ME 586: Lab 6 Report
Analog to Digital and Vice-Versa

Harsh Savla & TJ Wiegman

2022-10-25

Abstract

In this lab, we created C programs for the STM32 microcontroller and tested them on a STM32F100RB. These programs aimed to allow practicing conversion to and from analog signals. An external digital-to-analog converter (DAC) was used to build a software-controlled analog-to-digital converter (ADC) with external circuitry. Additionally, the internal DAC and ADC that are integrated in the STM32F100RB system were characterized and used for several tasks as well.

1 Lab Checkpoints

1.1 Using an External DAC

Overview

The first task was to construct a circuit that connected the external digital-to-analog (DAC) chip, a DAC0808 integrated circuit on a breakout board that included all necessary supporting circuitry, to the STM32's digital output pins. This allowed testing both parallel digital output from the STM32 board, as well as confirming the DAC's functionality.

Procedure

The DAC0808 breakout board was plugged into the breadboard, and jumper wires were used to connect the pins as given in Table 1. A photograph of the setup is shown in Figure 1. The code used to test different input bytes is given in Appendix A.2.1. Voltage measurements were recorded after each keypress, corresponding to increases of 16 in the number encoded to the DAC.

Results and Discussion

The measured voltage at each input byte value, as well as the difference from the ideal voltage (if the voltage output were perfectly linear from exactly 0 V to exactly 5 V) are given in Figures 2 and 3. The DAC0808 has very linear output, with negligible nonlinearity. The output offset, as shown in Figure 3, stayed within 50 mV, which is less than 1% error from the ideal case. With a quantization error of less than 10 mV, this was quite a fine DAC.

1.2 Construction of Comparator

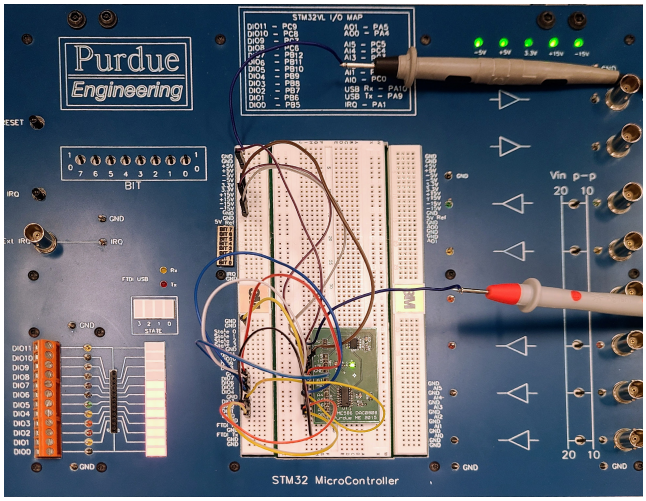
Overview

The next task was to construct a comparator with a LM311P comparator chip and a 1kΩ pull-up resistor.

Table 1: How the ME586 DAC0808 breakout board was connected to the STM32 system.

Pin	Connection
Vo	Multimeter
GND	Ground
+15	+15V
-15	-15V
+5V	+5V
GND	Ground
A1	DIO7
A2	DIO6
A3	DIO5
A4	DIO4
A5	DIO3
A6	DIO2
A7	DIO1
A8	DIO0

Figure 1: A photograph of the ME586 DAC0808 breakout board as it was connected to the STM32 system.



Procedure

The comparator was built according to the wiring diagram given in Figure 4, and output of the power supply was set to 2.5 V. The voltages for logic HIGH and logic LOW were measured both (a) without pull-up resistor (b) with pull-up resistor.

Results and Discussion

By carefully following the wiring diagram, the comparator circuit was created successfully. It was observed that while the pull-up resister was connected, the voltages for logic HIGH and LOW were nearly 5 V and 0 V, respectively, as expected. Once the pull-up resistor was disconnected, the measured voltage for logic LOW continued to be about 0 V, but there was not a stable voltage for logic HIGH. As there was no pull-up resistor, the output pin (Pin 7) was left floating and gave no definite voltage.

1.3 Building a Discrete ADC

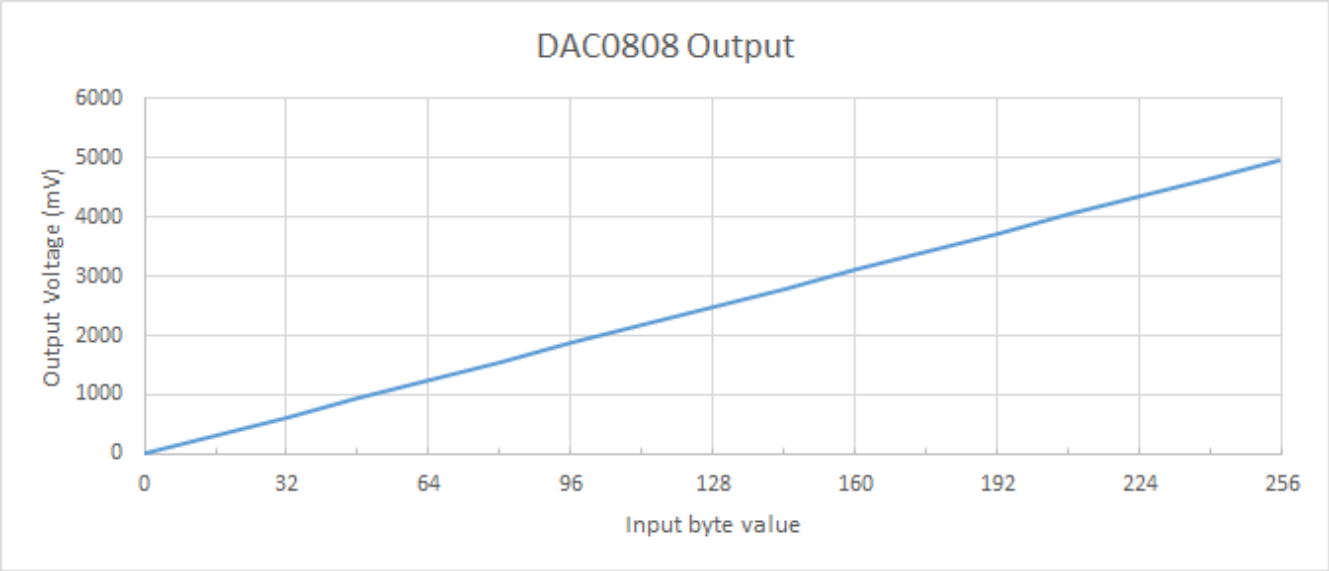
Overview

The third task was to construct an ADC using the DAC from task 1 and the comparator from task 2.

Procedure

The ADC was constructed by connecting the DAC from task 1 to one input of the comparator, and the unknown analog voltage to the other input. The output of the comparator was pulled up to 5 V as in the previous section, and a digital input pin on the STM32 was used to record whenever the DAC output exceeded the unknown voltage. To verify this circuit, first a DC voltage ranging from 0 V to 5 V was applied and corresponding digital values were noted. Then, a sine wave from the function generator was used as the analog input. The ADC measurement signals were compared to the actual input sine signals on an oscilloscope.

Figure 2: The measured DAC voltage as a function of input number.



Results and Discussion

The ADC was successfully constructed using the external DAC and comparator from previous tasks. The ADC was tested by using it to measure sinusoidal signals at several frequencies, and the results were observed on the oscilloscope as shown in Figure 5.

By those various inputs it was concluded that the ADC converts analog signals to digital values more accurately at lower frequencies, as its sampling time is fairly slow. Additionally, lower input frequencies meant that the few milliseconds of latency in the measurement signal were less noticeable.

1.4 Using the Internal DAC and ADC

Overview

The final task was to use the DAC and ADC that are integrated into the STM32F100RB board. The program prepared for this task was very simple, as the flowchart given in Figure 8 shows: it reads an input analog signal with the ADC, then it outputs the same value with the DAC.

Procedure

First, the DAC output range was measured in order to determine the minimum and maximum output voltages. Once recorded, a sine wave was created with a function generator used as the input analog signal for the `analog_copy` function given in Appendix A.2.3. The DAC output waveforms were then compared to the actual input sine signals with an oscilloscope.

Results and Discussion

It was observed that the STM32F100RB board's internal DAC could output voltages ranging from $-10\,\mathrm{V}$ to $10\,\mathrm{V}$. This was the same as the range of analog voltages that the internal ADC could measure, which meant that the next phase of this task could continue.

As shown in Figure 6, the internal ADC and DAC have much faster sampling times than the more crude external ADC constructed for the previous task. This meant that the internal systems were able to give much more accurate measurements of signals at higher frequencies. However, even these had their limits, as a signal at $1\,\mathrm{kHz}$ proved too fast for this system to effectively measure.

Figure 3: The difference between measured DAC voltage and ideal output voltage.

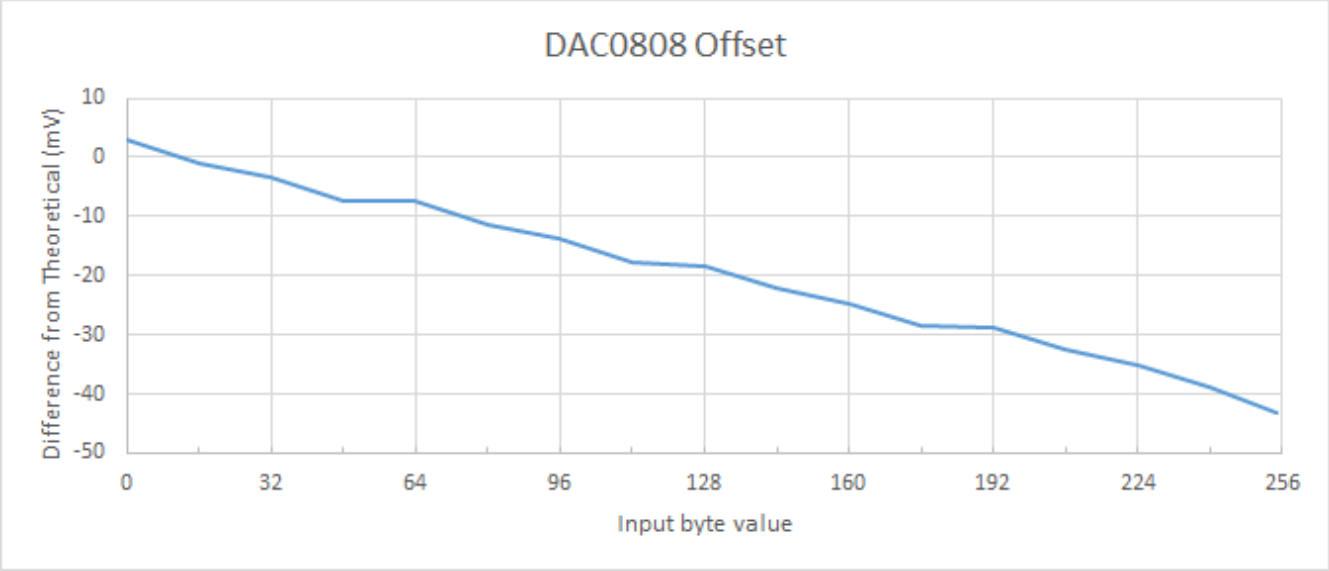


Figure 4: LM311P Comparator Chip wiring diagram

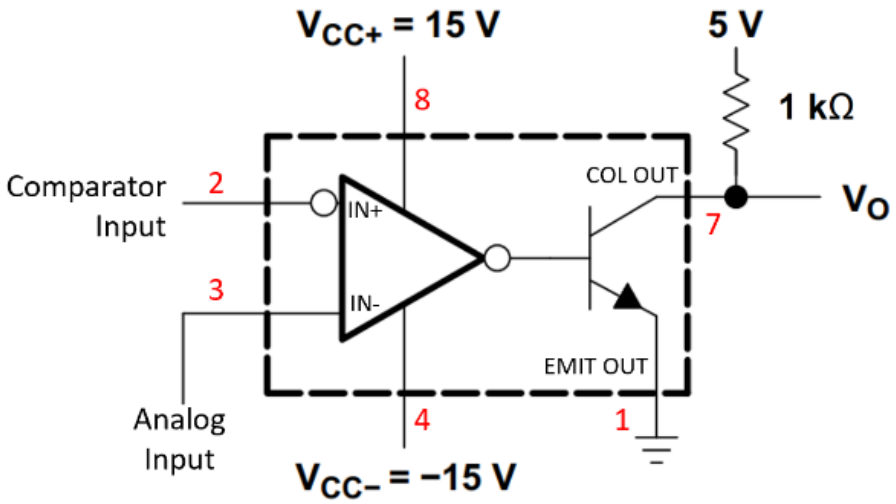
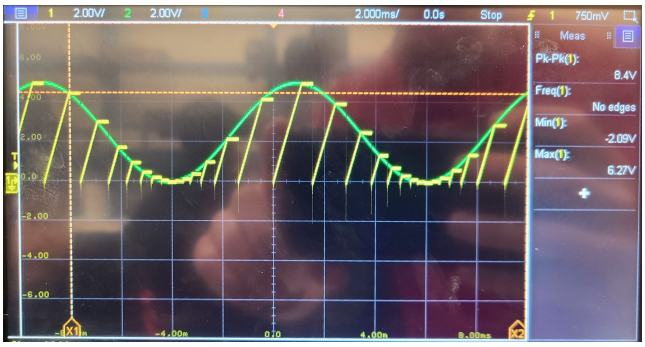
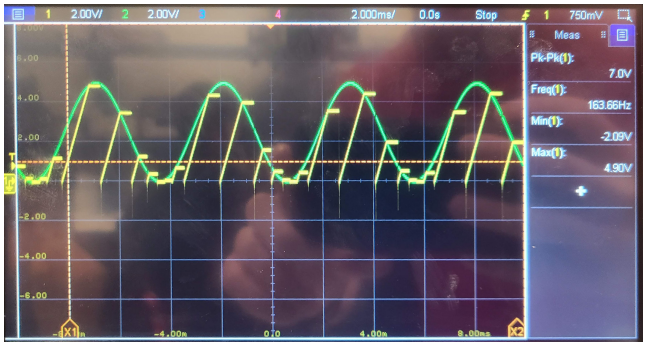


Figure 5: Comparing the counting ADC, as coded in Appendix A.2.2, with input sine signals.

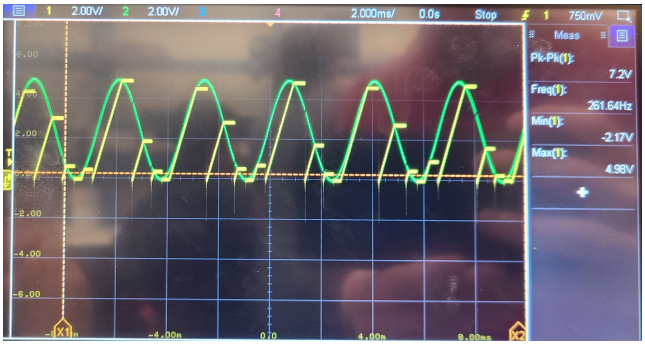
(a) 100 Hz



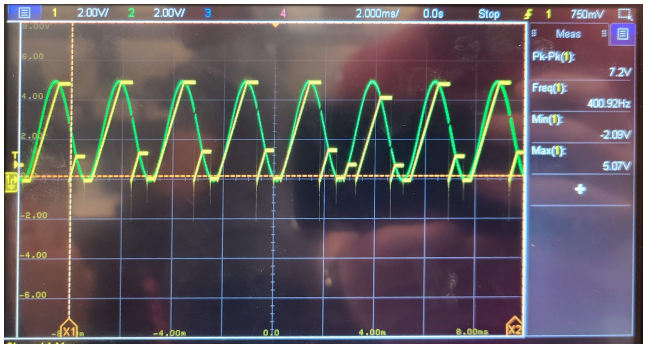
(b) 200 Hz



(c) 300 Hz



(d) 400 Hz



(e) 500 Hz

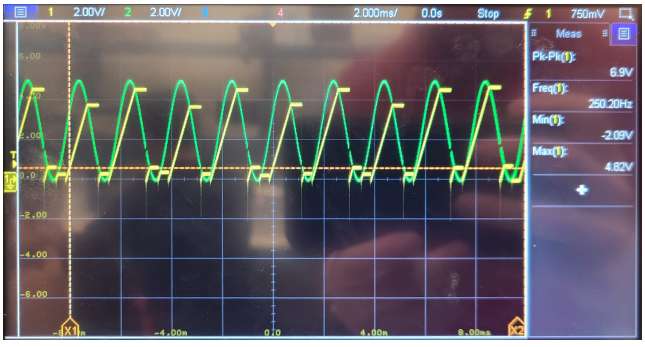
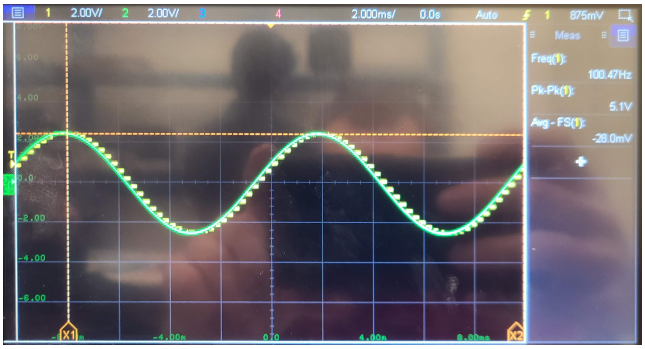
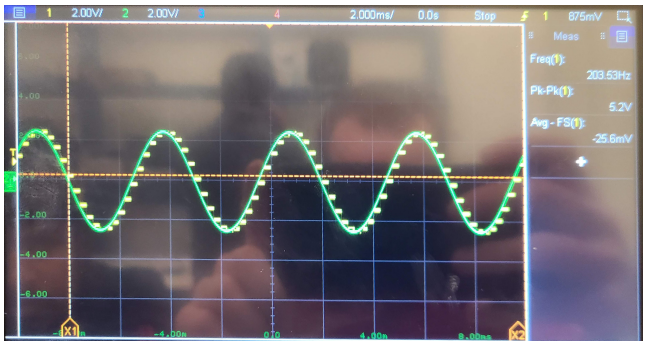


Figure 6: Comparing the internal ADC measurements (through DAC output), as coded in Appendix A.2.3, with input sine signals. At high frequencies such as 1000 Hz, the sampling time is too slow to accurately track the sine curves.

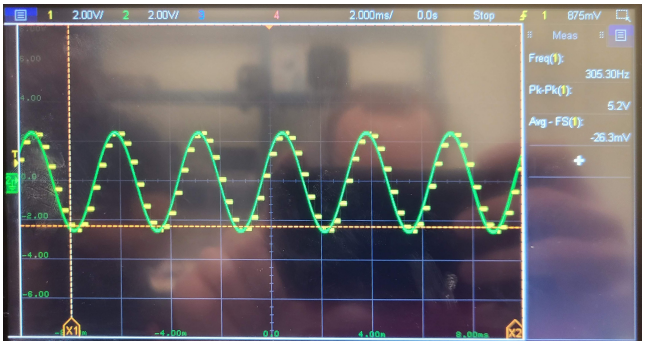
(a) 100 Hz



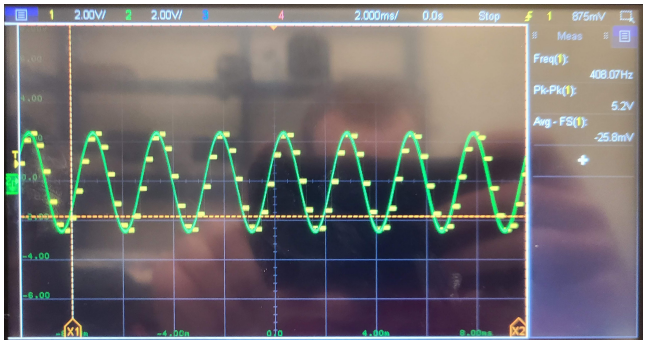
(b) 200 Hz



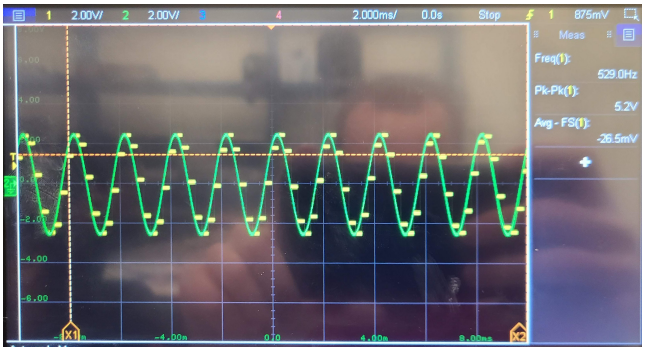
(c) 300 Hz



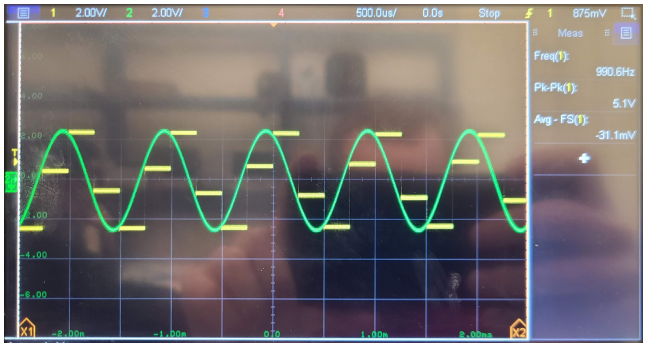
(d) 400 Hz



(e) 500 Hz



(f) 1000 Hz



2 Conclusion

This lab gave excellent practice working with more realistic circuits, as analog signals are ubiquitous in physically-useful systems. Building the counter ADC from discrete components was quite educational for teaching both logical circuit design as well as providing an opportunity for hands-on prototyping practice. Additionally, the later tasks' use of the internal DAC and ADC functions gave good practice working with specialized microcontroller peripherals that are helpfully abstracted by a header library, which was a welcomed change of pace from the grittier, more low-level approach of manipulating peripheral registers directly.

A Appendix

A.1 Flow Charts

Figure 7: Flowchart for the `counter_ADC` function given in Appendix A.2.2.

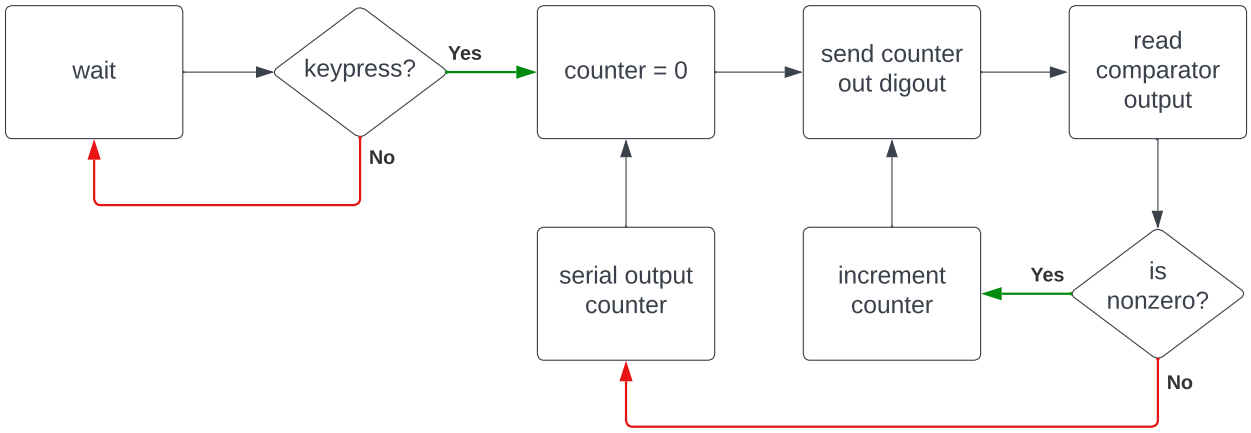
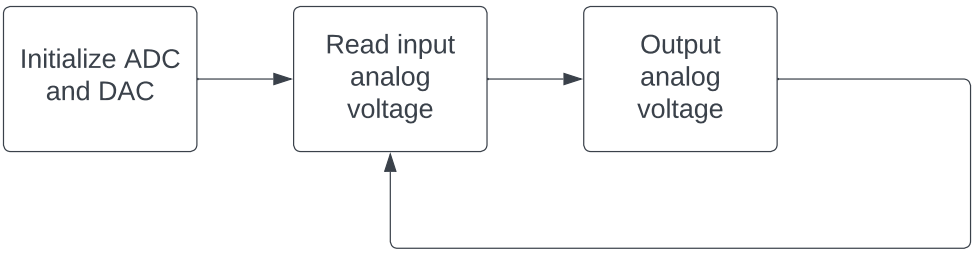


Figure 8: Flowchart for the `analog_copy` function given in Appendix A.2.3.



A.2 Code

A.2.1 dac_calib.c

```
/**
  ******************************************************************************
  * @file dac_calib.c for ME586 Lab 6
  * @author Harsh Savla & TJ Wiegman
  * @version V1.0
  * @date 2022-10-17
  * @brief Output increase internal DAC output by 16 counts with each keypress
  ******************************************************************************
  */

#include "ME586.h"
extern void WaitForKeypress();
extern void initports(int);
extern short digin();
extern void digout(short);
extern void shownum(int);

void dac_calib() {
       short counter=0;
       initports(0xF00);
       initcom();
   while (1) {
       WaitForKeypress();
       counter = counter + 16;
              if(counter>255){
                  counter=255;
              }
              digout(counter);
              shownum(counter);
   }
}
```

A.2.2 counter_ADC.c

```
/**
  ******************************************************************************
  * @file counter_ADC.c for ME586 Homework 6
  * @author Harsh Savla & TJ Wiegman
  * @version V1.0
  * @date 2022-10-17
  * @brief Software-driven ADC based on an external DAC and comparator
  ******************************************************************************
  */

#include "ME586.h"

extern void WaitForKeypress();
extern void initports(int);
extern short digin();
extern void digout(short);
extern void shownum(int);
void counter_ADC() {
    short counter;
    short comparator;
    char nonzero = 1;

    // expect serial communications to be initialized already
    WaitForKeypress();
    initports(0xF00); // PB5-12 are outputs, PC6-9 are an inputs

    while (1) {
        // Start counting from zero
        counter = 0;
        while (nonzero == 1) {
            // Output current counter value and then check comparator
            digout(counter);
            comparator = digin();

            // If comparator has been triggered, then stop counting
            if (comparator < 0x100) {
                nonzero = 0;

            // Otherwise increase the counter and try again
            } else if (counter < 0xFF) {
                counter = counter + 1;

            // If counter reaches 0xFF, then stop counting
            } else {
                nonzero = 0;
            }
        }
        shownum(counter);
        putchar(0x0A);
        putchar(0x0D);
        nonzero = 1;
    }
}
```

A.2.3 analog_copy.c

```
/**
  ******************************************************************************
  * @file analog_copy.c for ME586 Homework 6
  * @author Harsh Savla & TJ Wiegman
  * @version V1.0
  * @date 2022-10-17
  * @brief Read analog input, output the same voltage on analog output
  ******************************************************************************
  */

#include "ME586.h"

extern void initadc();
extern void initdac();
extern int a_to_d(int);
extern void d_to_a(int, int);
void analog_copy() {
    // Setup
    int input;
    initadc();
    initdac();

    // Read and write analog signals
    while (1) {
        input = a_to_d(1);
        d_to_a(0, input);
    }
}
```