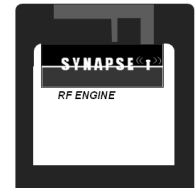


3 A/D Conversion



Analog devices like speakers, temperature sensors, strain gauges, position sensors and light meters need to communicate with digital circuits in a way that goes beyond techniques in threshold triggering. An Analog-to-Digital Converter (ADC) can assist in managing the Mixed Signal cross between these two electronic circuits. An ADC converts an analog signal into series of binary numbers, each number being proportional to the analog level measured at a given moment. The digital words generated typically by the ADC are fed into a microprocessor or microcontroller, where they can be processed, stored, interpreted, and manipulated. Examples of ADC usage include but not limited to data- acquisition systems, digital sound recording, and within simple digital test Instruments (e.g., light meters, thermometers, etc.).

Triggering Simple Logic Responses from Analog Signals

There are times when you need to drive logic from simple on/off signals generated by analog devices. For example, the device of choice may want to latch an alarm (via a flip-flop) when an analog voltage, say, one generated from a temperature or light sensor, reaches a desired threshold level. Although the control requirement task is simple, an ADC can assist in reading the continuous voltage signals produced by the analog component and wire to the appropriate threshold triggering circuit.

This unit will explore how to use the SNAP Protoboard to read analog data, process it and control an LED, electromechanical relay or a small dc electric motor.

Pedagogical Concept: ADC Value Analysis

To convert an ADC value to its equivalent analog unit, the following examples are presented below and on the following pages.

GPIO Pin Mapping to the Analog Input Channels

The Synapse –Wireless RF Engine Module have several pins available for reading “continuous” analog signals as shown in the Table of **Figure 3-1**.

ANALOG INPUT CHANNEL	GPIO PINS
0	18
1	17
2	16
3	15
4	14
5	13
6	12
7	11

Figure 3-1. Analog Input Channel to GPIO Mapping Table

GPIO18 will be used in the lab project which corresponds to Channel 0 of the Mapping Table.

The ADC Count Value vs the Analog Unit

The relationship between the ADC count value and its equivalent analog unit is shown below.

EQ 1:
$$\frac{V}{ADC} = \frac{Vsense}{ADC\ Value}$$

whereby:
$$\frac{V}{ADC} = \frac{3.2mV}{ADC}$$

To determine the value of “V_{SENSE}” the following analysis equation is used.

EQ 2:
$$Vsense = (V/ADC) \times ADC_Value(raw)$$

Example Problem 1:

Determine the value of “ V_{SENSE} ” of a sensor’s raw ADC Value of a 452.

Solution:

$$\begin{aligned}\text{EQ 2: } V_{\text{sense}} &= (V/\text{ADC}) \times \text{ADC_Value}(\text{raw}) \\ V_{\text{sense}} &= (V/\text{ADC}) \times \text{ADC_Value}(\text{raw}) \\ &= (3.2\text{mV}/\text{ADC}) \times 452 \\ &= 1.4464\text{V}\end{aligned}$$

The Target Sense value ($\text{Target}_{V_{\text{sense}}}$) can be determined using the following analysis equation below.

$$\text{EQ 3: } \text{Target } V_{\text{sense}} = (V/\text{ADC}) \times \text{ADC_Value}(\text{threshold})$$

Example Problem 2:

An alarm (piezo-buzzer) is to sound off when a sensor’s raw ADC Value is less than the ADC threshold value of 400. Determine the $\text{Target}_{V_{\text{sense}}}$ value of the ADC detection circuit.

Solution:

$$\begin{aligned}\text{EQ 3: } \text{Target } V_{\text{sense}} &= (V/\text{ADC}) \times \text{ADC_Value}(\text{threshold}) \\ \text{Target } V_{\text{sense}} &= (V/\text{ADC}) \times \text{ADC_Value}(\text{threshold}) \\ &= (3.2\text{mV}/\text{ADC}) \times 400 \\ &= 1.28\text{V}\end{aligned}$$

The two analysis equations will be put to practice for the following the Lab Project.

Lab 1: ADC Basics

This lab project will investigate the basics of ADC by displaying the equivalent ADC count value within the Portal Software Event Log window. A potentiometer wired as Rheostat will allow for new adjusted ADC count values to be displayed in the Event Log window by rotating a knob attached to its shaft. The analysis equations illustrated earlier will be validated against measured values taken from a dc voltmeter.

Part Lists

- (1)SN171 Proto Board
- (1)SN163 Bridge Demonstration Board or
- (1)SN132 SNAPstick USB Module
- (1)10K Ω Potentiometer
- (1)10K Ω resistor (Brown, Black, Orange)
- (1)red LED
- (1)330 Ω resistors (Orange, Orange, Brown)
- (2)1.5V Batteries or DC Power Supply
- Smart Logic Probe Kit
- Solderless Breadboard
- DMM (Digital Multimeter) (Optional)
- 22 AWG (American Wire Gauge) Solid Wire

Assembly and Test Procedure

1. Build the Wireless ADC Circuit shown in **Figure 3-2** on a solderless breadboard.
2. Type the SNAPpy Script shown in **Figure 3-3** using the Portal IDE Text Editor.
3. Save the file as “Read_Rheostat_CtrlBuzr.py”

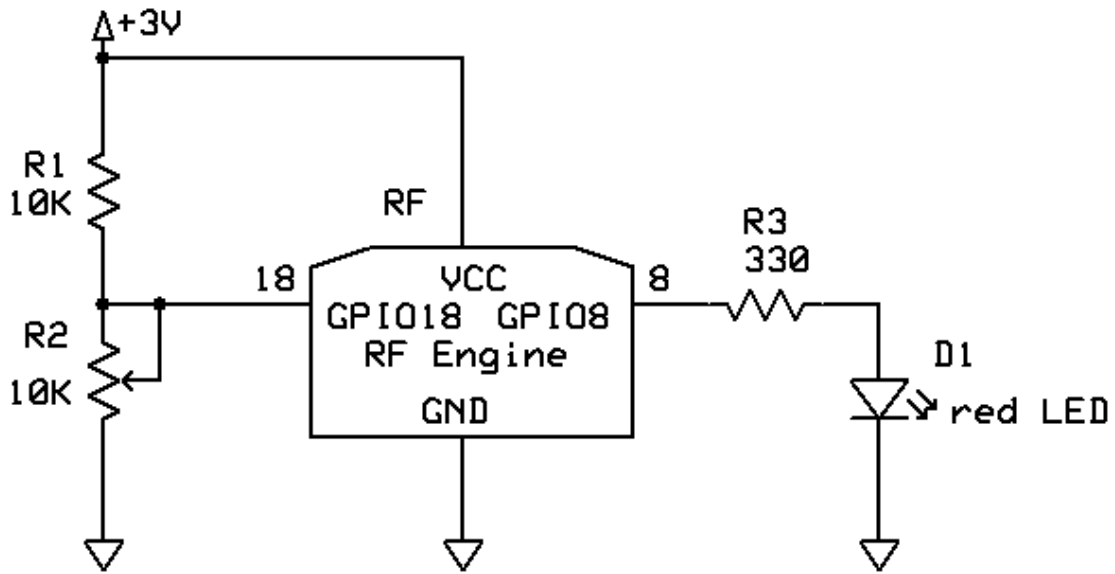


Figure 3-2. Wireless ADC Circuit

4. Apply power to the Proto Board, and the 10K Ω Rheostat Interface circuit.



A Rheostat is a potentiometer with the wiper arm wired to either pin of the device (See Figure 3-2)

5. Upload the script (the **image**) to the Proto Board.
5. On the Portal IDE, click the Int STDOUT Icon.
6. Adjust the 10K Ω potentiometer (rheostat) until the raw ADC count value reads 452 on Portal's Event Log. See **Figure 3-4**.
7. Measure the input voltage at Port GPIO18 with a DC Voltmeter. Did the measured input voltage equal to the calculated value of 1.4464V?

8. Adjust the 10K Ω potentiometer (rheostat) until the raw ADC count value reads 400 on Portal's Event Log. See **Figure 3-5**.
9. Measure the input voltage at port GPIO18 with a DC voltmeter. Did the measured input voltage equal to the calculated value of 1.281V?

```

1  """
2  Read Rheostat Demo for EK2100 (for use on proto-board)
3
4      This SNAPPy script is used on the proto-board module to demonstrate
5      how to read data from an analog sensor. Also, to demonstrate software
6      re-use, the TemperatureAlarm.py SNAPPy script was used as the core
7      programming code for the demo.
8
9
10     Hardware Requirements:  10K Potentiometer
11                            10K Pull-up resistor
12
13
14     """
15     # Wiring Diagram
16     #
17     #   +3
18     #   |
19     #   >
20     #   < 10K Pullup
21     #   >
22     #   -- |----->|GPIO18 |
23     #   | >         |   GPIO9 |----->Output
24     #   |-->< 10K Pot. |-----|---
25     #   >
26     #   |
27     #   | GND
28     # Use Synapse Evaluation Board definitions
29     from synapse.evalBase import *

```

Figure 3-3. Code Listing for Read_Rheostat_CtrlBuzzer.py

```

31 def startupEvent():# Setup and Initializing Hardware
32     """This is hooked into the HOOK_STARTUP event"""
33     global adcThreshold, buzzerPin
34
35
36     # Setup the buzzer
37     buzzerPin = 9
38     writePin(buzzerPin, False)
39     setPinDir(buzzerPin, True)
40
41 def sensorUpdate():# Read Sensor Function (User - Defined)
42     global adcThreshold
43     adcThreshold = 400
44     adcValue = readAdc(0) # Check the data
45     print "The raw adc value: ", adcValue #Print the value within the Event Log
46     print "adcThreshold value:", adcThreshold #Print the Threshold value within the Event Log
47
48     if adcValue < adcThreshold: # If Sensor Value is less than the Threshold Value, turn on the buzzer
49         soundTheAlarm() # buzzer Function Call
50
51 def soundTheAlarm():# buzzer Function (User - Defined)
52     """Sound the alarm"""
53     global buzzerPin
54     pulsePin(buzzerPin, 500, True) # turn buzzer ON
55
56 def doEverySecond():# Poll Sensor every second
57     sensorUpdate()# Read sensor Function Call
58
59
60 def timer100msEvent(msTick):# SNAPpy 100msecond Function
61     """Hooked into the HOOK_100MS event. Called every 100ms"""
62     doEverySecond()# Poll sensor every second Function call
63
64
65 # Set event hook functions
66 snappyGen.setHook(SnapConstants.HOOK_STARTUP, startupEvent)
67 snappyGen.setHook(SnapConstants.HOOK_100MS, timer100msEvent)

```

Code listing continued

```

Read_Rheostat_CtrlBuzzer: The raw adc value: 452
adcThreshold value:400

Read_Rheostat_CtrlBuzzer: The raw adc value: 452
adcThreshold value:400

Read_Rheostat_CtrlBuzzer: The raw adc value: 452
adcThreshold value:400

Read_Rheostat_CtrlBuzzer: The raw adc value: 453
adcThreshold value:400

Read_Rheostat_CtrlBuzzer: The raw adc value: 452
adcThreshold value:400

Read_Rheostat_CtrlBuzzer: The raw adc value: 452
adcThreshold value:400

```

Figure 3-4. ADC Count (452) and Threshold Values displayed on Portal's Event Log

```
Read_Rheostat_CtrlBuzzer: The raw adc value: 400
adcThreshold value:400
-----
Read_Rheostat_CtrlBuzzer: The raw adc value: 400
adcThreshold value:400
-----
Read_Rheostat_CtrlBuzzer: The raw adc value: 400
adcThreshold value:400
-----
Read_Rheostat_CtrlBuzzer: The raw adc value: 400
adcThreshold value:400
-----
Read_Rheostat_CtrlBuzzer: The raw adc value: 400
adcThreshold value:400
-----
Read_Rheostat_CtrlBuzzer: The raw adc value: 400
adcThreshold value:400
```

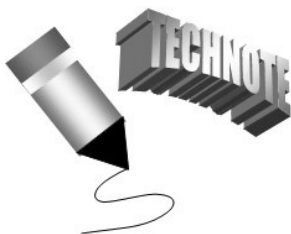
Figure 3-5. ADC Count (400) and Threshold Values displayed on Portal's Event Log



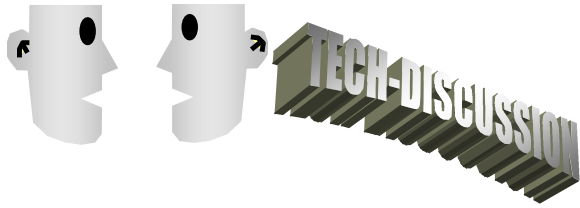
In the circuit schematic diagram shown in Figure 3-2 has a red LED indicator circuit wired to pin 8 (GPIO8). This red LED turns ON based on the following conditional relationship between the ADC Count value and the Threshold value. The line of conditional instruction responsible for turning ON the red LED is shown below.

```
if adcValue < adcThreshold: # If Sensor Value is less than the Threshold Value, turn on the buzzer
    soundTheAlarm() # buzzer Function Call
```

Therefore, by adjusting the 10KΩ rheostat for an ADC Count value less than the Threshold value, the red LED will turn ON. The LED indicator circuit can easily be replaced with a small piezo-buzzer, thereby turning a visual alarm into an audible device.



GPIO is the abbreviation for General Purpose Input – Output.



Analog to Digital Conversion is an important technique to use when matching continuous varying signals with discrete voltage levels. Although, the world is based on analog data, digital techniques can enhance them by improving performance, function and reliability. The SNAP Proto-board has 8 channels designated for A/D conversion. As illustrated in Lab Project 1, a simple voltage divider circuit can easily be interface with the wireless proto-board for sensor monitoring. By varying the 10K Ω potentiometer, Analog to Digital Count (raw) data was observed on Portal's Event Log. A conditional statement was used to establish a switching threshold (trigger) whereby the SNAP Proto-board can be used to control an external electrical load, in this case a simple LED Circuit. The next Lab Project will explore using actual sensors for monitoring physical stimuli and controlling a small dc electric motor.

Questions

1. Determine the voltage across a 10K Ω thermistor in series with a fixed 1K Ω and a 3V supply.
2. Using Figure 3-2 as a reference, redraw the circuit using the voltage divider circuit wired to pin GPIO18 of the RF Engine Module.
3. Draw the appropriate transistor driver circuit for switching a small dc motor.
4. Using lines of instruction 51-54 as a reference from the Read_Rheostat_CtrlBuzr.py code, write the appropriate software driver for the circuit drawn in Question 3.
5. Explain how the circuit in Figure 3-2 could be used to test solar cell batteries for small electronic calculators.
6. Draw a System Block Diagram of the Solar Cell Battery Tester based off the circuit explanation of Question 5.