EGR 262

Fundamental Circuits Lab

File: N262L7

**Lab # 7**

**Successive-Approximation ADC**

A. **Objectives**

 The objectives of this laboratory are to introduce the student to:

1. Comparators
2. Successive-approximation analog-to-digital conversion (ADC)
3. Writing a program for the Arduino UNO to implement a ***binary search*** as part of the ADC

B. **Materials**

 Breadboard with Arduino UNO

 Adaptor (120V AC to 12 VDC, 1000 mA)

 Seven 220 Ω resistors

 Ten 2.2 kΩ resistors

 Two 10 kΩ resistors

 Common-anode 7-segment display (GNS 3011 or similar)

 LMC 660 Quad Operational Amplifier

 10kΩ potentiometer

C. **Introduction**

 Circuit 1 below will be constructed using the program from Lab #4 to test for proper operation of the comparator.

 Circuit 2 below will be constructed using a new program written for Lab #7 to implement a successive-approximation ADC.

 See the ***Presentation for Lab #7*** for more detailed background information.

Circuit 1 – Used with program from Lab #4 to test the comparator

Circuit 2 – Used with a new program for Lab #7 to implement a successive-approximation ADC

D. **Pre-Lab Tasks**

1. Show the pinout for the LMC 660 op amp.
2. Use ***Fritzing*** to draw breadboard layout for Circuit 1(no schematic is required).
	* You should be able to simply add on to your Lab #4 breadboard layout.
	* Use the generic IC part in the Core Library, change the number of pins, and label it as the LMC 660.
	* Save your breadboard layout (.fzz file)
	* Export the breadboard as a jpg image to insert in your lab report.
3. Repeat step 2 for Circuit 2.
4. Write a C++ program for the Arduino UNO to use a ***binary search*** to implement a ***successive-approximation ADC*** where:
* The 3-bit binary guess will be output to an R-2R ladder network using the function ***R2R( )*** from Lab 4.
* The value of the binary guess will be displayed on a 7-segment display using the function ***display\_digit()*** from Labs 3-4.
* No buttons are required, so the functions for debouncing button inputs are not needed.
* No output to the computer monitor is required (although you can use it for debugging if necessary).
* An outline of the program might look as follows:

**// Lots of comments (as usual)!**

**void setup( )**

**{ // Define input and output pins**

**}**

**void loop( )**

**{ // Assign initial values to upper, lower and N**

 **// guess = (upper + lower)/2**

 **// for i = 1 to N**

 **{ // Call R2R( ) to send guess to output pins**

 **// Read comparatorInput on D10**

 **if(comparatorInput == 1) // guess > analog input**

 **upper = guess**

 **else**

 **lower = guess**

 **// guess = (upper + lower)/2**

 **}**

 **// display guess on 7-segment display**

**}**

**void display\_digit(int N)**

**{ // Use code from Lab #3 (do not make any changes)**

**}**

**void R2R(int N)**

**{ // Use code from Lab #4**

**}**

E. **In-Lab Tasks**

1. **Build Circuit 1**
	* Build Circuit 1. Most of the circuit should already be in place from Lab #4. Add the additional parts for the comparator circuit (note the unusual pins for the supply voltage on the LMC 660).
	* The circuit should match your Fritzing breadboard layout.
	* Use neat wiring practices.
2. **Testing Circuit 1**
	* Download and run the program from Lab 4.
	* Verify that the display increments from 0 to 7 as the button is pressed and that the DAC (R2R ladder network) still outputs the 8 correct voltage values measured earlier in Lab 4. Record these 8 DAC voltages in the table shown below.
	* Connect a voltmeter to the output of the comparator and another voltmeter to the analog input (center pin on the potentiometer). The comparator should be around 5V when the analog input is greater than the DAC output from the R-2R ladder network and should be around 0V otherwise. Test it and complete the table shown below. Note that the table is easily completed by columns: set the analog input value and then press the button to record the corresponding 8 outputs from the comparator.

|  |  |  |
| --- | --- | --- |
|  |  | Comparator Voltage |
| DisplayValue | DACVoltage | Analog Input0.3V | Analog Input0.9V | Analog Input1.5V | Analog Input2.2V | Analog Input2.8V | Analog Input3.4V | Analog Input4.0V | Analog Input4.9V |
| 0 |  |  |  |  |  |  |  |  |  |
| 1 |  |  |  |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |  |  |  |
| 4 |  |  |  |  |  |  |  |  |  |
| 5 |  |  |  |  |  |  |  |  |  |
| 6 |  |  |  |  |  |  |  |  |  |
| 7 |  |  |  |  |  |  |  |  |  |

* + Demonstrate the program to the instructor .
	+ Record comments for any problems encountered or lessons learned.

(continued on the next page)

1. **Build Circuit 2**
	* Build Circuit 2. The button switch from Circuit 1 can be removed or left in place as it won’t be used anyway. Add a wire connecting the output of the comparator to D10 as indicated.
2. **Testing Circuit 2**
	* Download and run the new program for Lab 7.
	* If the successive-approximation ADC is operating properly, the 7-segment display should vary from 0 to 7 (the digital output) as the potentiometer is turned (analog input varied from 0 to 5V).
	* Once the ADC is working properly, carefully record the range of voltages that are converted to each digital value. For example, the digit 1 might be displayed for input voltages in the range 0.625 – 1.250 V.

|  |  |
| --- | --- |
| Analog Input Voltage Range (V) | Digital Output on 7-segment display |
|  | 0 |
|  | 1 |
|  | 2 |
|  | 3 |
|  | 4 |
|  | 5 |
|  | 6 |
|  | 7 |

* + Demonstrate the program to the instructor .
	+ Record comments for any problems encountered or lessons learned.

F. **Post-Lab Tasks**

1. Discuss the performance of the comparator in Circuit 1. Did it operate as expected in all cases?
2. Discuss the performance of the successive-approximation DAC. Were the voltage ranges for each digital output as expected?
3. Create a stair-step graph of digital output (shown on 7-segment display) versus the analog input range.
4. What is the resolution of the 3-bit DAC built in lab (in volts and as a percentage)?
5. The DAC constructed and tested in lab was a 3-bit DAC. If you were to build a 6-bit DAC:
	1. What hardware changes would be required?
	2. What software changes would be required?
	3. What would the resolution of the DAC be (in volts and as a percentage)?
6. In Lab 8 we will use one of the built-in DACs on the Arduino UNO, which is also a successive-approximation DAC. How many bits are used (search online for the answer if you don’t know)? What is the resolution (in volts and as a percentage)?

G. **Report**

 A lab report is due 1 week after the date of the experiment. Use the same format as in previous labs.