Digital Sampling Oscilloscope

MAE 511 - Experimental Methods I

Andreas W. Rousing, Anthony J. Savas, Ben A. Reimold

511: Experimental Methods I
Princeton University
May 20, 2016

Contents

Contents ii

1 Digital Sampling Oscilloscope Design **1**
1.1 Front End: Variable Gain and Offset . . . 2
1.2 Sample and Hold . . . 2
1.3 Counter/Sample Clock . . . 3
1.4 ADC . . . 4
1.5 Memory . . . 5
1.6 DACs . . . 6
1.7 Display . . . 6

2 Programmed Data Display/Teensy **9**

3 Conclusion **12**

4 Damped Oscillator Circuit **13**

A Schematic - Damped Oscillator Circuit **15**

B Teensy/LabVIEW Code **21**

C Orcad **24**

Introduction

The purpose of an oscilloscope is to display an electrical signal as it varies in time. Oscilloscopes are useful for a wide variety of applications, as they are able to display many characteristics of electronic systems, such as noise, frequencies, or voltages. While a voltmeter is able to display a time-averaged voltage value, an oscilloscope can display how electronic parameters of a system change in time. As such, an oscilloscope has to be adjustable to a wide variety of frequencies and signal amplitudes. It needs to be able to sample rapidly changing signals and display them in a manner which is resolvable by the human eye.

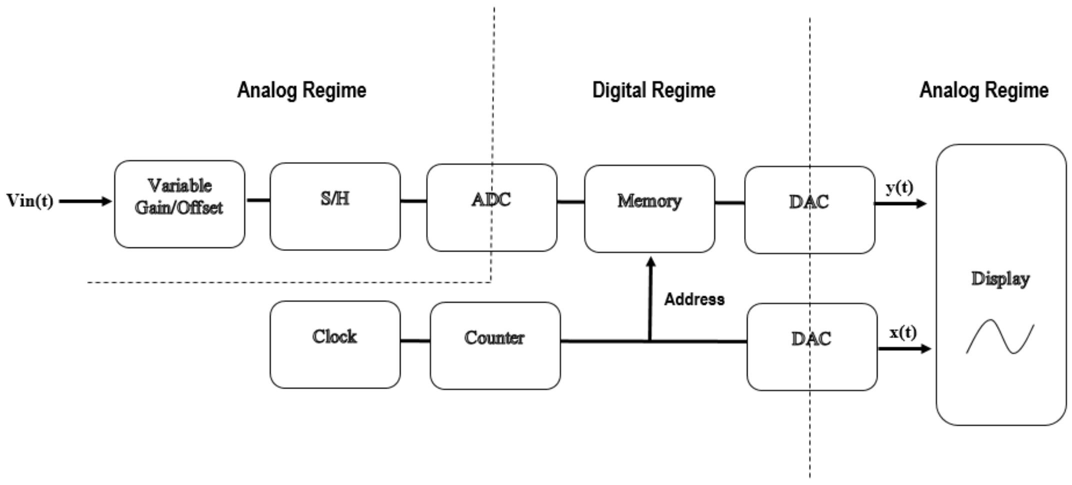


Figure 1: Block Diagram of Oscilloscope

The basic subsystems of an oscilloscope are summarized in Figure 1. The oscilloscope can be divided into two sections, an "analog world" and a "digital world". An analog input signal is processed and prepared for sampling by the variable gain and offset subsystem. The sample and hold subsystem temporarily holds the input signal long enough for the analog to digital converter (ADC) to transform the analog signal into a digital one which can be processed by the digital logic systems. The ADC takes an analog voltage between 0 and 5 volts and sends a digital serial output to a shift register. The shift register parallelizes the signal, which is then latched by the latch until the user requests it. When requested by the user, the captured signal is written to static random-access memory (SRAM) and sent to a digital to analog converter (DAC) which converts the digital signal back to an analog one which is then amplified so that it can be displayed using an analog display. The SRAM addresses are also processed by a separate DAC which are used as a surrogate timescale for the voltage signal. The analog display plots the voltage signal against the timescale signal, providing a snapshot of the behavior of the electronic signal under investigation.

1 Digital Sampling Oscilloscope Design

In the following sections we will give an overview of each subsystem of our circuit as well as how these various subsystems work together to produce the final output of our oscilloscope.

1.1 Front End: Variable Gain and Offset

The input stage of the oscilloscope allows the user to "tune" the input signal by adjusting its gain and DC offset (see Figure 21 in Chapter C of the Appendix). The operational amplifiers (op-amps) used in this stage are railed to ±15 V, meaning that the oscilloscope can sample signals within that range (otherwise the sampled signals will be railed). The ability to offset the incoming signal by a set amount or to adjust its gain is a critical capability of an oscilloscope as it allows the user to manipulate the data and display it in a meaningful way.

There is an initial voltage follower which accepts the inputted signal and outputs an identical signal at its output. We have chosen to include this voltage follower as a way to isolate the oscilloscope circuit from any potential circuit that it may be connected to, with the initial voltage follower giving our circuit a large input impedance. To add in the DC offset we adjust the wiper of a 1 kΩ potentiometer which has its other two terminals connected to +5 V and -5 V, respectively. As such our oscilloscope can afford a DC offset within ±5 V. To add in this offset to the inputted signal we send both the output of the wiper and the output of the initial voltage follower into an inverting summing amplifier (both through separate resistors). At this stage we do not want to introduce any gain with this inverting amplifier, so the two resistors are chosen to be identical to the feedback resistor of the inverting amplifier (1.2 kΩ).

The next stage allows the user to vary the gain of the inputted signal. To do this we employ another inverting amplifier. The resistor leading into the inverting input of this op-amp was chosen to be 510 Ω and we have a variable feedback resistor which can be set between 0-9.7 kΩ. Thus we can calculate the variable gain of our oscilloscope as follows:

$$V_{out} = -\frac{R_f}{R_{in}} V_{in} \tag{1}$$

Equation 1 gives the relationship between the input voltage to the inverting amplifier (V_{in}) and the output voltage (V_{out}) as a function of the resistor leading into the inverting input (R_{in}) and the feedback resistor (R_f). We can see that the quantity R_f/R_{in} equals the magnitude of the gain, and with $R_{in} = 510\ \Omega$ and $R_f = 0 - 9.7\ \text{k}\Omega$ we see that our gain can vary between 0 and roughly 19. The output of this amplifier is then sent to the sample and hold stage.

Note: the sample and hold circuit (to be discussed next) will clip any signals to 0-5 V, so while the user has offset and gain control, it is necessary to keep the tuned signal to within 0-5 V at this stage, otherwise the signal will be clipped by our sample and hold circuitry.

1.2 Sample and Hold

The ADC needs the inputted signal to be held constant while it is performing the conversion so that the signal is accurately digitized. Therefore a sample and hold circuit was implemented in our circuit using two op-amps (voltage followers) along with a capacitor and a bilateral switch (see Figure 21 in Chapter C of the Appendix). The purpose of the initial voltage follower is to isolate the switch from the variable gain and offset circuitry discussed previously. The switch, among other things, allows us to clip the incoming signal within desired voltage bands, which is necessary as our ADC can only accept signals ranging from -0.3 V to V$_{CC}$ + 0.3 V, with a maximum bound on V_{CC} of 6.5 V. The switch has a drain voltage, V_{DD}, and a source voltage,

V_{SS}, with the output being clipped between the two. Thus, to meet the voltage specifications we have chosen $V_{SS} = 0$ V and $V_{DD} = +5$ V, meaning that our oscilloscope will be able to output signals within this range.

The opening and closing of the switch is controlled with a clock signal (TTL) from our baud rate generator, with the frequency of the signal determining the "sampling rate" of our sample and hold circuit. The switch is closed when the control voltage is equal to V_{DD} and it opens when this control voltage equals V_{SS}; thus, a high clock signal (+5 V) will close the switch and a low clock signal (0 V) will open it. When the switch is closed the capacitor is subsequently charged and gives us a sampled signal, and when the switch is opened the capacitor holds the sampled voltage for the duration of the low clock cycle. Finally, the second voltage follower is used to keep the capacitor isolated from the remainder of the circuit. See Figure 2 which shows snapshots of signals being sampled and held at various frequencies.

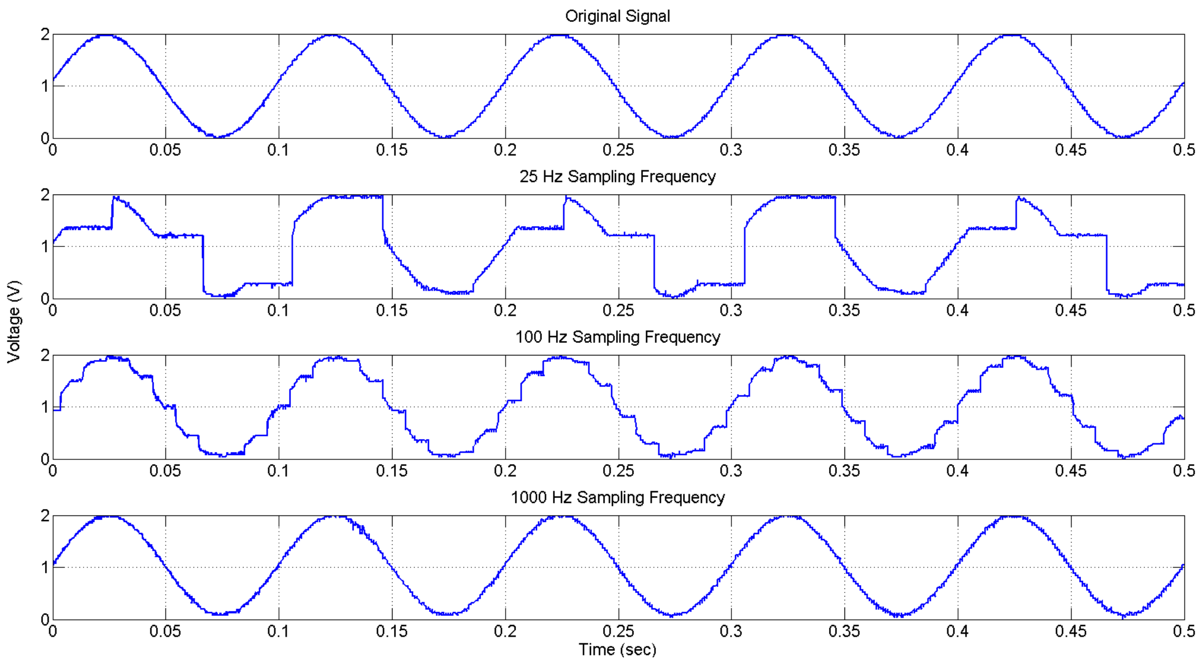


Figure 2: Sample and hold operation for three different sampling frequencies.

1.3 Counter/Sample Clock

The master clock timing used in our circuit is generated by a crystal oscillator with frequency 1 MHz (see Figure 22 in Chapter C of the Appendix). A baud rate generator is then used to divide down the frequency of this signal. One output of this baud rate generator is our master clock at 250 kHz while the other is our sampling frequency used in the sample and hold circuit (nominally 62.5 kHz but can be manually adjusted). The master clock is used to clock the ADC, but it is also critical to provide the ADC with a "chip select" signal which provides the timing for the actual conversion to take place. When the chip select signal is low the ADC will output data, and since our ADC is 8-bit we must ensure that chip select is low for 8 counts of the master clock; however, in reality we actually need chip select to be low for 10 counts of the master clock as the data bits at the beginning and end of the conversion window are considered to be invalid. In this way we will have 8 good bits of data in the end. To accomplish this we need to create a modulo-10 version of the master clock and then divide the frequency of this modulated signal by two. To create the modulo-10 signal we used a 14040 12-bit counter along with an AND gate. The counter was clocked with the master clock and we fed bits 1 and 3 of the counter into the inputs of the AND gate, whose output was then connected to the reset pin of the counter. Thus, when we reach $2^1 + 2^3 = 10$ the counter is forced to reset. The output of the AND gate is incidentally a modulo-10 version of the master clock, so to get the final frequency division by two we send this modulo-10 signal as the clock to a 74LS74 flip-flop. By connecting the D and $\overline{Q}$ pins of this flip-flop together in a feedback loop we get a frequency divider, and the signal outputted at the Q pin of the flip-flop will be at one-half the frequency of the modulo-10 signal. In this way we can generate the appropriate chip select signal. See Figure 3 for an illustration of these timing signals.

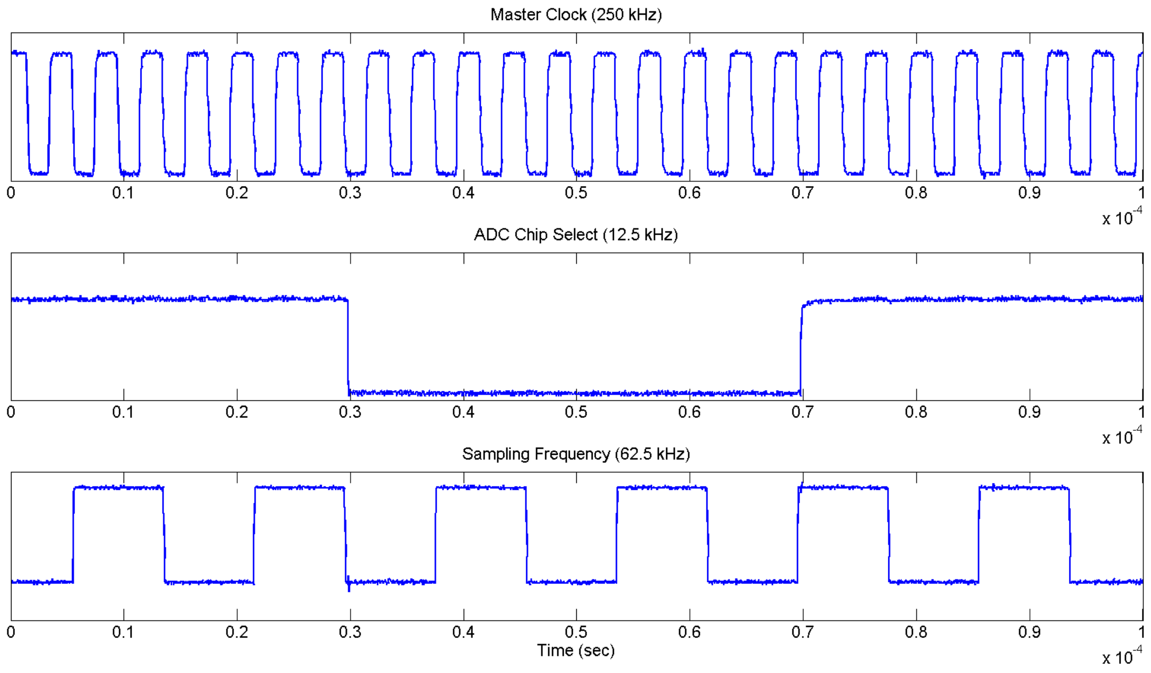


Figure 3: Illustration of timing signals.

1.4 ADC

The purpose of the ADC is to receive the analog signal from the sample and hold circuitry and then convert this signal into an 8-bit binary representation. This is what is meant by "digitizing" the signal. One thing to note is that the ADC we have used (ADC0831) is serial in that it outputs one bit at a time. Because of this we send the output of the ADC to a shift register which acts to parallelize the digital data and which is able to store 8 bits at one time.

The output of the shift register is then sent to a latch which is clocked with the ADC chip select. The latch has an output control signal ($\overline{OC}$) which, when low, causes the latch to output its data. We employ a push button on the trainer board along with logic such that data is output from the latch only when the push button is pressed and the ADC chip select signal is low. The truth table for our latch is shown in Table 1:

INPUTS			**Output**
$\overline{OC}$	CLK	D	Q
L	↑	H	H
L	↑	L	L
L	L	X	Q_0
H	X	X	Z

Table 1: Truth table for the 74LS374N Latch

The output of the latch is then sent into an octal tri-state buffer. The buffer has two separate gate pins which each control four of the eight outputs. When the gate pins are set low the buffer outputs its data. For our operation we link both of these gate pins to the $\overline{OC}$ pin of the latch such that when the latch is outputting data it is subsequently all passing through the buffer.

1.5 Memory

Up to this point we have been discussing how we are measuring the value of the inputted signal (essentially the y-axis); however, to display a full xy-plot of a signal response we also need the x-axis component. We use a 12-bit counter to "count" forward in time, and this data is sent as addresses to the SRAM. Since the SRAM has 11 address inputs we decided to make use of all 11 of them to have the best resolution possible. This is easily accomplished by connecting the first 11 bits of the counter to the 11 address bits of the SRAM and then sending the 12th bit of the counter to its reset pin.

The timing of the SRAM can be controlled in one of two ways, with a write-enable signal ($\overline{WE}$) or with a chip select signal ($\overline{CS}$). A third signal, output enable ($\overline{OE}$), also plays a role in the operation of the SRAM. A summary of the interplay between these signals is shown in Table 2:

Mode	$\overline{CS}$	$\overline{OE}$	$\overline{WE}$	**I/O**
Standby	H	X	X	High-Z
Read	L	L	H	$DATA_{OUT}$
Read	L	H	H	High-Z
Write	L	X	L	$DATA_{IN}$

Table 2: Truth Table for the SRAM

We chose to operate in a $\overline{WE}$-controlled environment by holding $\overline{CS}$ and $\overline{OE}$ low and connecting $\overline{WE}$ to the push button logic. This ensures that when the button is pushed by the user the data is written into the correct port in the memory. When the button is not depressed, the SRAM is in read mode and cycles through the data points stored in the memory.

1.6 DACs

Since we have an 8-bit ADC and 11-bit SRAM, we chose two different DAC chips to handle these different data streams from the memory. Our 12 bit address DAC (DAC1222) cycles through the 11 bits of memory address lines, providing the highest level of address resolution for our given components (see Figure 24 in Chapter C of the Appendix). Furthermore, the signal data is processed by an 8-bit DAC (DAC0832). It is worth noting that our 8-bit DAC can operate in either double-buffered, single-buffered, or flow-through modes. We chose to operate the DAC in the flow-through mode, meaning that its analog output would continuously reflect the state of the digital input. We did this by grounding $\overline{CS}$ (chip select for DAC), $\overline{WR}_1$ (write 1), $\overline{WR}_2$ (write 2), and $\overline{XFER}$ (transfer control signal) and tying ILE (input latch enable) high at +5 V (see Figure 25 in Chapter C of the Appendix). Both the data and address line DACs pass their signals to separate amplifiers which in turn connect to a display unit.

1.7 Display

We have configured our oscilloscope to be able to display on either a Tektronix TDS 2014C oscilloscope screen or in LabVIEW using a Teensy (to be discussed later). To better resolve the address and data signals we captured the raw data points from the Tektronix oscilloscope and exported them to a *.csv file which we were able to plot using Matlab. See Figure 4 below which shows the sampled signal along with the address and data DAC outputs when we are writing to memory.

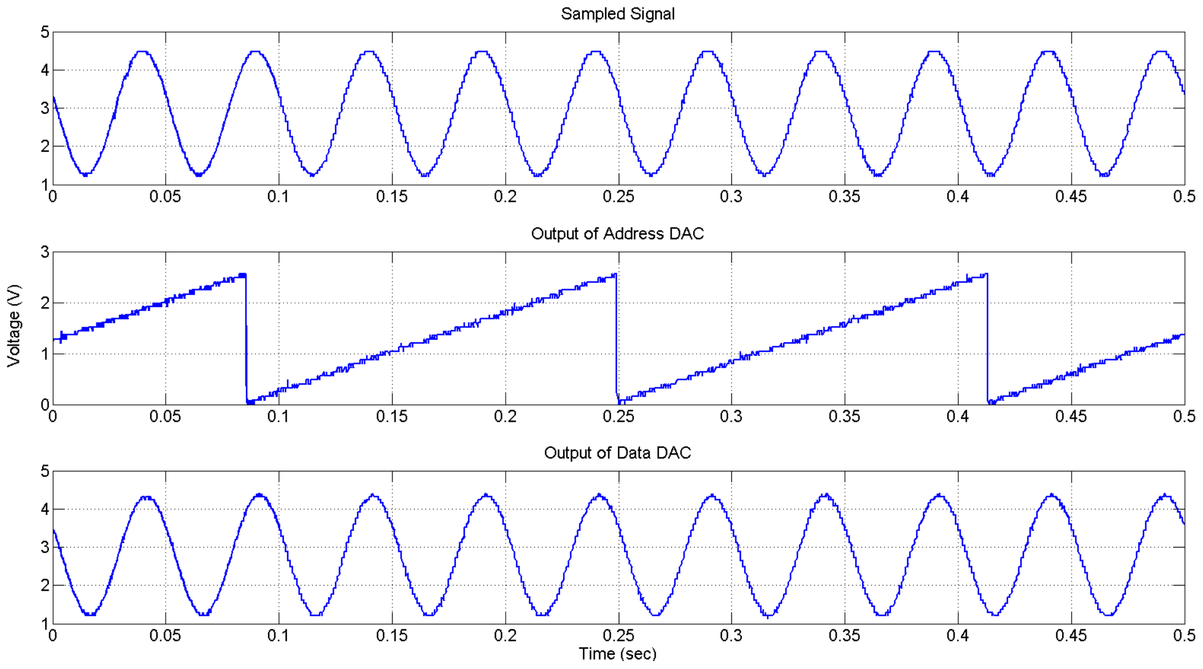


Figure 4: Address and data output in "write" mode.

Our oscilloscope can also save the signal input in SRAM and retrieve it at a later time. See Figure 5 for an example of the oscilloscope operating in read mode.

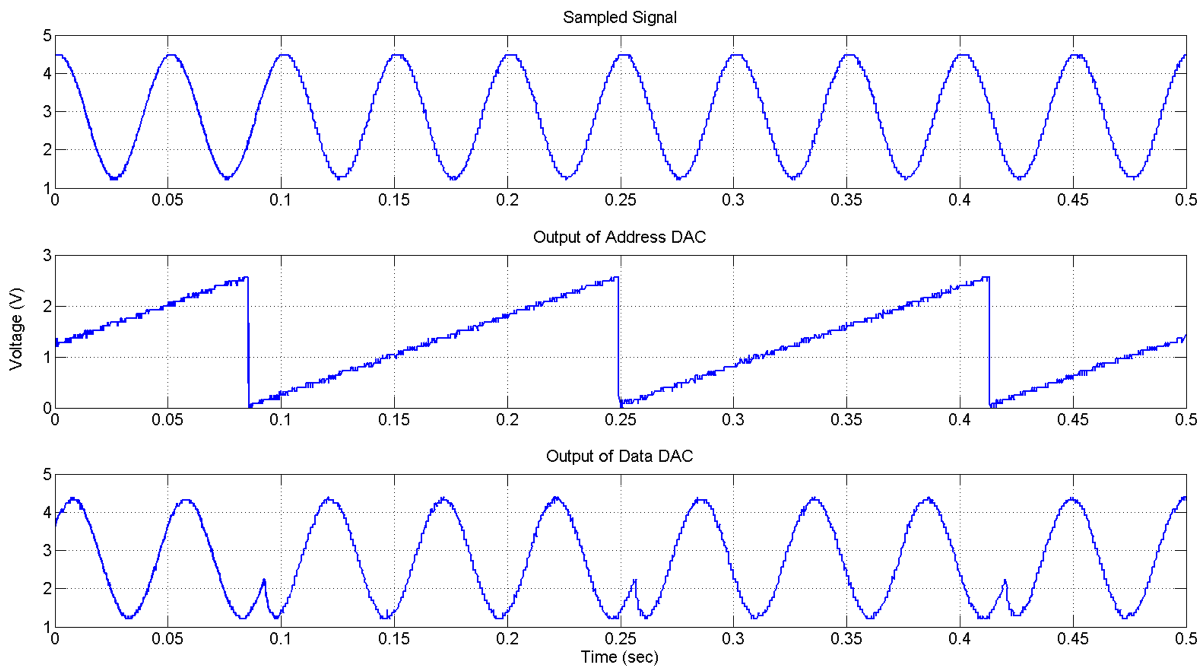


Figure 5: Address and data output in “read” mode.

One thing to notice is that the output of the data DAC is a repeating signal, which of course makes sense when comparing to the address output. Our addresses are counting up in time but we are limited to 11 bits worth of addresses (in the SRAM). This is seen by examining the ramp output of the address DAC in Figures 4 and 5. The ramp is a repeating signal and the SRAM cycles through all of its available addresses during each ramp cycle; thus, we can only capture a signal for the duration of one ramp cycle. This is what gives rise to the discontinuities in Figure 5 - they correspond to points where the ramp resets and the memory limit is reached.

As mentioned previously, the addresses are a surrogate for time, so we would like to plot the output of the data DAC vs. the address output to get a full xy-plot of our sampled data. We have done this in two ways, the first of which is to send the outputs of the address and data DACS to the Tektronix commercial oscilloscope and then just use its XY plotting mode to display the signals against one another. These results for a sample sine wave are displayed in Figure 6:

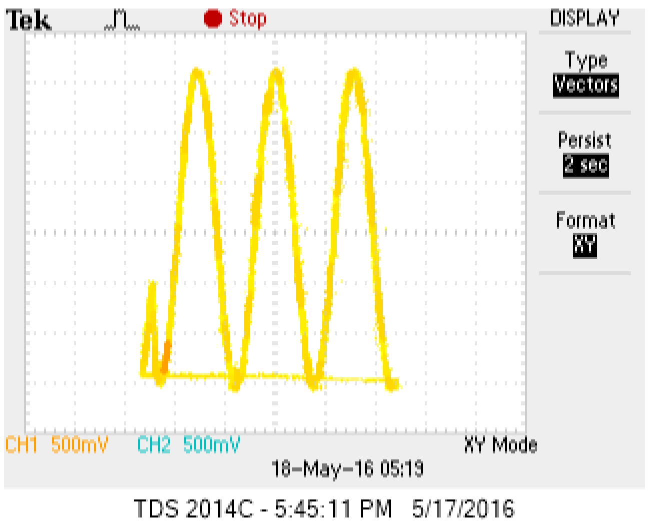


Figure 6: Plotting address output vs. data output directly on a Tektronix TDS 2014C oscilloscope using the XY mode.

We can see that we recover our sampled sinusoid. A second way we have chosen to display this is to export the address and data DAC output to a *.csv file and manually plot the signals against one another in Matlab. The results of this manual plotting are shown in Figure 7.

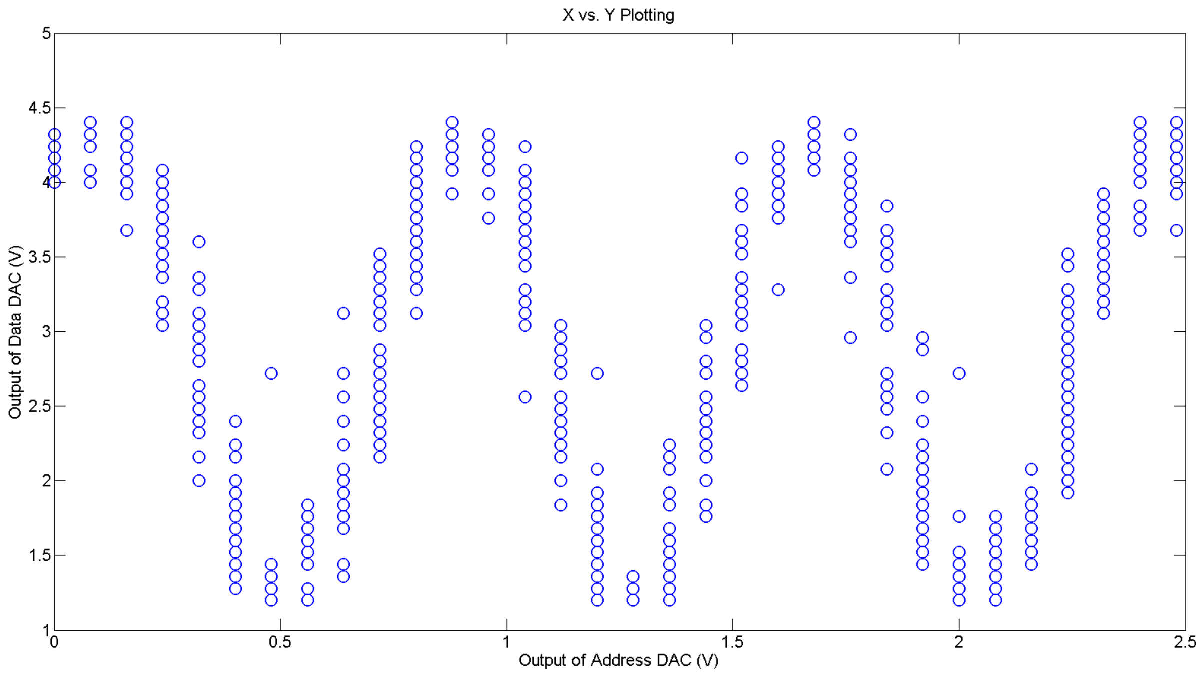


Figure 7: Manually plotting address output vs. data output (XY plot)

Here we can see the resolution of our DACs in that we only have discrete voltage values with which to work with (see how the data is grouped into rows and columns in Figure 7). Nevertheless we are able to recover our sinusoidal signal.

Our final circuit is shown below in Figure 8:

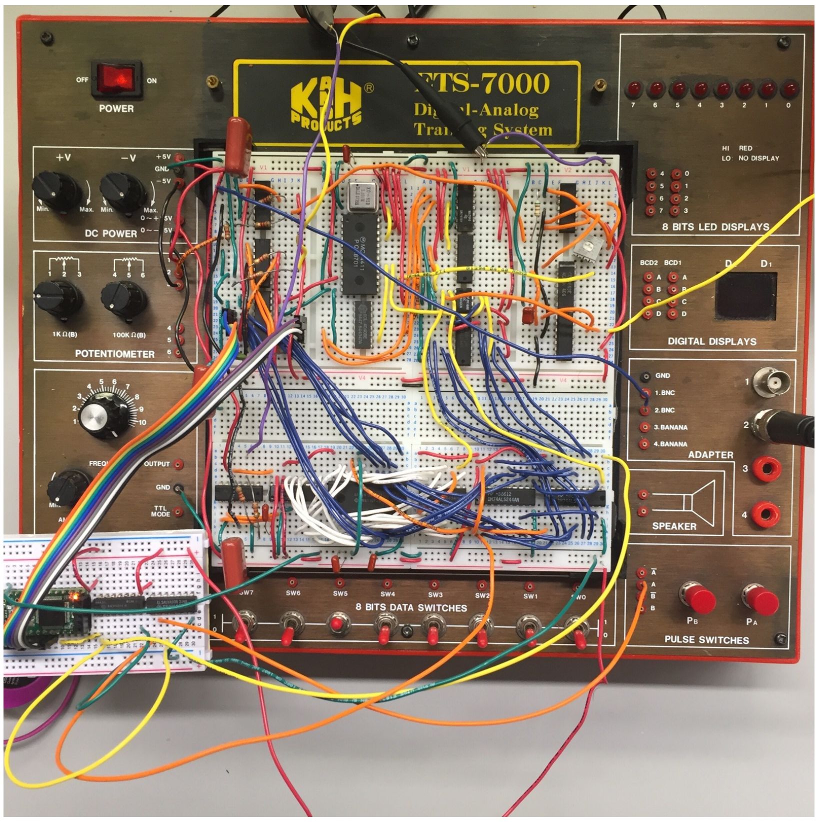


Figure 8: Photo of complete oscilloscope circuit.

The color coding scheme for our circuit is given below in Table 3:

Color	Connection
Red	VCC+, +15V or +5V
Black	VCC-, -15V
Green	Ground
Orange	Inter-chip
Blue	Data
White	Addresses

Table 3: Color Coding for Oscilloscope Circuit

2 Programmed Data Display/Teensy

In order to be able to display the data we use a Teensy 3.1 with several peripherals and a Cortex-M4 96 MHz (overclocked) processor. Since the output data from the ADC and SRAM

is 8-bit we utilize 8 I/O pins on the Teensy programmed to read the individual bits incoming on every pin and combining each data cycle into a byte arranging the bits from MSB to LSB. In order to ensure that every byte is only recorded once we utilize interrupts. Interrupts call functions which usually do not return anything and are executed when a specific event occurs, thus halting all other processes on the microprocessor until the interrupt routine is completed. The interrupt is chosen such that when the I/O Pin 8 on the Teensy detects a rising edge, the interrupt is triggered. The interrupt routine is chosen to be the byte reading sequence described above and the interrupt signal to be the ADC chip select. In this way the Teensy only records a databyte every time 8 new databits (corresponding to one reading) has been outputted which ensures that we only read each datapoint once. Having the Teensy read continously would result in every byte being read multiple times before a new byte would be recieved.
Since the Teensy is programmed with the Arduino program (using the Teensyduino application to interface with the Arduino program), the serial output from the Teensy can simply be outputted on a monitor. The program code for reading the data bytes timed with the interrupt can be seen in Figure 19 in Chapter B of the Appendix. The Teensy can be seen in Figure 9:

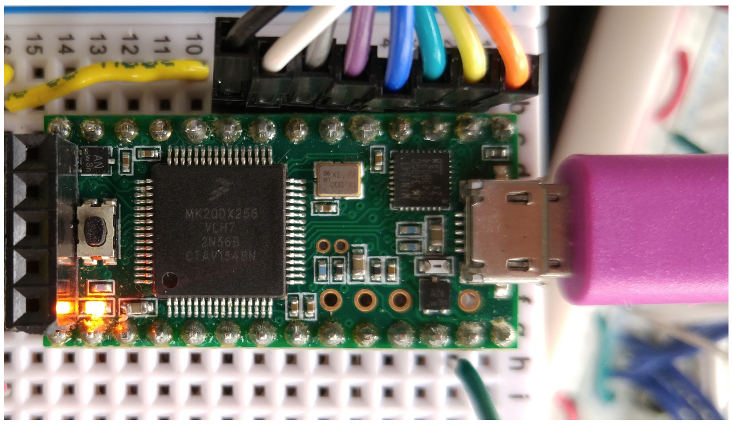
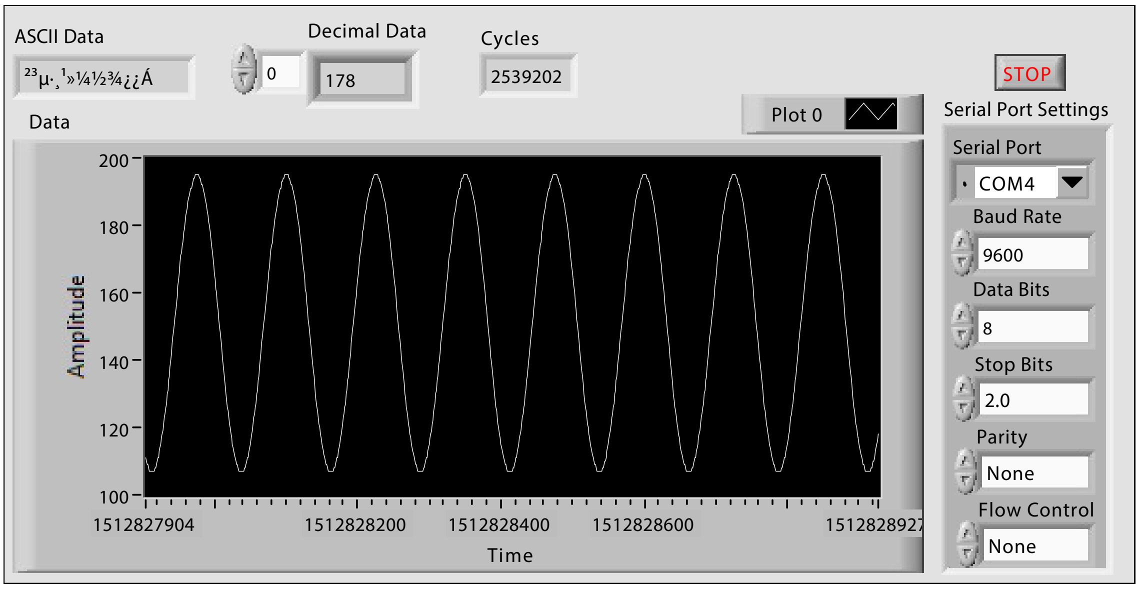


Figure 9: The Teensy 3.1 connected to individual bit outputs (wire row on top) from our oscilloscope, ADC chip select (leftmost yellow wire) and ground

.

However since the goal is to design a digital sampling oscilloscope capable of displaying the data visually, the serial output is fed to a LabVIEW program for data-handling. Using the blocks and connections seen in the VI (Figure 20 in Chapter B of the Appendix), the data is read from the serial output coming from the Teensy and displayed on the GUI front panel, as seen in Figure 10.


ASCII Data: ²³µ·¸¹»¼½¾¿¿Á

Decimal Data: 0 178

Cycles: 2539202

Serial Port Settings

Serial Port: COM4

Baud Rate: 9600

Data Bits: 8

Stop Bits: 2.0

Parity: None

Flow Control: None

Figure 10: Front Panel of the LabVIEW Program

In the ASCII Data display the incoming bytes are displayed in ASCII. The Decimal Data display shows the latest byte in decimal, while the cycles correspond to the number of cycles the loop in the VI has been running in total. The settings for the serial data being read are seen in the right panel. At the top is the Serial Port, which automatically is detected as the Teensy is connected. The Baud Rate is set to 9600 which fits the Baud Rate from the Teensy, and Data Bits are set to 8. Because of the interrupt in the Teensy program every byte is only read once, as mentioned earlier. By inputting a sine wave from the function generator, reading it with the Teensy interrupting at every rising ADC chip select edge, writing to the serial port and finally interpreting and plotting in our LabVIEW program, we get the output seen in the data window on the LabVIEW GUI in Figure 10.

3 Conclusion

We were able to produce a digital sampling oscilloscope with variable gain and offset capabilities which could successfully sample an input signal at adjustable sampling rates. Our oscilloscope was able to write/read the digital data to/from memory and then display the re-analogued signal on either a commercial oscilloscope or in LabVIEW using the Teensy.

Over the course of this project we learned not just about the individual components, but also about how these various parts interact with each other. Because we chose an ADC with serial output, we needed a shift register to convert this signal into a parallel output which could be received by the memory. However, because the memory I/O lines are used for both writing and reading, the input device needs to have a high-Z or tri-state enabled mode when not activated. Because the shift register we selected did not have these modes, the combination of the latch and buffer was added to ensure that the component inputted into the memory was tri-state enabled.

Another lesson learned was the importance of understanding the timing dynamics of our selected components and how the timing signals dictated the interactions between systems in our circuit. Because the ADC we chose did not operate using an end of conversion (EOC) signal output, much of our circuit was dependent on the chip select signal of the ADC. This effectively determined the length of our 8-bit data stream, and therefore the latch timing must also be controlled by this timing signal. When this timing was incorrect our circuit initially performed incorrectly; however, after the timing issues were corrected we were able to generate valid output.

4 Damped Oscillator Circuit

Before the design of the digital sampling oscilloscope was undertaken, a damped oscillator circuit was designed and fabricated. This circuit was initially assembled on a breadboard in order to tweak its functionality to the desired configuration with the desired dampening. The schematic and printed circuit board (PCB) was designed in EAGLE CAD in order to compress the physical design. The schematic can be seen in Figure 13 and the PCB design in Figure 14 and Figure 15, all in Chapter A of the Appendix. The PCB design was milled on a double-sided board using the milling machine *Othermill* developed by Othermachine. In order to make sure that the PCB is compliant with the *Othermill* milling constraints a Design Rule Check (DRC) from Othermachine, *"Othermill EAGLE Design Rules v2, DRC for Othermill, 1/32" tool"* was downloaded and imported into EAGLE. The DRC was executed to check if the board design was compliant with *Othermill* and the chosen tool size. After correcting non-compliant dimensions the .brd file was exported from EAGLE CAD to the *Othermill*. The top-side of the board was milled as seen in Figure 14. After this was completed the board was flipped and reattached according to instructions by Othermachine, after which the bottom-side was milled. The final result with all components soldered can be seen in Figure 11 and Figure 12. The Netlist is shown in Figure 16 and Figure 17 and the Partlist is shown in Figure 18, all in Chapter A of the Appendix.

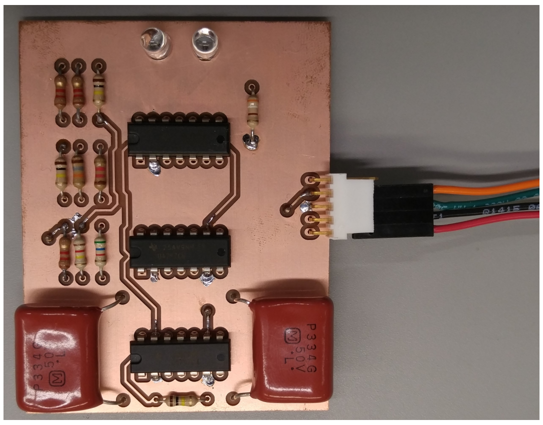


Figure 11: Fabricated PCB frontside

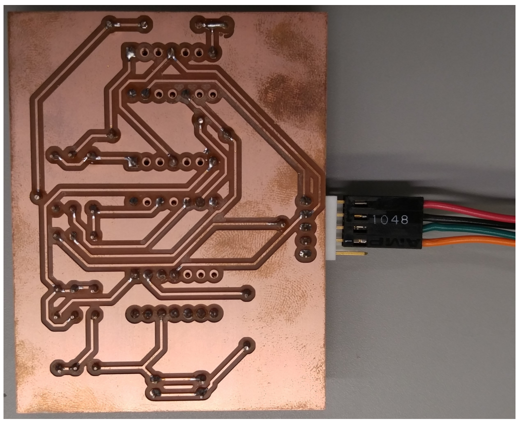


Figure 12: Fabricated PCB backside

The footprints for the capacitors we used did not exist in the EAGLE library which required custom design of these footprints. As can be seen, this presented no issues after measuring the physical dimensions of the components. The wire color-coding scheme for this circuit can be seen in Table 4,

Color	Connection
Red, PIN 1	VCC+, +15V
Black, PIN 2	VCC-, -15V
Green, PIN 3	Ground
Orange, PIN 4	Input Signal

Table 4: Color Coding for the Damped Oscillator PCB

With the schematic and PCB designed and fabricated in the *Othermill* and after all components and vias were soldered the board was tested with a sinusoidal input signal from a function generator. The two diodes lit up as expected, with the magnitude of the light slowly dampening as the input signal damped.

A Schematic - Damped Oscillator Circuit

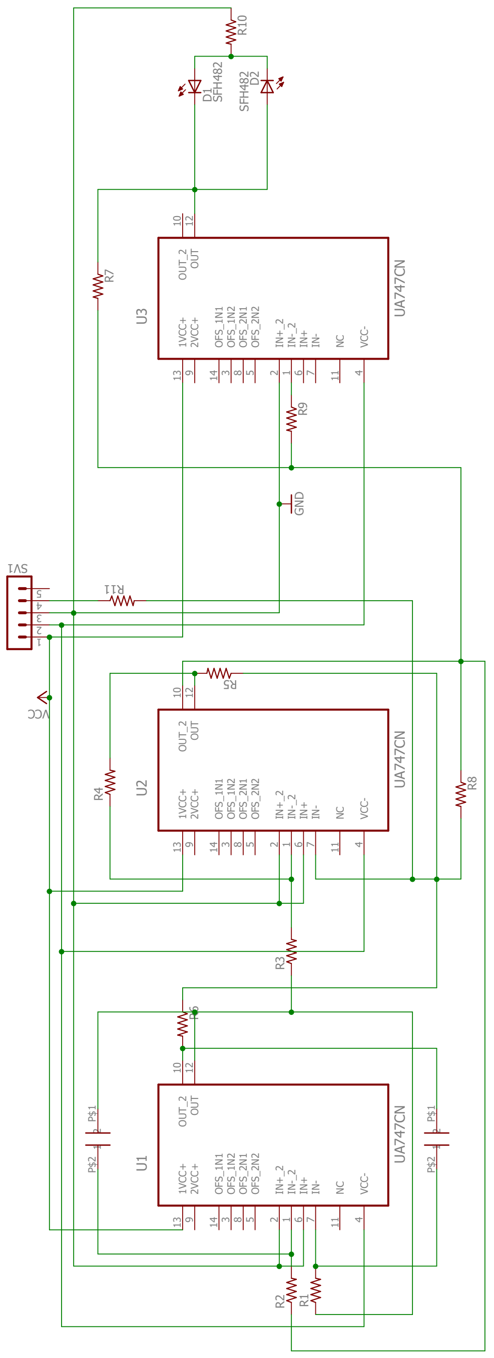


Figure 13: Schematic of the Damped Oscillator Circuit designed in EAGLE CAD

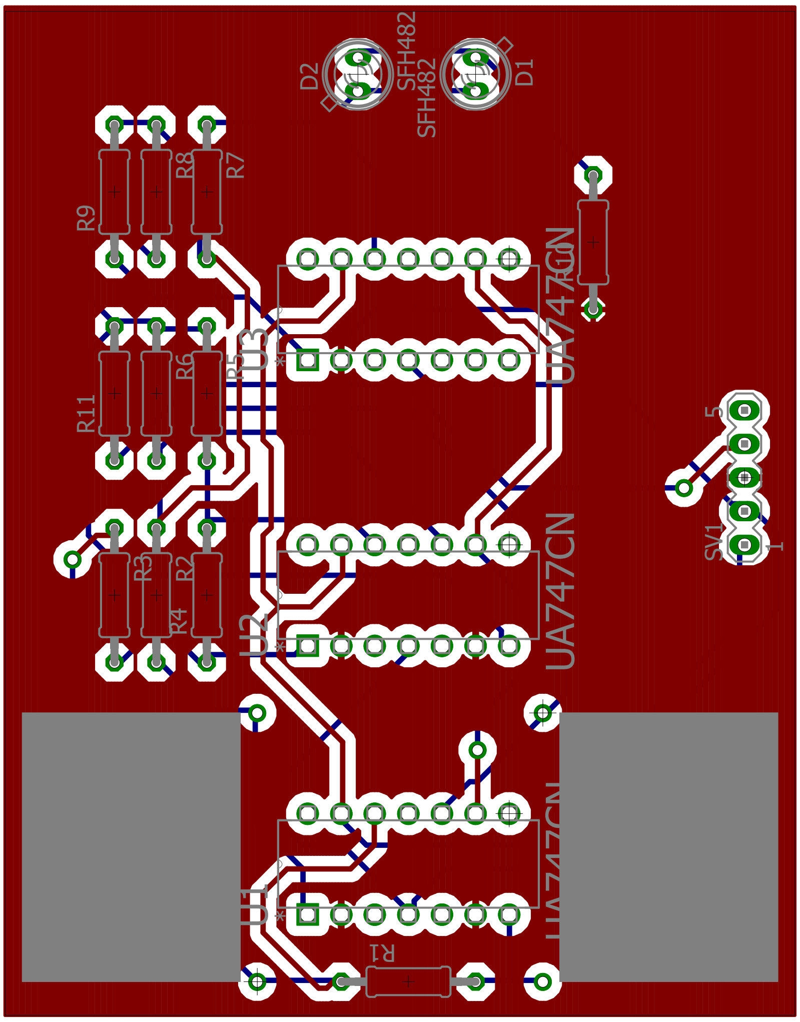


Figure 14: Frontside of the designed PCB

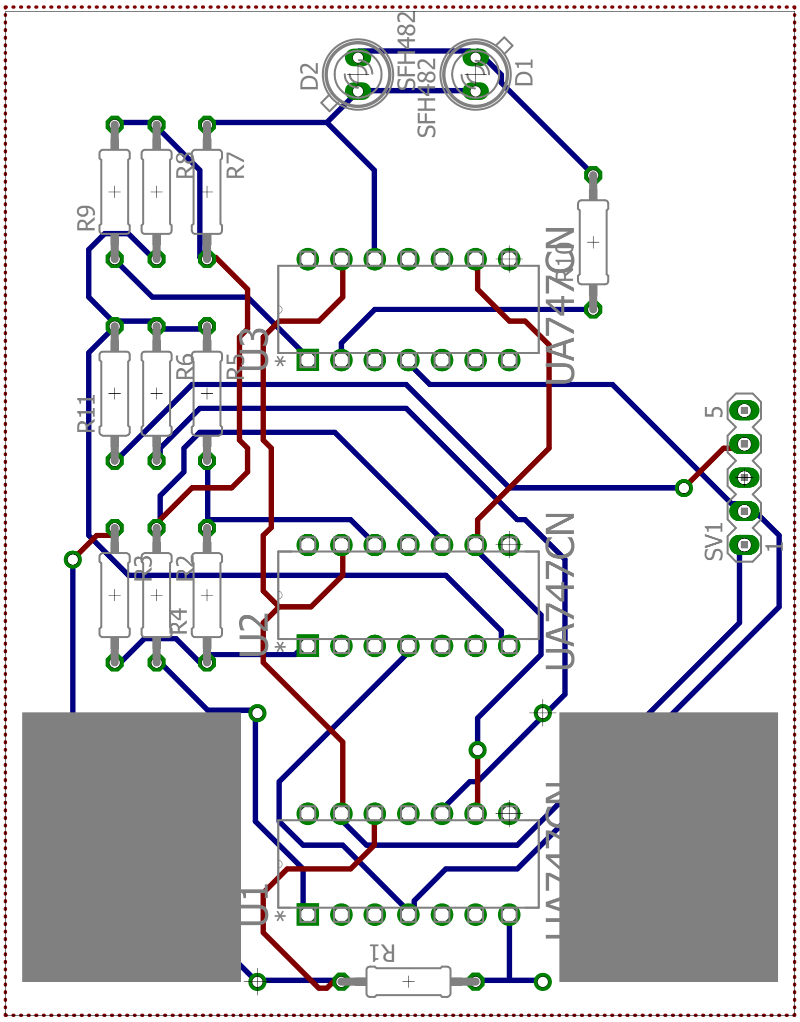
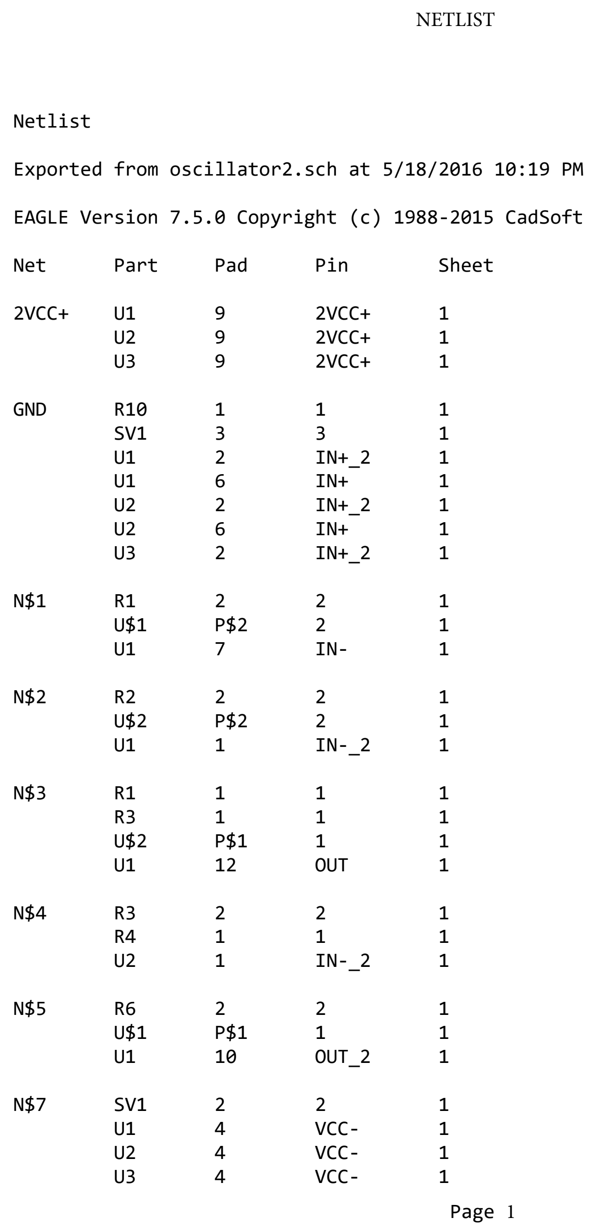


Figure 15: PCB with all traces visible

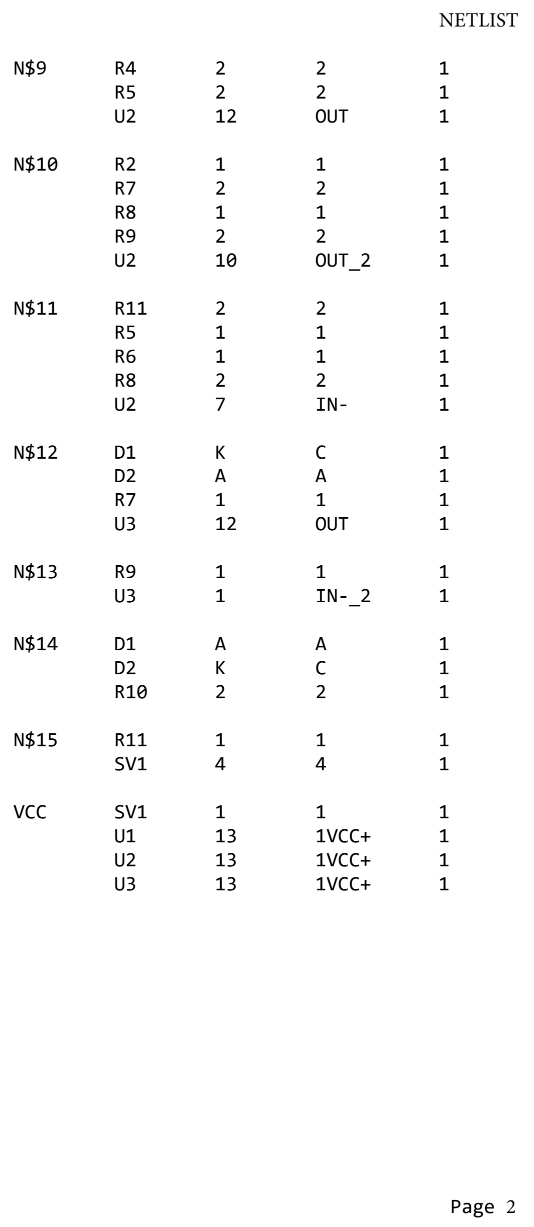


NETLIST

```
Netlist

Exported from oscillator2.sch at 5/18/2016 10:19 PM

EAGLE Version 7.5.0 Copyright (c) 1988-2015 CadSoft

Net      Part     Pad      Pin        Sheet

2VCC+    U1       9        2VCC+      1
         U2       9        2VCC+      1
         U3       9        2VCC+      1

GND      R10      1        1          1
         SV1      3        3          1
         U1       2        IN+_2      1
         U1       6        IN+        1
         U2       2        IN+_2      1
         U2       6        IN+        1
         U3       2        IN+_2      1

N$1      R1       2        2          1
         U$1      P$2      2          1
         U1       7        IN-        1

N$2      R2       2        2          1
         U$2      P$2      2          1
         U1       1        IN-_2      1

N$3      R1       1        1          1
         R3       1        1          1
         U$2      P$1      1          1
         U1       12       OUT        1

N$4      R3       2        2          1
         R4       1        1          1
         U2       1        IN-_2      1

N$5      R6       2        2          1
         U$1      P$1      1          1
         U1       10       OUT_2      1

N$7      SV1      2        2          1
         U1       4        VCC-       1
         U2       4        VCC-       1
         U3       4        VCC-       1
```

Page 1

Figure 16: Netlist for the Damped Oscillator PCB

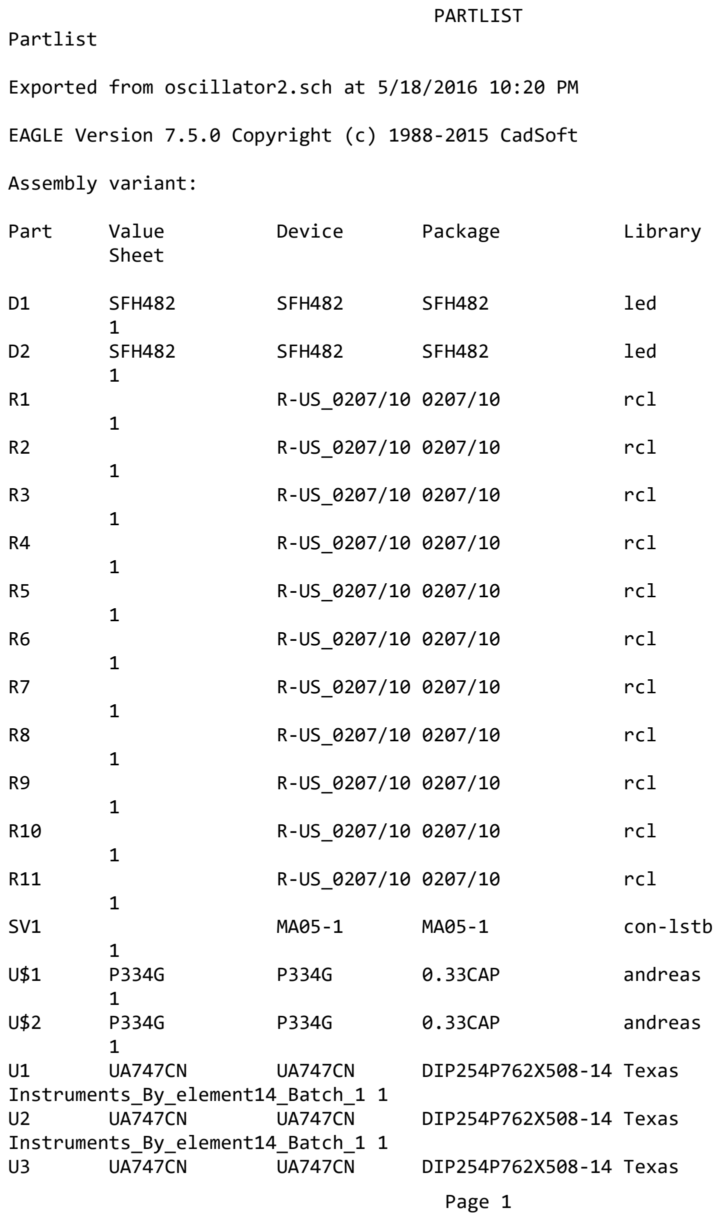


NETLIST

```
N$9     R4    2    2       1
        R5    2    2       1
        U2    12   OUT     1

N$10    R2    1    1       1
        R7    2    2       1
        R8    1    1       1
        R9    2    2       1
        U2    10   OUT_2   1

N$11    R11   2    2       1
        R5    1    1       1
        R6    1    1       1
        R8    2    2       1
        U2    7    IN-     1

N$12    D1    K    C       1
        D2    A    A       1
        R7    1    1       1
        U3    12   OUT     1

N$13    R9    1    1       1
        U3    1    IN-_2   1

N$14    D1    A    A       1
        D2    K    C       1
        R10   2    2       1

N$15    R11   1    1       1
        SV1   4    4       1

VCC     SV1   1    1       1
        U1    13   1VCC+   1
        U2    13   1VCC+   1
        U3    13   1VCC+   1
```

Page 2

Figure 17: Netlist for the Damped Oscillator PCB



```
                                          PARTLIST
Partlist

Exported from oscillator2.sch at 5/18/2016 10:20 PM

EAGLE Version 7.5.0 Copyright (c) 1988-2015 CadSoft

Assembly variant:

Part     Value          Device       Package            Library
         Sheet

D1       SFH482         SFH482       SFH482             led
         1
D2       SFH482         SFH482       SFH482             led
         1
R1                      R-US_0207/10 0207/10            rcl
         1
R2                      R-US_0207/10 0207/10            rcl
         1
R3                      R-US_0207/10 0207/10            rcl
         1
R4                      R-US_0207/10 0207/10            rcl
         1
R5                      R-US_0207/10 0207/10            rcl
         1
R6                      R-US_0207/10 0207/10            rcl
         1
R7                      R-US_0207/10 0207/10            rcl
         1
R8                      R-US_0207/10 0207/10            rcl
         1
R9                      R-US_0207/10 0207/10            rcl
         1
R10                     R-US_0207/10 0207/10            rcl
         1
R11                     R-US_0207/10 0207/10            rcl
         1
SV1                     MA05-1       MA05-1             con-lstb
         1
U$1      P334G          P334G        0.33CAP            andreas
         1
U$2      P334G          P334G        0.33CAP            andreas
         1
U1       UA747CN        UA747CN      DIP254P762X508-14 Texas
Instruments_By_element14_Batch_1 1
U2       UA747CN        UA747CN      DIP254P762X508-14 Texas
Instruments_By_element14_Batch_1 1
U3       UA747CN        UA747CN      DIP254P762X508-14 Texas
                                           Page 1
```

Figure 18: Partslist for the Damped Oscillator PCB

B Teensy/LabVIEW Code

```
//LED
int ledPin = 13;

//Interrupt ADC/Memory Address Update Pin
const int ADCCS = 8;

// Datapins
const int data1 = 0;
const int data2 = 1;
const int data3 = 2;
const int data4 = 3;
const int data5 = 4;
const int data6 = 5;
const int data7 = 6;
const int data8 = 7;

// Databits
boolean databit1 = 0;
boolean databit2 = 0;
boolean databit3 = 0;
boolean databit4 = 0;
boolean databit5 = 0;
boolean databit6 = 0;
boolean databit7 = 0;
boolean databit8 = 0;

byte dataByte = 00000000;
byte newByte;

void setup() {
  // Baud Rate
  Serial.begin(9600);
  // Initializing datapins to be inputs
  pinMode(data1,INPUT);
  pinMode(data2,INPUT);
  pinMode(data3,INPUT);
  pinMode(data4,INPUT);
  pinMode(data5,INPUT);
  pinMode(data6,INPUT);
  pinMode(data7,INPUT);
  pinMode(data8,INPUT);

  //ADCCS Pin Init
  pinMode(ADCCS,INPUT);
  pinMode(ledPin,OUTPUT);

  //Turn on cool LED
  digitalWrite(ledPin,HIGH);

  //Initialize Interrupt for Falling Edge of ADC
  //Chip Select/Address Counter
  attachInterrupt(ADCCS,mode1,FALLING);

}

  void loop() {
  //Nothing to see here
  }
```

```
void mode1(){
  databit1=digitalRead(data1); // LSB
  databit2=digitalRead(data2);
  databit3=digitalRead(data3);
  databit4=digitalRead(data4);
  databit5=digitalRead(data5);
  databit6=digitalRead(data6);
  databit7=digitalRead(data7);
  databit8=digitalRead(data8); // MSB

  bitWrite(dataByte,0,databit1); //LSB
  bitWrite(dataByte,1,databit2);
  bitWrite(dataByte,2,databit3);
  bitWrite(dataByte,3,databit4);
  bitWrite(dataByte,4,databit5);
  bitWrite(dataByte,5,databit6);
  bitWrite(dataByte,6,databit7);
  bitWrite(dataByte,7,databit8); //MSB

  Serial.write(dataByte);
 //Write the byte to
 Serial
 }
```

Figure 19: Program code for the Teensy in order to read the data from ADC or RAM

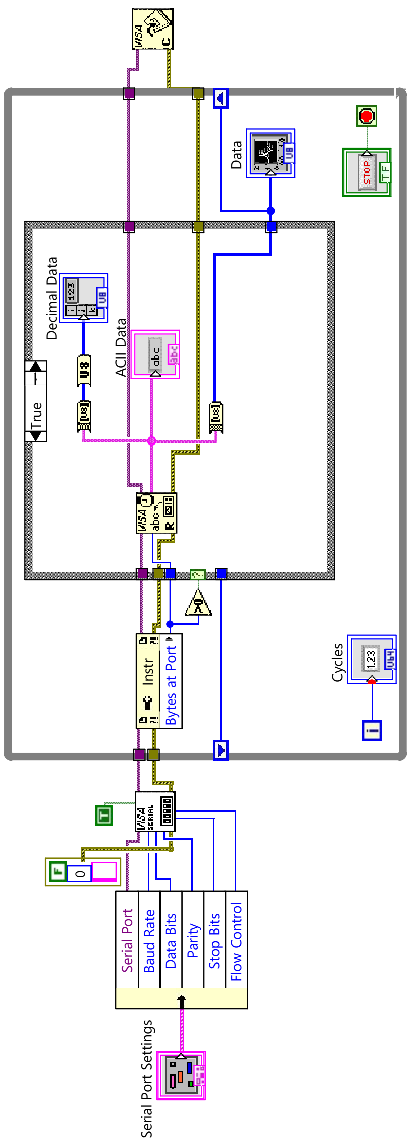


Figure 20: LabVIEW VI which interprets and reads the data from the Serial output from the Teensy

C Orcad

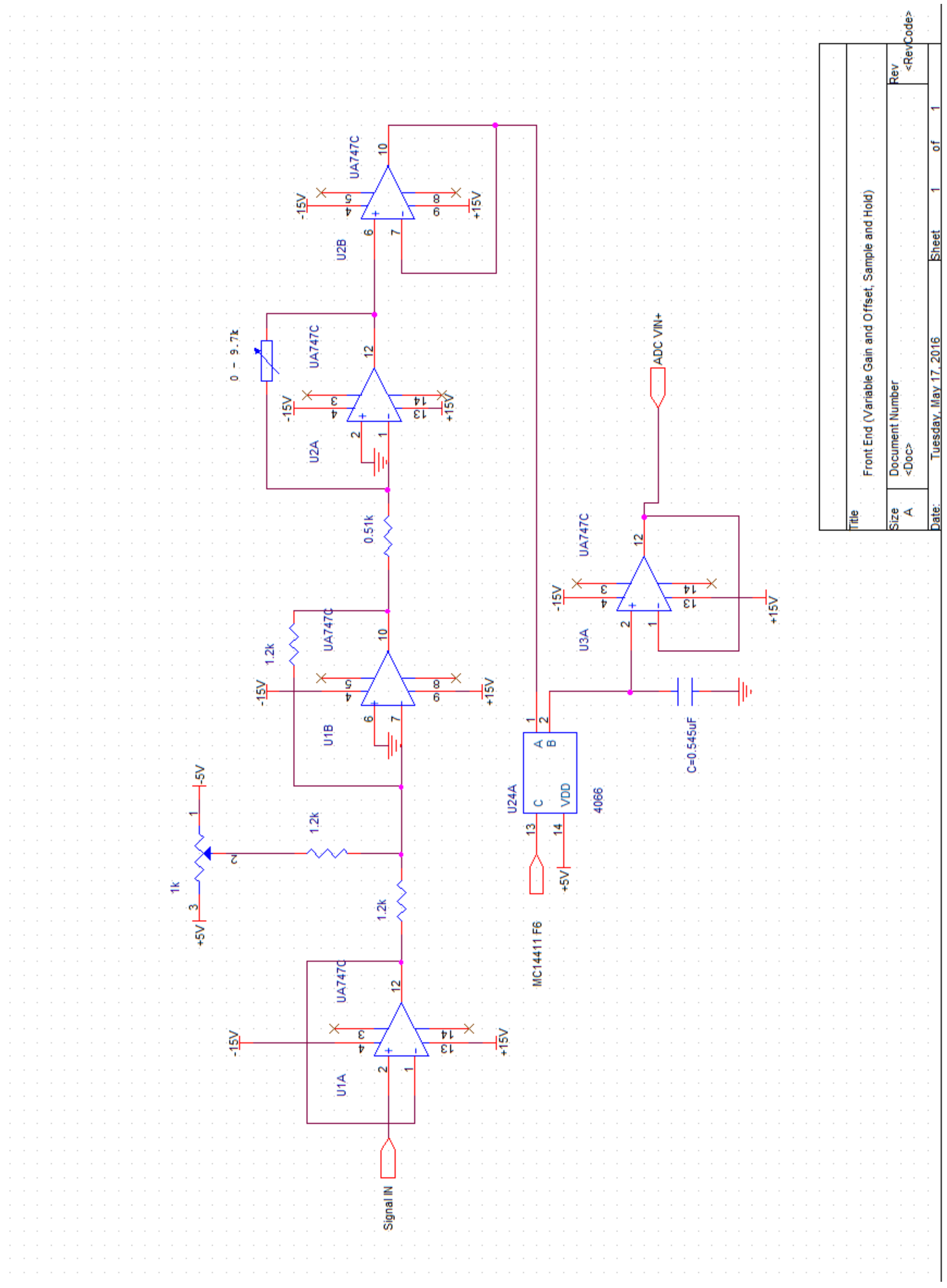


Figure 21: Schematic of front end components (variable gain/offset, sample and hold)

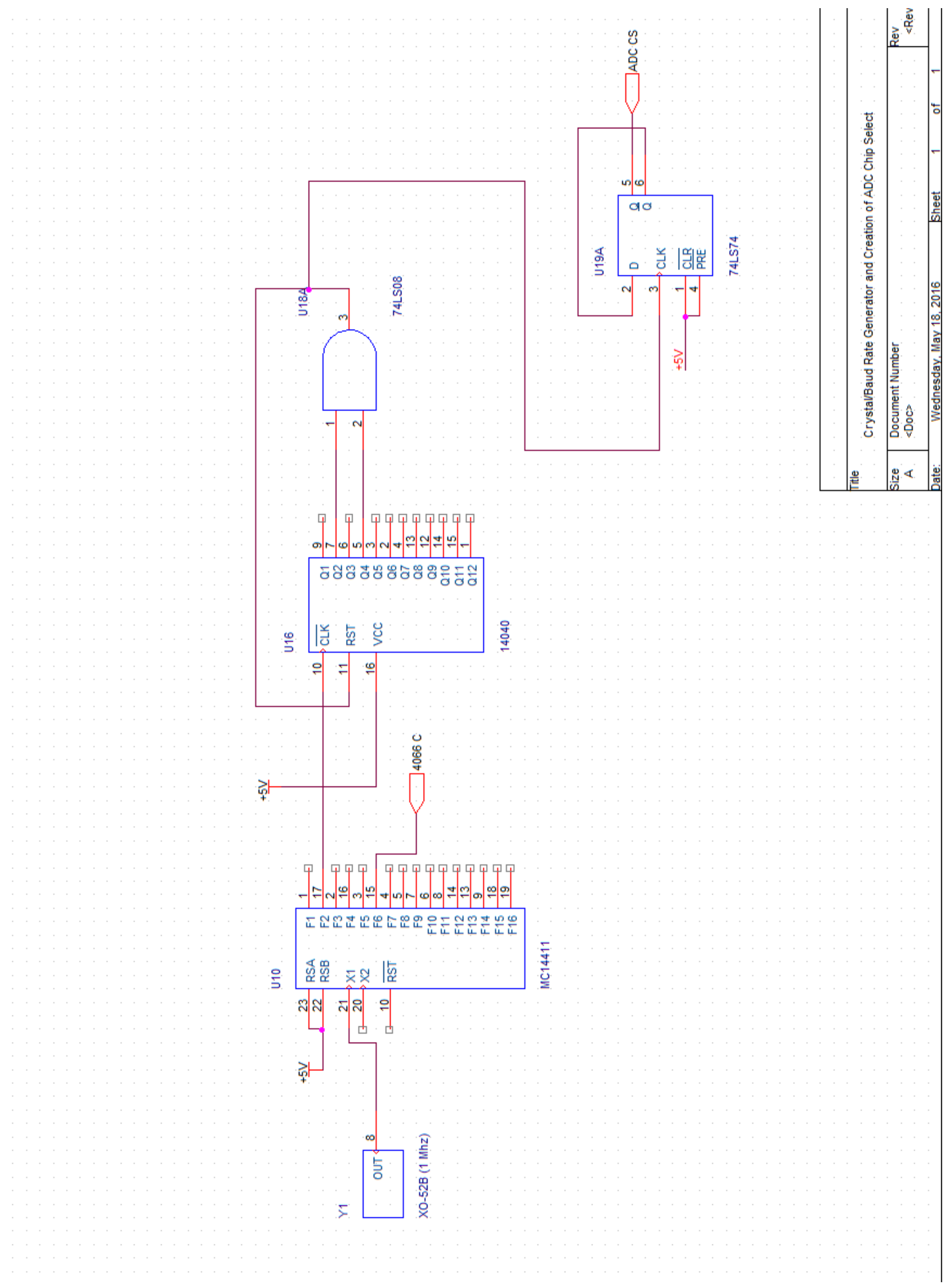


Figure 22: Schematic of timing components (crystal oscillator, baud rate generator)

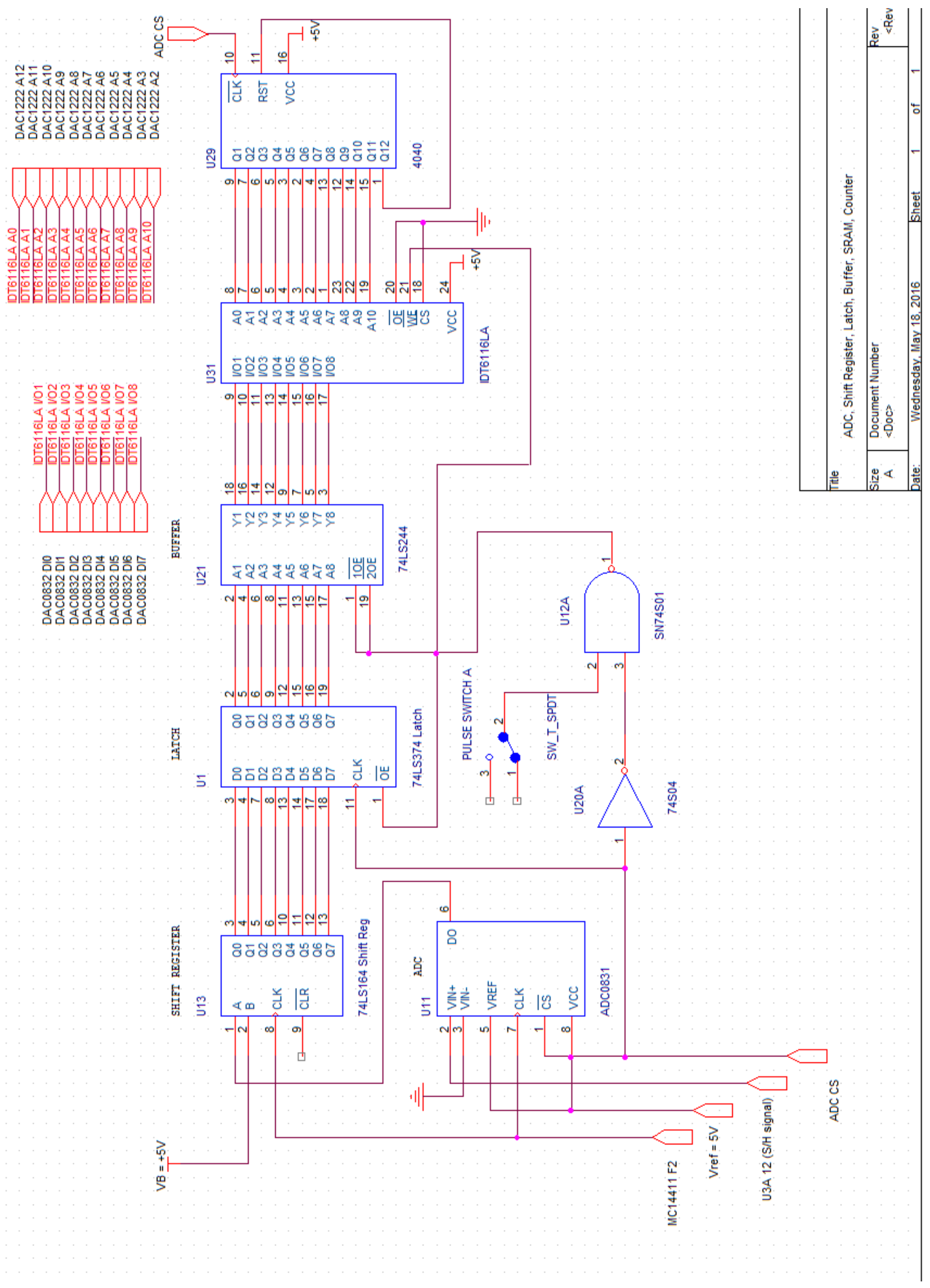


Figure 23: Schematic of ADC, shift register, latch, buffer, SRAM, and address counter.

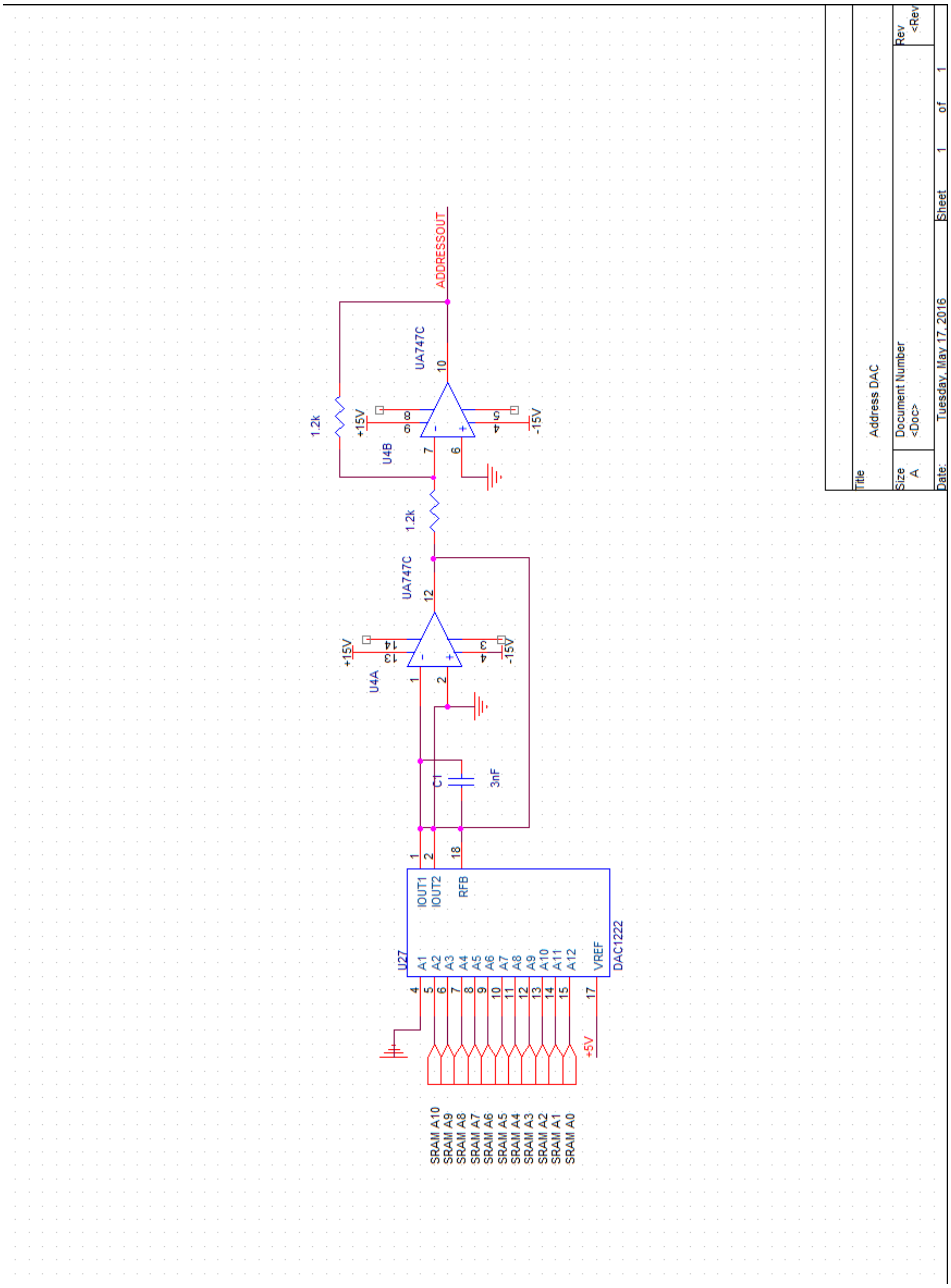


Figure 24: Schematic of address DAC.

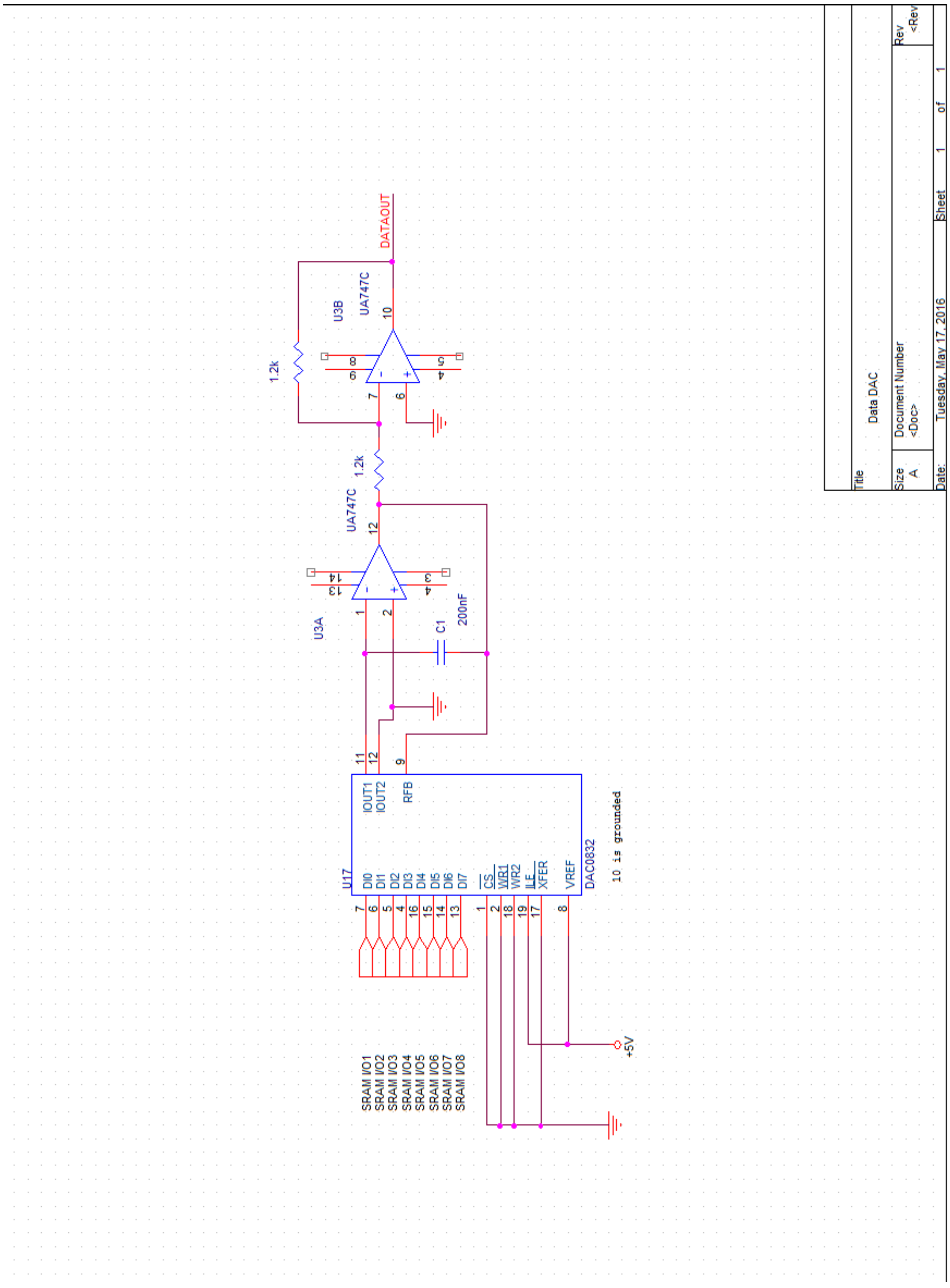


Figure 25: Schematic of data DAC.