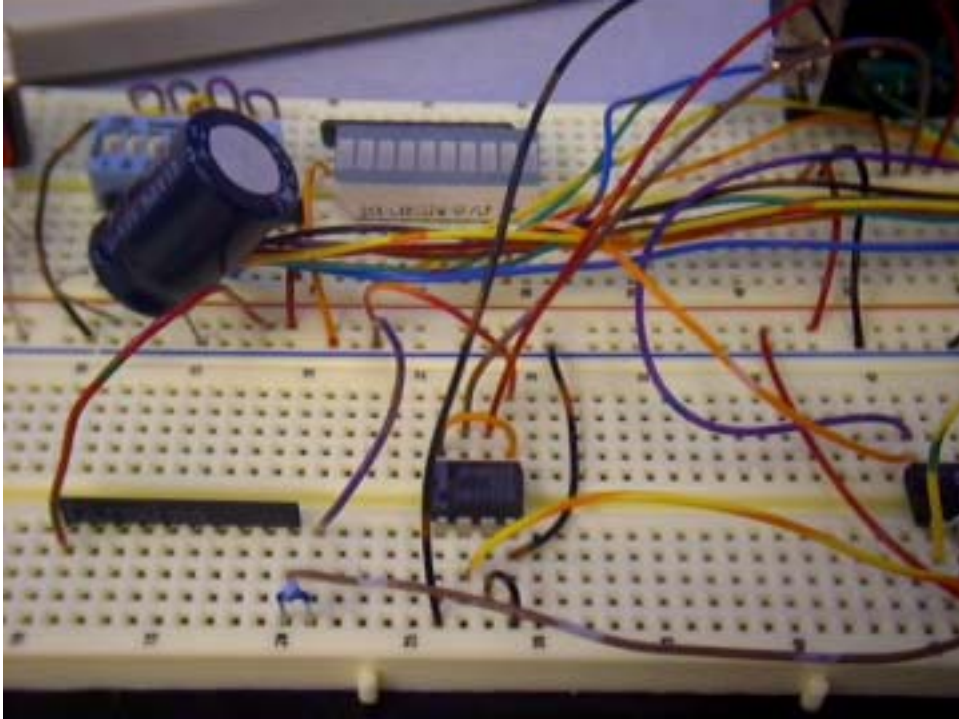


Figure 6

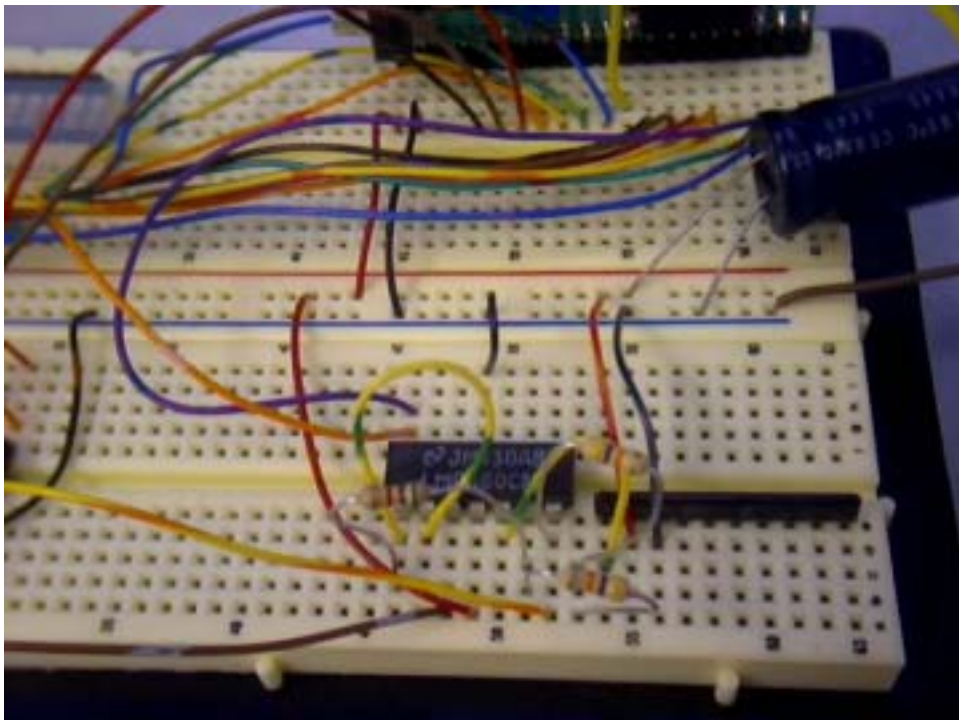


Figure 7

3. Enter the program for a functional DC voltmeter. This program displays the 8-bit serial output of the ADC0831. A subroutine has been left in so that you may convert the binary information to volts in the future.

```
'ADC0831 Binary output display.
'Declarations.
adcbits var byte
v var byte
R var byte
v2 var byte
v3 var byte

CS con 0
CLK con 1
D0 con 2

'Start display.
debug cls

'Main routine.
main:

gosub ADCDATA
gosub CALC_VOLTS
gosub DISPLAY

goto main

ADCDATA:
high CS
low CS
low CLK
pulsout CLK,210
shiftdin D0,CLK,msbpost,[adcbits\8]
return

CALC_VOLTS:
return

DISPLAY:
debug home
debug "8-bit binary value: ", bin8 adcbits
return
```

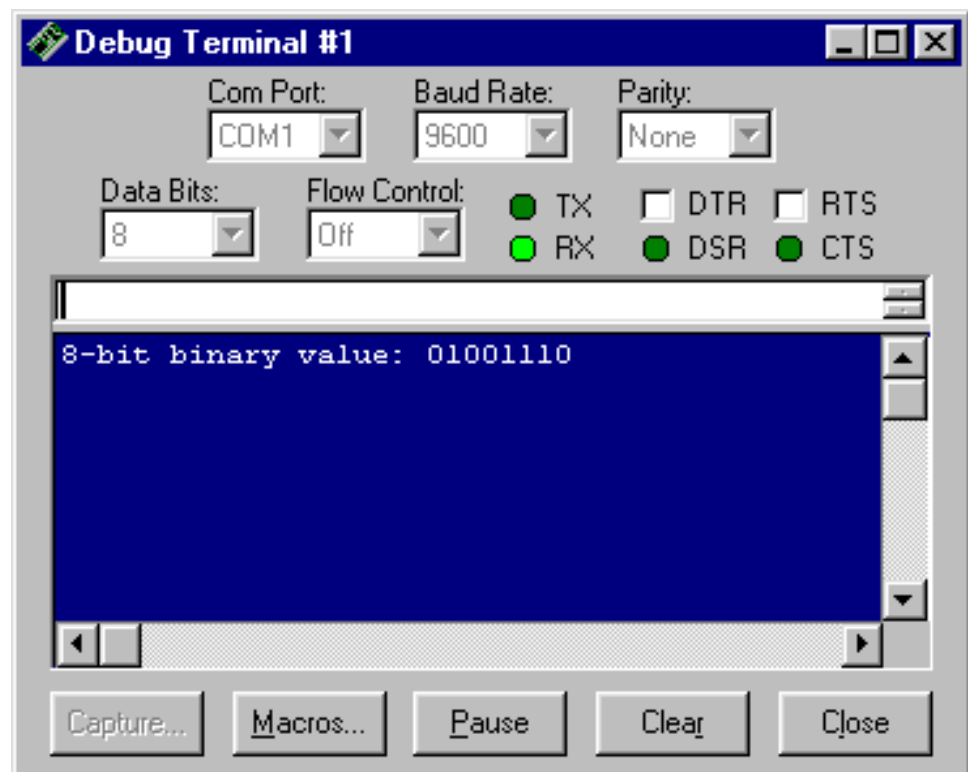
4. Program Details

- a. The variable declarations section, and it begins with a comment explaining that this is the declarations section. This program uses just the `adcbits` variable at present. We'll add code that will make use of the other four variables, `v`, `R`, `v2`, and `v3`.
- b. Three constants are defined using the `con` directive. After we define these constants, we can use `CS` in place of the number 0, `CLK` in place of the number 1, and `D0` in place of the number 2. The names for the constants were chosen to correspond with the ADC0831's pin labels. The numbers were chosen based on BASIC Stamp I/O pin numbers.
- c. Next there's a loop that contains three `gosub` commands. The `main:...goto main` routine runs 3 different subroutines over and over again. The subroutines are named `ADCDATA:`, `CALC_VOLT:`, and `DISPLAY:`. The label `main:` is used in the same manner that we used the `loop:` label in the first two programs. The label name `main:` was chosen because, as the comment preceding this routine indicates, it's the "main routine" in the program.
- d. The subroutine `ADCDATA:`
 - i. Sends control signals to and collects output data from the ADC0831. This subroutine is where the usefulness of the `con` directive really shows. P0 on the BASIC Stamp is connected to the `/CS` pin on the ADC0831. Likewise, pins P1 and P2 are connected to `CLK` and `D0`. When sending signals to the `/CS` pin, we can enter a command like `high CS` instead of `high 0`. It makes more sense when writing the code, and it makes deciphering the code easier too. It's also easier to change one constant in the top of the program should you decide to connect the ADC0831 to a different BASIC Stamp I/O pin.
 - ii. The command `high CS` sends a high signal to the ADC0831's `/CS` pin. To start a conversion, we need to send a high signal (5 volts). Then we need to send a low signal (0 volts) to the `/CS` input on the ADC0831 using `low CS`. The signal sent to the ADC0831's `/CS` input needs to stay low for the duration of the conversion.
 - iii. The `low CLK` command is necessary so that the clock pulses take the right form. Using this command guarantees that the next command (`pulsout`) will send a clock pulse that has the right shape, low-high-low. Sending high and low signals using the high and low commands is an alternative to the `out0=1` and `out0=0` techniques.
 - iv. The `pulsout CLK,210` command sends a clock pulse to the ADC0831's `CLK` input. This is the first clock pulse, and all it does is tell the ADC0831 to start converting on the next clock pulse. Because of this, we don't need to check for input from `D0` after this first clock pulse. Since we set the clock low just before this command, `pulsout` sends the desired low-high-low signal. The duration of the high segment is twice the number specified in the `pulsout` command, in microseconds (us). $1 \text{ us} = 1/1,000,000$ of a

second. Therefore the duration of this high segment is $2 \text{ us} \times 210 = 420 \text{ us}$.

- v. The command `shiftin D0,CLK,msbpost,[adcbits\8]` is a powerful instruction that takes care of all the synchronous serial communication so that we don't have to program it as we did in Experiment #2. In effect, this command sends clock pulses to the ADC0831's CLK input and reads output bits from ADC0831's D0 output. This command also loads each of the ADC 0831's output bits into the `adcbits` byte. In our case, the data pin is D0, a constant equal to the number 2. This constant is used to reference BASIC Stamp I/O pin P2 in this program. Likewise, the clock pin is CLK, which is a constant equal to the number 1, and it references BASIC Stamp I/O pin P1. The mode in this case is `msbpost`, and it's one of four transmission modes that can be used in this command. It indicates that the the ADC0831's output bits are ready after the clock pulse's negative edge, the transition from high to low. It also indicates that the bits are transmitted in descending order, starting with the MBS. `[adcbits\8]` means the data is shifted into the `adcbits` variable, and 8-bits are expected.
 - vi. The `CALC_VOLTS`: subroutine is empty. Should you desire you can place a subroutine that will calculate the measured voltage to the hundredth's decimal place. At present, the `DISPLAY`: subroutine just displays the binary output for each analog voltage sample taken by the ADC0831.
 - vii. The command `debug home, cr, "8 bit binary value: ", bin8 adcbits` sends the cursor to the top-left "home" position in the Debug window. Then it prints the message in quotes. The modifier `bin8` makes it so the value of the `adcbits` variable is displayed as 8 binary digits. If the number of digits displayed is likely to vary, when using the `debug home` command, always specify how many digits the numeric outputs should have with modifiers like `bin8`, `dec3`, etc. When `debug cls` is used, it's OK to skip specifying the number of digits, so modifiers such as `bin` and `dec` can be used instead. The `debug home` command is better for programs that cycle through loops where the Debug window display is updated frequently and rapidly. When `debug cls` is used under these circumstances, the repeated clearing the Debug window causes a flicker that makes the display difficult to read.
 - viii. The return command sends the program back to the line immediately following the `gosub display` command.
- e. The command `debug CR, CR, "Decimal value: ", dec3 adcbits` tells the Debug window to display two carriage returns followed by the message in quotes, followed again by the 3-digit decimal value of `adcbits`. If the actual number only has one or two digits, the Debug window will automatically display leading zeros since `dec3` was specified. For example, the number 7 will display as 007, and the number 85 as 085, etc.

5. Interpreting the Binary Output - The ADC0831 measures an analog voltage at its input. Then it sends the BASIC Stamp a binary number describing the value it measured. For now, we'll focus on a voltage scale that starts with 0 volts and ends at 5 volts. With an 8-bit binary number, you can start counting with 00000000 and count all the way up to 11111111 (i.e., range = 255, levels = 256). Translated to decimal numbers, it's the same as counting from 0 to 255. When applied to a 5 Volt scale that starts at 0 volts, it's the same as counting from 0 to 5 volts using 255 voltage steps. For the 5 Volt scale, when the ADC0831 measures 0 volts, you get 00000000. When it measures 5 volts, the output is 11111111. It turns out that the Debug window output 01001110 from Figure 8 is the same as the decimal number 78. Decimal - 78 in turn corresponds to a measured voltage of 1.53 volts (i.e., 5volts x (78/255)).



6. Assuming you've gotten this far . . .

Figure 8

- a. Show the TA that you get something like Figure 8 from the construction of the board and running the program.
- b. Show the TA that when you heat the thermistor the binary value changes, then slowly returns to the starting value. Unfortunately, there isn't much change even after amplification, perhaps only five or six bits will show a change. Blowing on the thermistor heat the sensor up more than using your fingers.

- c. Show the TA that when you change the input of the ADC0831 (i.e., pin 2 of ADC) from the thermistor circuit to ground the binary output is near zero. Some noise may cause it not to display zero.**
- d. Show the TA that when you change the input of the ADC0831 to +5 volts the binary output is near 1111,1111.**
- e. Show the TA that when you change the input of the ADC0831 to the 330 Ohm voltage divider at Tap 1 on Figure 4, your binary output is near 1000,0000 in binary (i.e, 128 in decimal or 2.5 volts).**