555 Timer- Timer

HARDWARE ARCHITECTURE SPECIFICATION

REV 2.21

Developed By:

Jesse Farrell

Contact:

Jessefarrell92@gmail.com

Table of Contents

Table of Figures

Revision History

Revision	Release Date	Comments
Rev 1.0	Jan 10, 2022	Initial document for 555 contest submission Contains theoretical design including schematic and pcb
Rev 1.1	Mar 22, 2022	Added validation section using custom eval boards
Rev 2.1	May 19, 2022	Updated document to reflected updated board Added new PCB layout section, pending some final sections
Rev 2.2	Aug 08, 2022	Updated PCB layout section Added abstract Added testing and validation Added results and recommendations Added schematic appendix
Rev 2.21	Aug 10, 2022	Corrections based on reader feedback Corrected statement in section 4.3.2 statement true for NAND not AND

Abstract

The *555 Timer-Timer* project detailed in this report, was developed in late 2021, early 2022 as a candidate to the Hackaday 555 timer contest. This widget acts as a configurable count-down timer that could be used as a cooking timer. To achieve this goal, the widget uses one main board that acts as an interconnect for ten add-in style PCBs. Each add-in PCB implements one of the main logical blocks of the final design. For example, there are individual PCBs for each digit of the binary coded decimal (BCD) down-counter, the multiplexer (MUX), and the BCD to 7-segment decoder. All of these logical blocks were custom made using 555 based logic. The RTL designs were simulated using LTSpice, and occasionally breadboarded to verify their functionality before implementing the design on hardware.

Although the final design successfully met the project goals its usefulness is very limited due to its poor efficiency, accuracy, and relatively large footprint. Most likely this device will live on as a display piece in a home lab or office.

1. Introduction

The following hardware architecture specification (HAS) outlines the design process for an alarm that, almost exclusively, uses 555 IC's. This project was undertaken as a potential submission to the 555-contest hosted by Hackaday in 2021. The contest was a celebration of the 50th anniversary of the IC created by Hans Camenzind.

2. Design Goals

This project will comply with the following contest statement provided by Hackaday.

> *"Design something cool that uses the 555-timer chip or one of its variants (556, 558, low power versions, etc.), or even build your own 555 from discrete components. If it has the classic 555 circuit as a central part of the design, it's fair game.*
>
> *You're not limited only using 555s, but the timer chip should be central to the circuit. In other words, we don't want to see a design where the 555 just plays an ancillary role. The timer chip should be the point of the design."*

A personal goal for this project was to use entirely 555/556/558 IC's and passive components (Including other chips kind of ruins the fun eh!). However, this goal conflicted with the board dimension and the overall cost of the project. As a tradeoff some jelly-bean components were introduced to reduce overall component count. The complete list of design goals is presented below.

Design Goals:

- All IC's should be 555/556/558 (excluding diode's, and single transistors)
- Vin 5->12v
- 7-seg strobe rate approx. >400Hz (avoid flicker)
- Dimensions < 15cm x 15cm (may stack PCB's if required)

3. Block Diagram

The block diagram for the project is shown in Figure 1. Each block will have the function defined as follows and will be implemented in Section 5.

BCD to 7Seg: Input 4 bits representing a single digit of the 7-segment display. The 4-bit input will be mapped to a 7-bit output representing the different elements of a 7-segment digit. See section 5.1.

MUX: The multiplexer will route 1 of the 4, 4-bit buses to the drive circuit. Each 4-bit bus represents one of the base 10 digits. See section 5.2.

Selector: The "selector" will define the cycling rate for each digit of the display. The output goes to both the MUX select and decoder. See section 5.3.

Decoder: A typical decoder that takes in a 2-bit input and toggles 1 of 4 bits on the output. The decoder will be used to toggle the common of each digit of the display sequentially. See section 5.4.

Clock A: Clock A controls the refresh rate of the 7-segment display. It must be greater than ~400Hz to avoid perceivable flicker. See section 5.5.

Clock B: Clock B drives the countdown timer block at 1 tick per minute. See section 5.6.

Counter: 16-bit counter (4x 4bits). Counts down in base 10 from 9999 to 0000. See section 5.7.

Buzzer: Buzzer that activates when the counter contains all zeros. See section 5.8.

User Input: The user will be able to toggle each digit by injecting clock pulses into the different blocks of the counter circuit. There will be an "edit-enable" that must be enabled for the counter to transition from 0000->9999. See section 5.9.

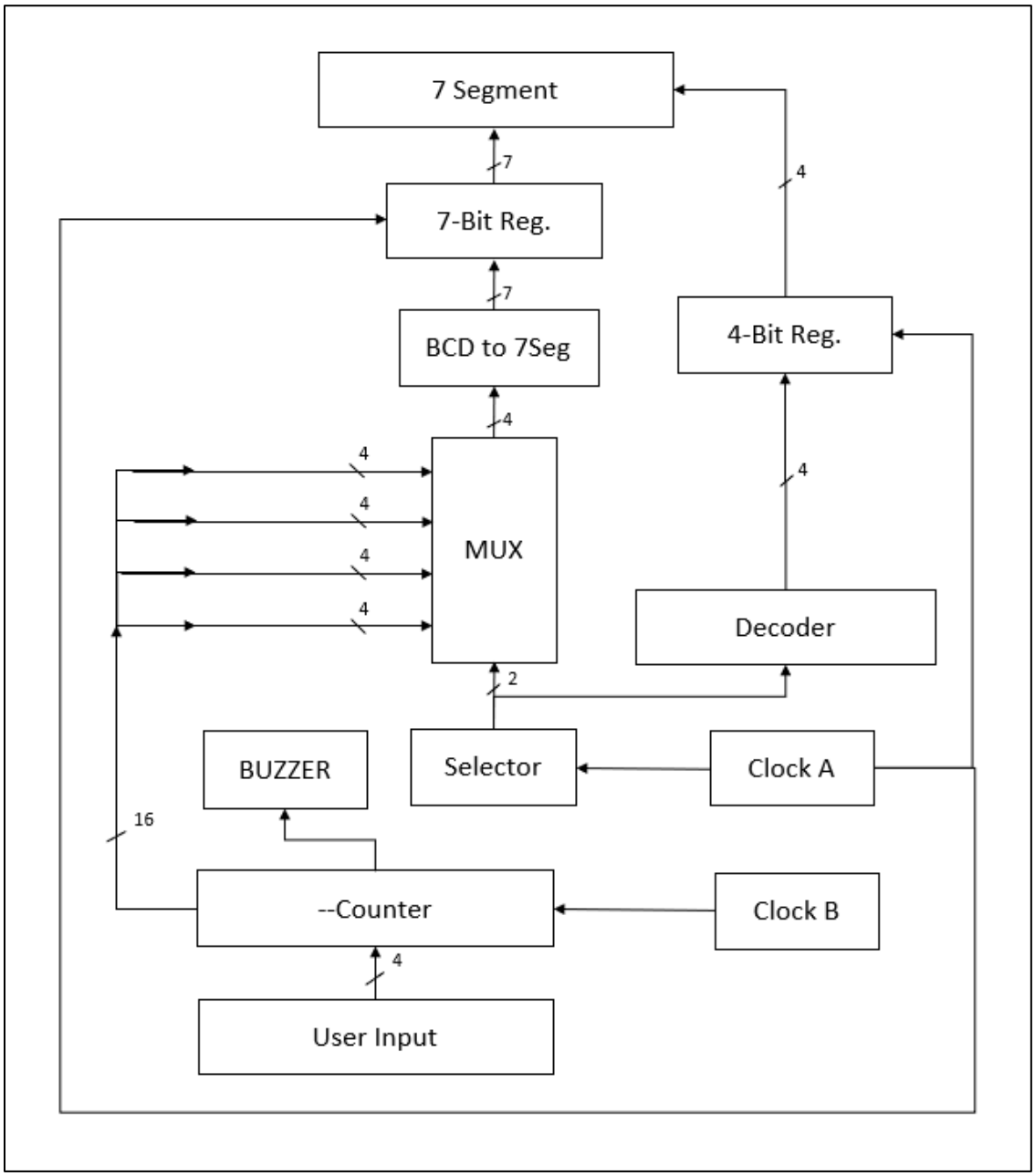


Figure 1. Timer Block Diagram

4. Theory of Operation

The following section covers some of the preliminary research regarding a variety of 555 implementations. The goal for this section is to understand how Boolean logic can be implemented using the IC.

A block diagram for the 555 is shown Figure 2. Its design includes two comparators that toggle the set and reset pins of an SR latch. Recall that an ideal op-amp will have equal potential at both inverting and non-inverting inputs, to accomplish this the output is driven accordingly to match the non-inverting input. Consider a comparator with non-inverting input IN+, and inverting input IN-. If the IN+ < IN- then the comparator will output low, and when IN+ > IN- the output is driven high.

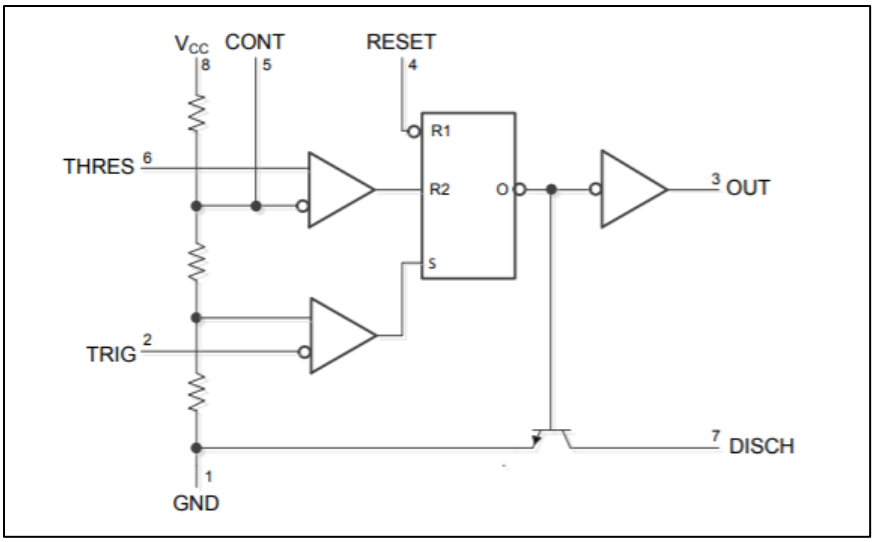


Figure 2. 555 Simplified Schematic [1]

If THRESH is greater than 2/3 VCC, R2 will go high, hence the SR latch is reset. Similarly, if TRIG falls below 1/3 VCC, S will go high, hence the SR latch is set. The output of the latch is also connected to an npn bjt whose collector is connected to the DISCH pin of the package.

4.1. Astable Operation

In astable operation the 555 timer outputs a pwm signal with configurable duty cycle and frequency. The duty cycle is restricted by the resistor R_A, meaning the charge rate of this circuit will always be greater than the discharge rate (time high > time low). The astable 555 configuration is shown in Figure 3.

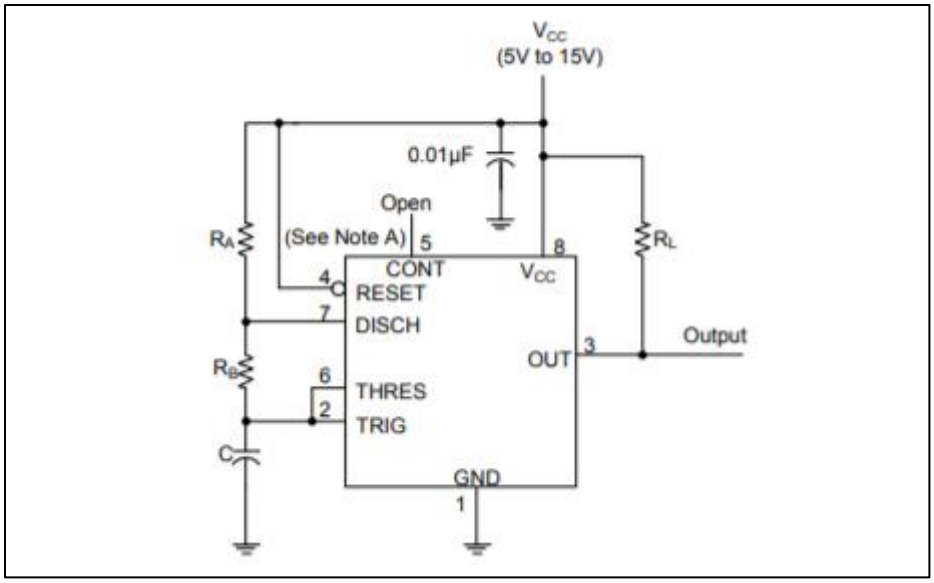


Figure 3. Astable Multivibrator [1]

The circuit can be briefly explained as follows. At startup, the capacitor C will have no charge, so pin 6/2 will be at 0 volts causing the output to go high and the discharge at pin 7 to be disabled. Since discharge is disabled, the capacitor can charge. Once charged to 2/3 VCC, the latch will be reset causing the output to go low, and discharge to be activated. Once the capacitor discharges to 1/3 VCC, the latch is set and the cycle repeats.

Based on the desired duty cycle and frequency, the values R_A , R_B, and C can be tweaked accordingly. If f is frequency, t_H is the pwm high time, and t_L is the pwm low time, then the relevant formula are as follows [1].

$$f = 1.44/(C * (R_A + 2R_B)) \quad \text{(eq.1)}$$

$$t_H = 0.693 * C * (R_A + R_B) \quad \text{(eq.2)}$$

$$t_L = 0.693 * C * (R_B) \quad \text{(eq.3)}$$

4.2. Monostable Operation

In monostable operation a single pulse is generated by pulling TRIG low. The pulse width will be dictated by the resistor (R_A) and capacitor (C). An explanation of the operation is as follows. At steady state, the capacitor remains discharged and THRESH is held low. Once TRIG falls below 1/3 VCC, the output goes high and discharge is disabled. Once the capacitor charges to 2/3 VCC, THRESH is triggered causing the output to go low discharging the capacitor. The system then remains at its idle state until TRIG falls below 1/3VCC.

Equation 4 describes the high time of the pulse generated by this monostable oscillator [1].

$$t_w = 1.1 * R_A * C \quad \text{(eq.4)}$$

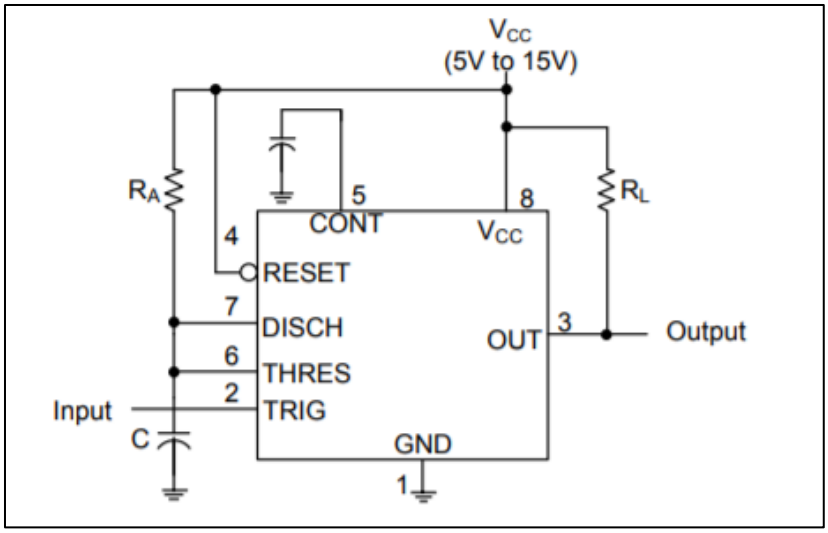


Figure 4. Monostable Operation [1]

4.3. 555 Based Boolean Logic

The following section outlines the implementation of different logic blocks using an assortment of 555's. In this section gates such as the AND, OR, and NOT will be constructed. Diodes may be included to simplify gate design as specified in section 2.

4.3.1. AND Gate

An AND gate takes in two (or more) inputs, and outputs a logical 1 when both inputs are 1. All other gates can be expressed with NAND gates, therefore other gates can be *brute forced* once this circuit, and an inverter are verified.

Consider the modified schematic shown in Figure 5. An AND gate with one inverting input can be constructed using RESET and THRES/TRIG as the gate inputs. The different conditions for this circuit are outlined below.

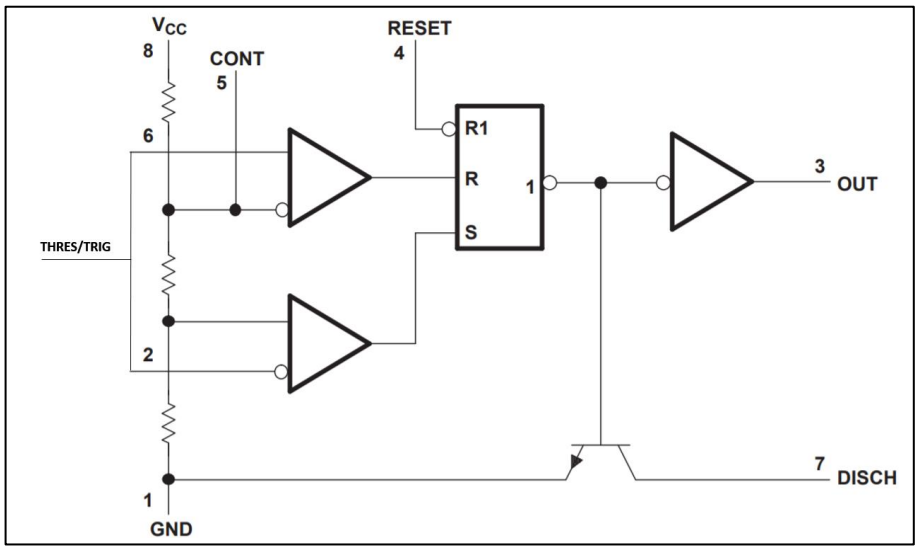


Figure 5. 555 AND Gate (w/ Single inverting input)

THRES/TRIG = 0, RESET = 0, OUT = 0:

Whenever RESET is 0, since the pin is inverting, the SR latch output will be held at 0 in a reset or disabled state.

THRES/TRIG = 1, RESET =0, OUT = 0:

Same reasoning as above; since RESET is 0, the output will be 0.

THRES/TRIG = 0, RESET = 1, OUT = 1:

Now that RESET has been de-asserted, the SR latch will function normally. For the upper comparator, the inverting pin will be at a higher potential then the non-inverting pin, so "R" is held low (it won't be asserted). The lower comparator has the opposite configuration. Its inverting pin will be at a lower potential then its non-inverting, therefore "S" is asserted. Since S is asserted, and RESET is not being held, the latch sets the output to 1.

THRES/TRIG=1, RESET = 1, OUT = 0:

RESET is de-asserted, so the SR flip-flop will not be locked low. For the upper comparator the inverting pin will be at a lower potential then the non-inverting pin, so "R" is held high. The lower comparator has the opposite configuration. Its inverting pin will be at a higher potential then its non-inverting, therefore "S" is low. Since S is low and R is high, the flip-flop resets the output to 0.

The results of the breadboarded circuit revealed an extended delay of ~15us when disabling the output via the inverting input, see Figure 6. Toggling the output via RESET took only ~100ns. However, due to the slow clock of this widget (~>400Hz or 2500us), the ~15us delay can be ignored.

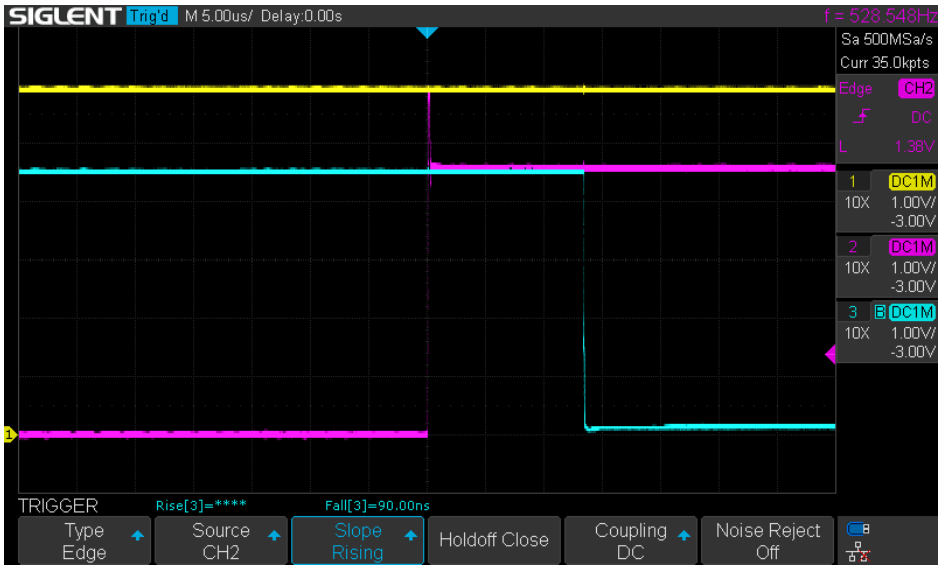


Figure 6. 555 AND (Yellow = reset/ Purple = trig/ Blue = output)

To create a true AND gate, without the inverting input, an inverter can be placed Infront of the appropriate input, see Figure 7. An explanation of the inverter implementation is provided in section 4.3.2.

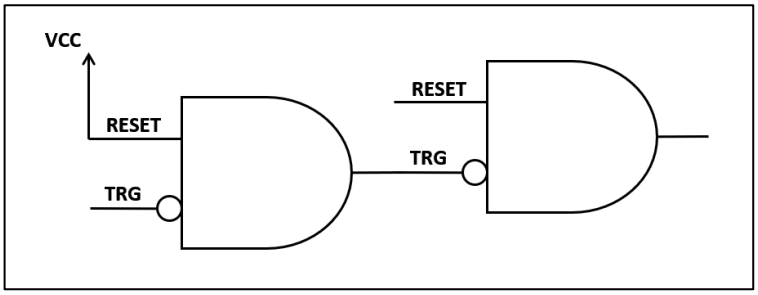


Figure 7. 555 AND

4.3.2. NOT Gate

Building from section 4.3.1, implementing an inverter is straight forward. Consider the modified schematic in Figure 8. Reset is pulled to VCC so the latch is enabled. When THRES/TRIG is low, the upper comparator will output 0, and the lower comparator will output 1. Conversely, when THRES/TRIG is high, the upper comparator will output 1, and the lower comparator will output 0. Therefore, an input of 0 causes an output of 1, and an input of 1 result in an output of 0.

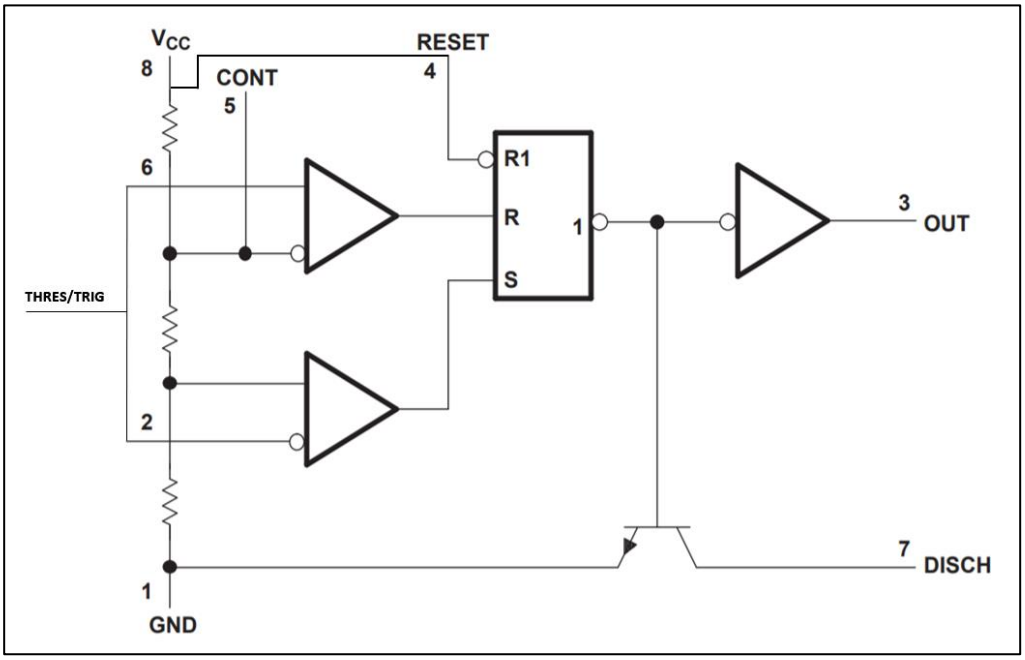


Figure 8. 555 Inverter

Breadboarding the circuit revealed the same delay observed for the AND gate, see Figure 9. As mentioned earlier, this delay won't have any notable impact on the widget.

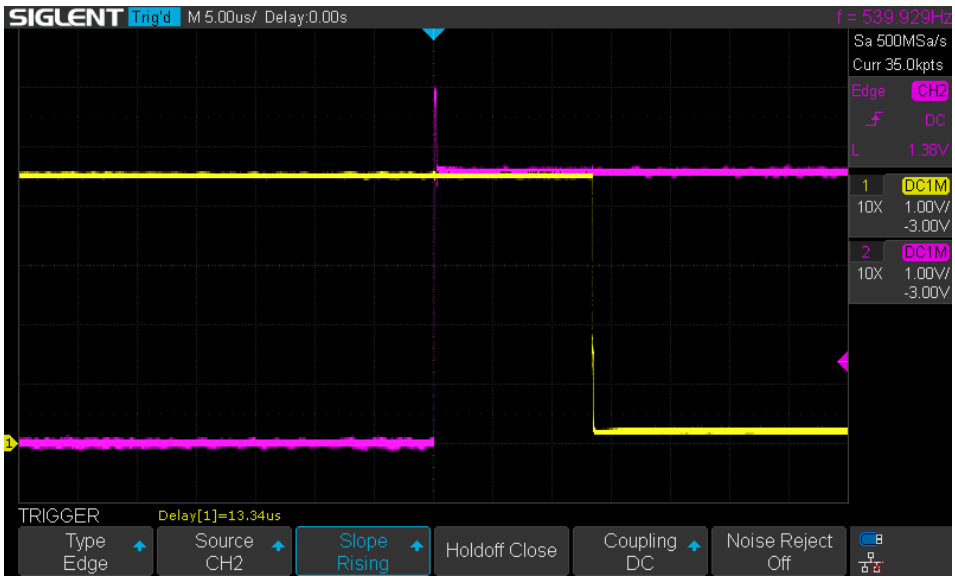


Figure 9. 555 Inverter Breadboard

4.3.3. Buffer

A buffer can be used to help reshape a waveform, or to improve the drive capabilities of a signal. This circuit can easily be constructed from the single 555 AND discussed in section 4.3.1 by tying the inverting input TRG to ground, and then using RESET as the gate input.

4.3.4. OR Gate

An OR gate takes two (or more) inputs, if one or more input is 1, the output will be 1. This may be implemented using the configuration shown in Figure 10. To keep things uniform, the AND blocks are represented in their single 555 configuration (i.e. the TRG input is inverted, see section 4.3.1).

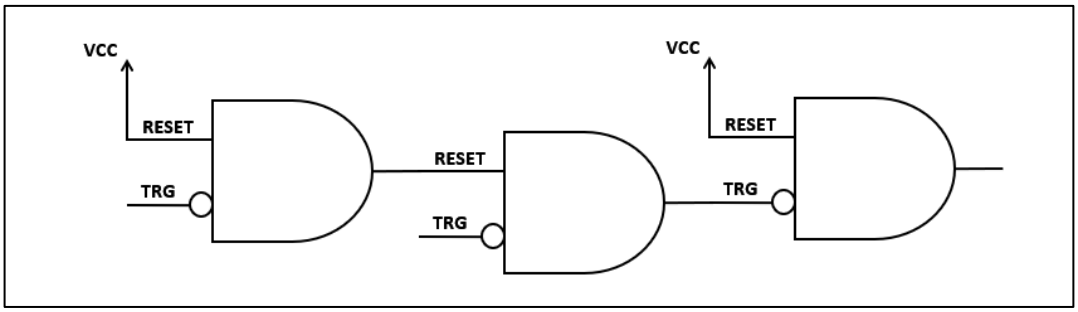


Figure 10. 555 OR Gate

Perhaps more useful, or at least efficient, is to create a NOR gate using two 555's, shown in Figure 11.

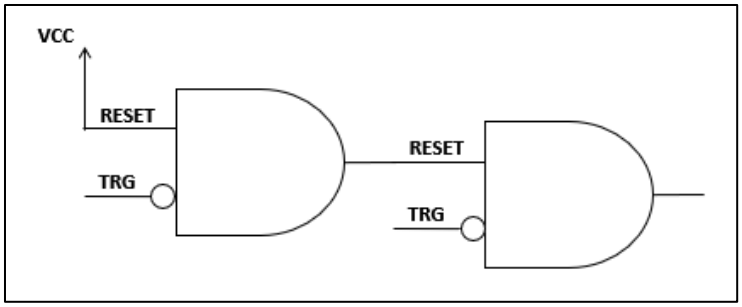


Figure 11. 555 NOR Gate

This circuit could also be implemented with a pair of diodes (Figure 12). Since this circuit replaces 3 relatively expensive ICs with 2 diodes and a resistor, the diode OR gate will be acceptable in this design. Unlike the 555 implementations, the diode OR gate **cannot** be cascaded due to losses across subsequent stages (diode drop). This circuit should use diodes with low leakage and forward voltage, likely some flavor of Schottky.

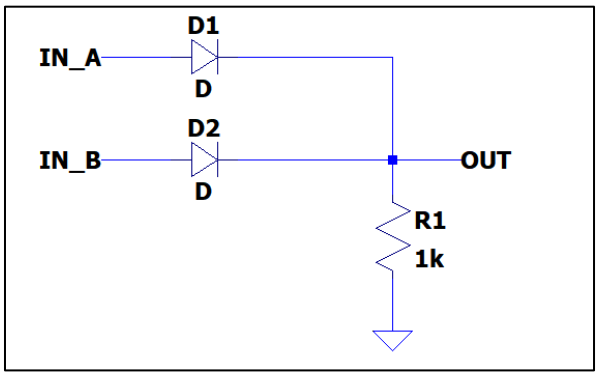


Figure 12. Diode OR Gate

4.3.5. XOR Gate

An XOR gate returns a 1 if a single input is high. If multiple inputs are high, then it returns a 0. The XOR gate may be implemented with 2x 555's and an OR gate (Figure 13).

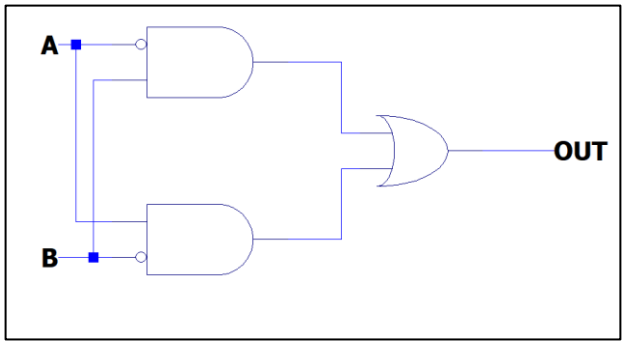


Figure 13. 555 & Diode XOR Gate

The breadboarded circuit worked as expected, albeit with the previously mentioned delay, see Figure 14. Note there is some voltage drop at the output due to the diode OR circuit. Channel 1 and 2 are the inputs and channel 3 is the output.

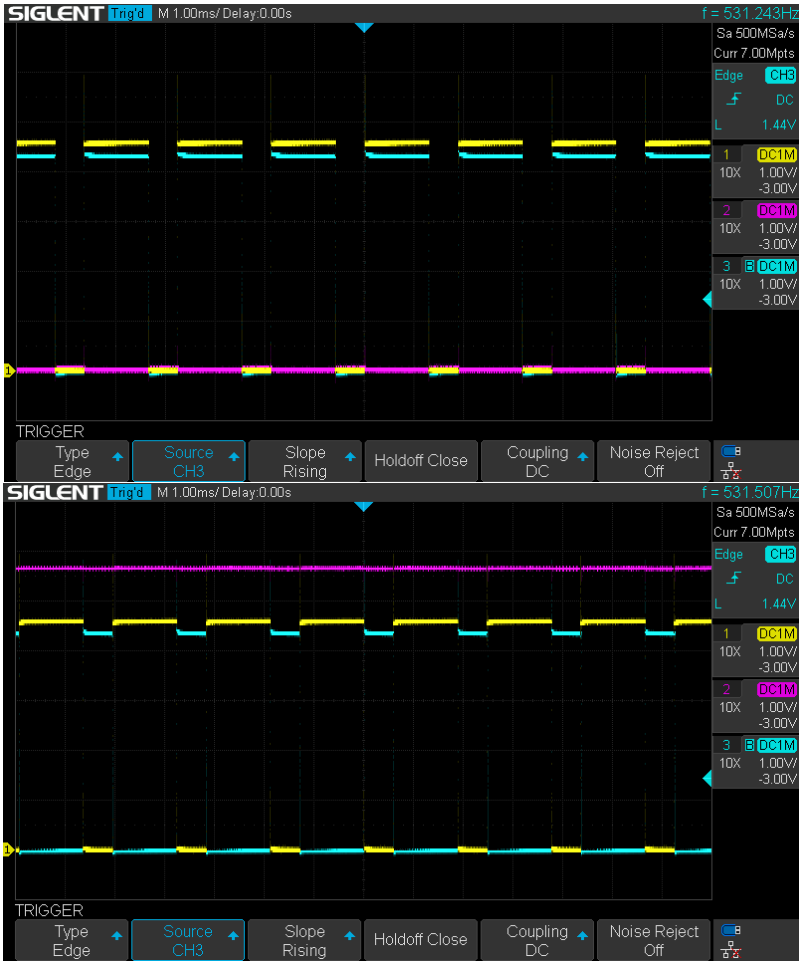


Figure 14. 555 XOR

4.4. D Type Latch

Designing a D type latch using only 555's was less trivial than initially expected. The main issue was implementing the functionality of the enable signal. The latch could easily be brute forced using 5 or so 555's but this would dramatically bloat the project BOM.

Consider the block diagram of section 3, the widget will need to store several bytes of data, using excess gates in these latches will dramatically increase the overall cost and footprint of the board. Therefore, some basic active components will be allowed for the D type latch. One good solution posted by Tim was to introduce a pass-through transistor and hold the THRES/TRIG pins at ½ VCC, see Figure 15.

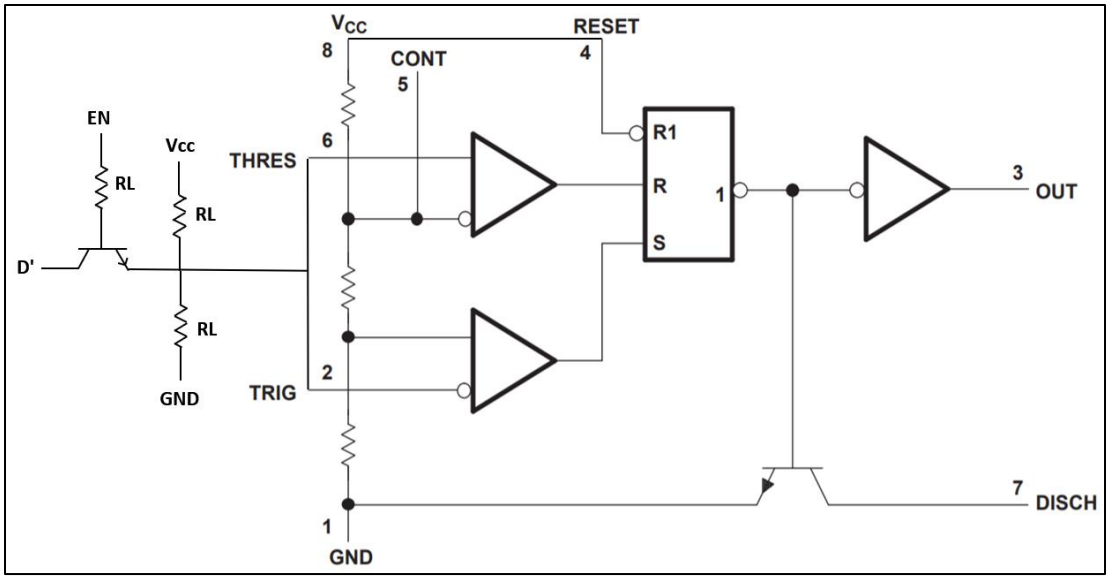


Figure 15. D Type Latch

It's important to note that the implementation in Figure 15 uses the inverted input D. Alternatively, think of this as taking in D and outputting Q' (Figure 16). This inversion could be removed a NOT, but it can be avoided altogether when designing a flip flop (see section 4.5). A small capacitor should be added to the emitter of the transistor to help stabilize the SR triggering.

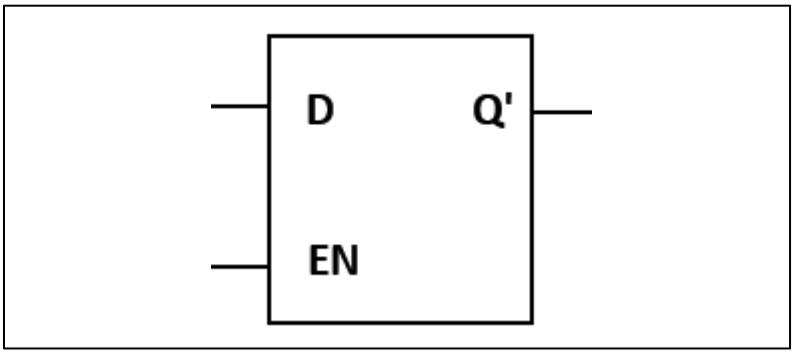


Figure 16. 555 D Type Latch

4.5. D Type Flip Flop

Given the block in Figure 16 described in the prior section, a flip flop can be created by cascading two of these latches. Note that this flip flop will trigger on the falling edge of the clock signal, and the previously mentioned inversion solves itself.

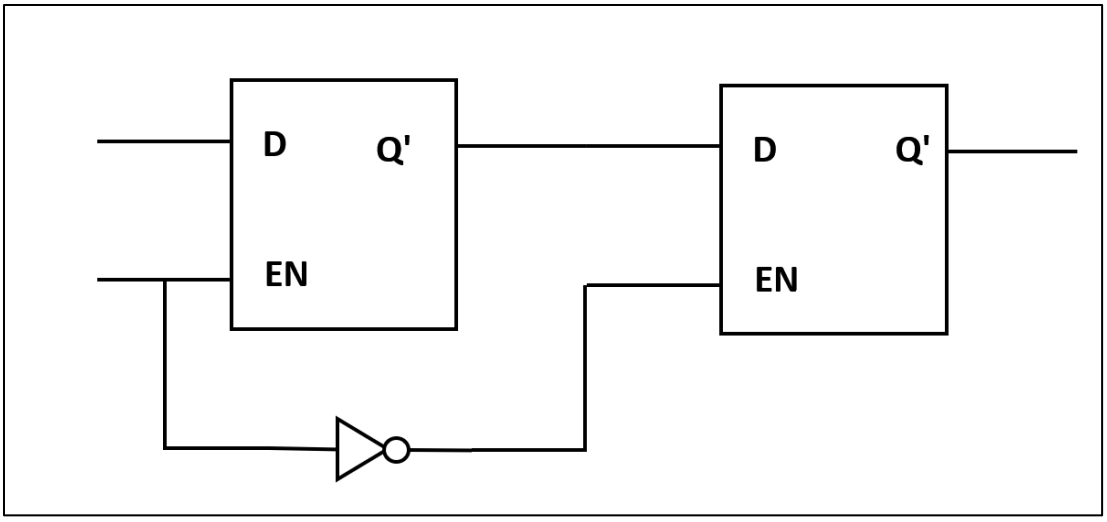


Figure 17. 555 D-Type Flip Flop (3x 555, 2x npn)

An example of the breadboarded circuit is shown in Figure 18. Channel 1 is the clock input, Channel 2 is the data in, Channel 3 is the latch output (note we expect this to be !data), and Channel 4 is the trigger pin of the 555.

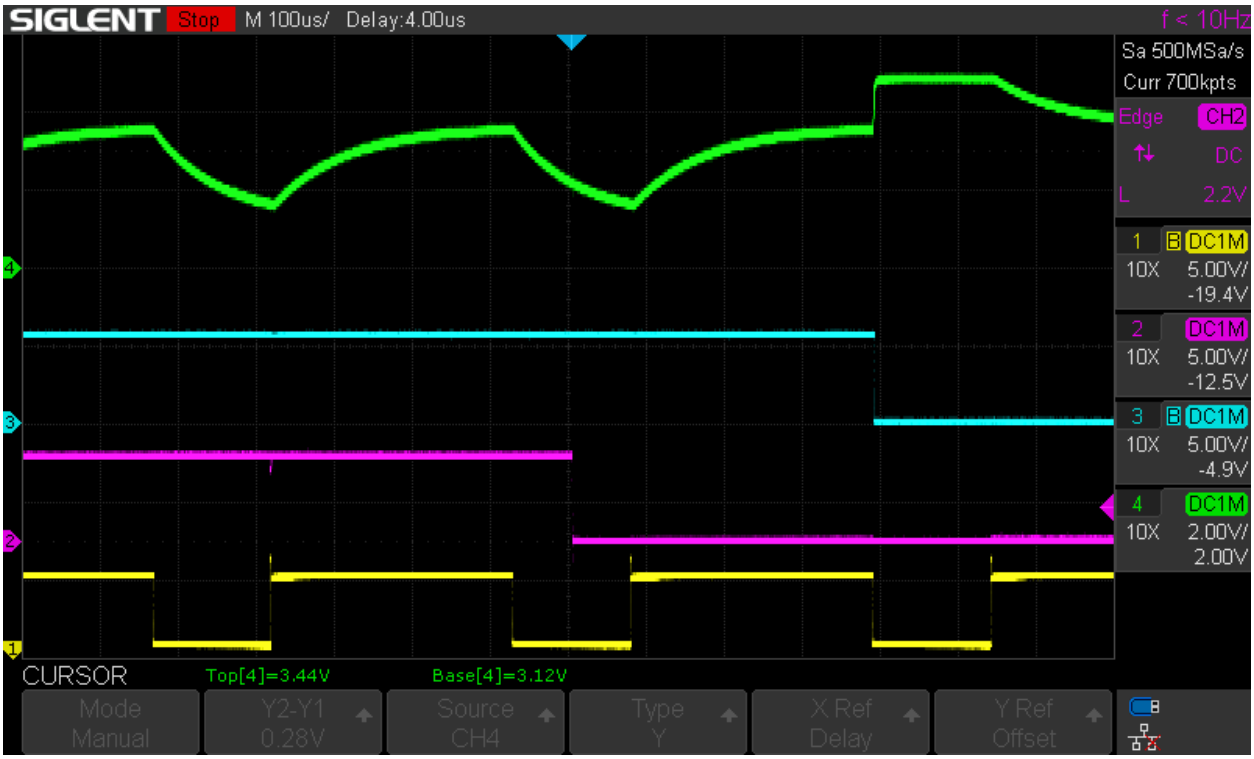


Figure 18. 555 Flip-Flop Scope Capture

5. Block Implementations

There are several blocks discussed in section 3 that need to be developed. This section covers the design of each of these blocks including simulation results.

5.1. BCD to 7 Segment

This block takes in 4 bits representing a single digit of the 7-segment display and maps them to a 7-bit output used to toggle specific elements of the display. There are two approaches that could be taken to design this circuit. Either (a) find the Boolean logic related to each element and create a network of gates that will decode the value on the fly, or I will (b) implement the logic as a mux with hardcoded results. See section 5.2 for the MUX design.

The MUX implementation would require at least 80x 555's based on the circuit in section 5.2. The logic-based approach is shown in Figure 19 and would require 26x 555's (excluding flip flops at output). Due to the drastically reduced gate requirement, the combinational logic approach will be used. A Karnaugh map was developed for each segment of the display to aid in the logic design.

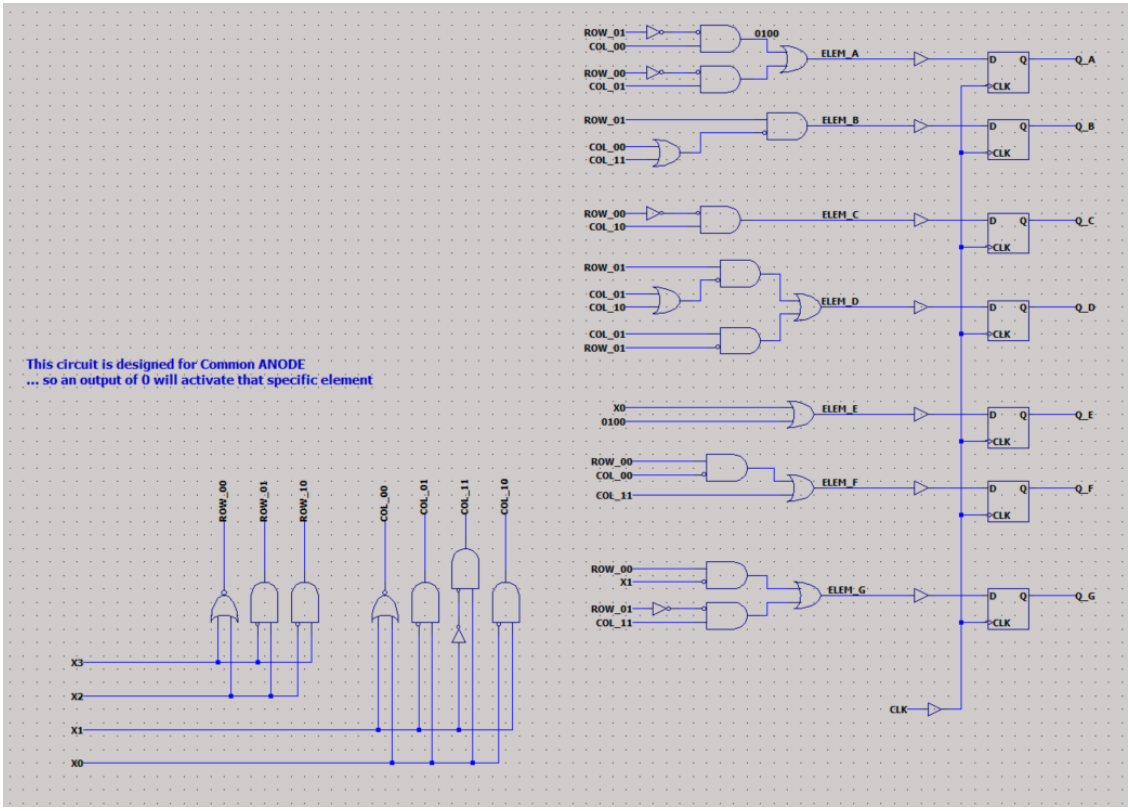


Figure 19. BCD to 7-Segment Logic

LTSpice supported the expected functionality of this block, see Figure 20. For example, when the input is 0000, element G is turned off and all others are left on, thus the 7-segment would display a 0. Note that the widget will use a common anode display, therefore when an element is 0, the segment turns on.

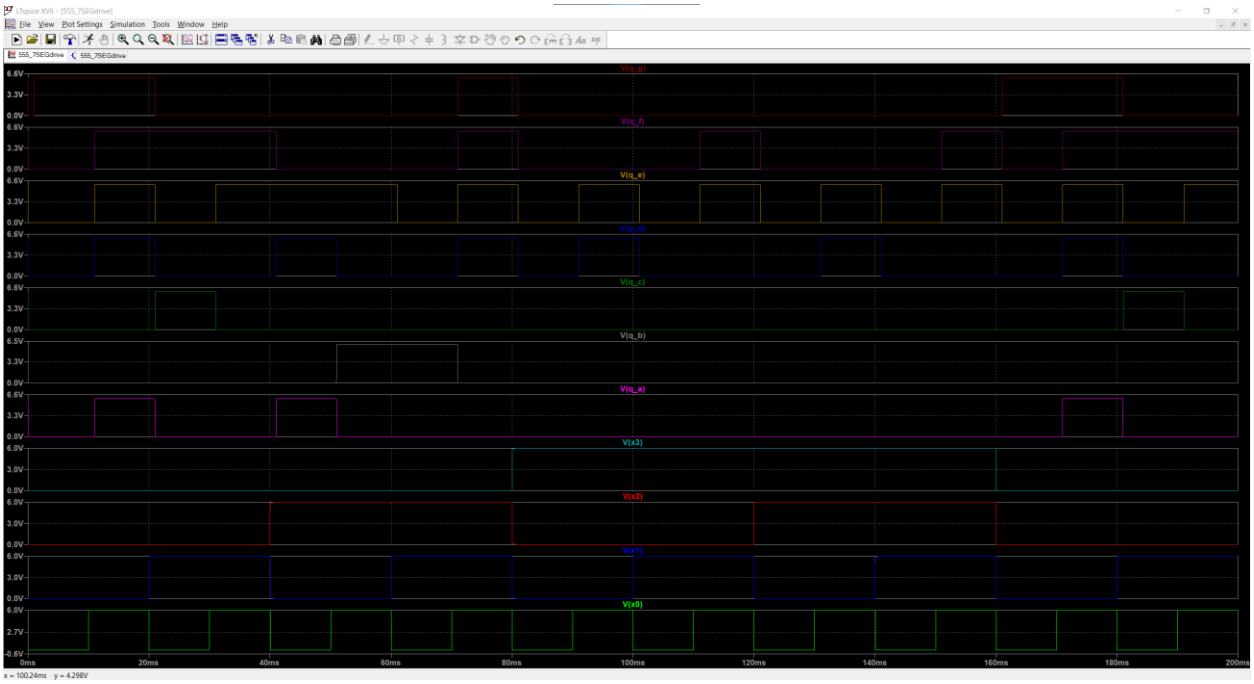


Figure 20. Binary to 7-Segment Simulation

5.2. MUX

The multiplexer block will route 1 of the 4, 4-bit buses to the decoder outlined in section 5.1. Its implementation is relatively common and covered in most introductory electronics courses. One difference being the basic 1x 555 AND block was used, leading to a few extra logical inversions. See Figure 21.

The final 4-1 MUX requires a total of 26x 555's, 16 diodes, and a handful of resistors. The circuit was simulated using LTSpice and worked as expected. Due to the losses through the diode OR gates, the output signals should be buffered.

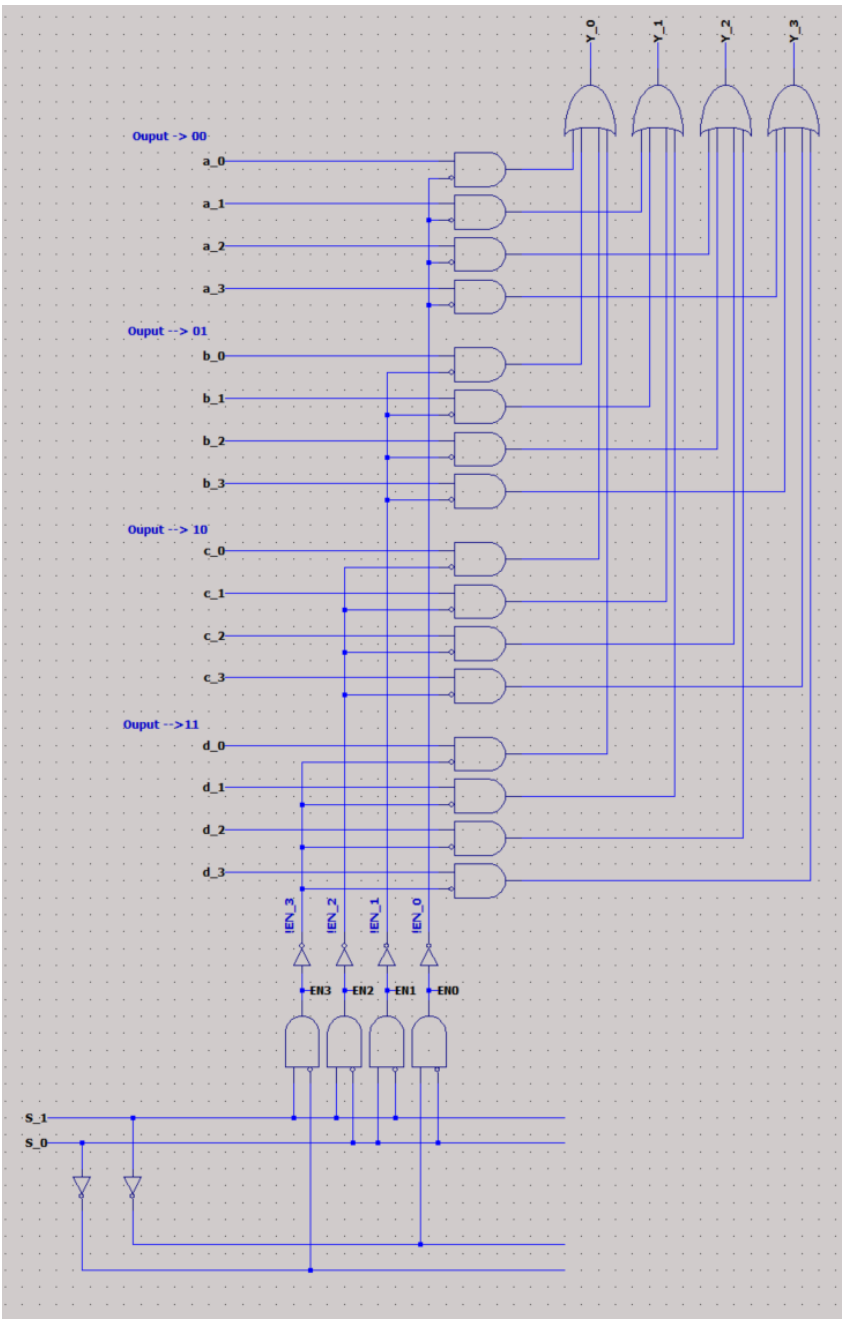


Figure 21. 555 MUX

5.3. Selector

The “selector” circuit cycles through the different lines of the MUX. It would be simple to implement this as two astable multivibrators, but over time the signals would drift apart. To get around this the selector circuit will be implemented as a frequency divider, see Figure 22. Note that the flip flops only toggle on the falling edge of the clock. The simulation works as expected, seen in Figure 23.

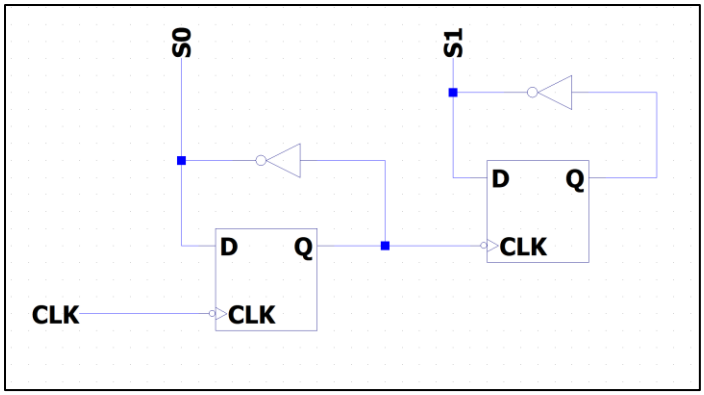


Figure 22. 555 Frequency Divider

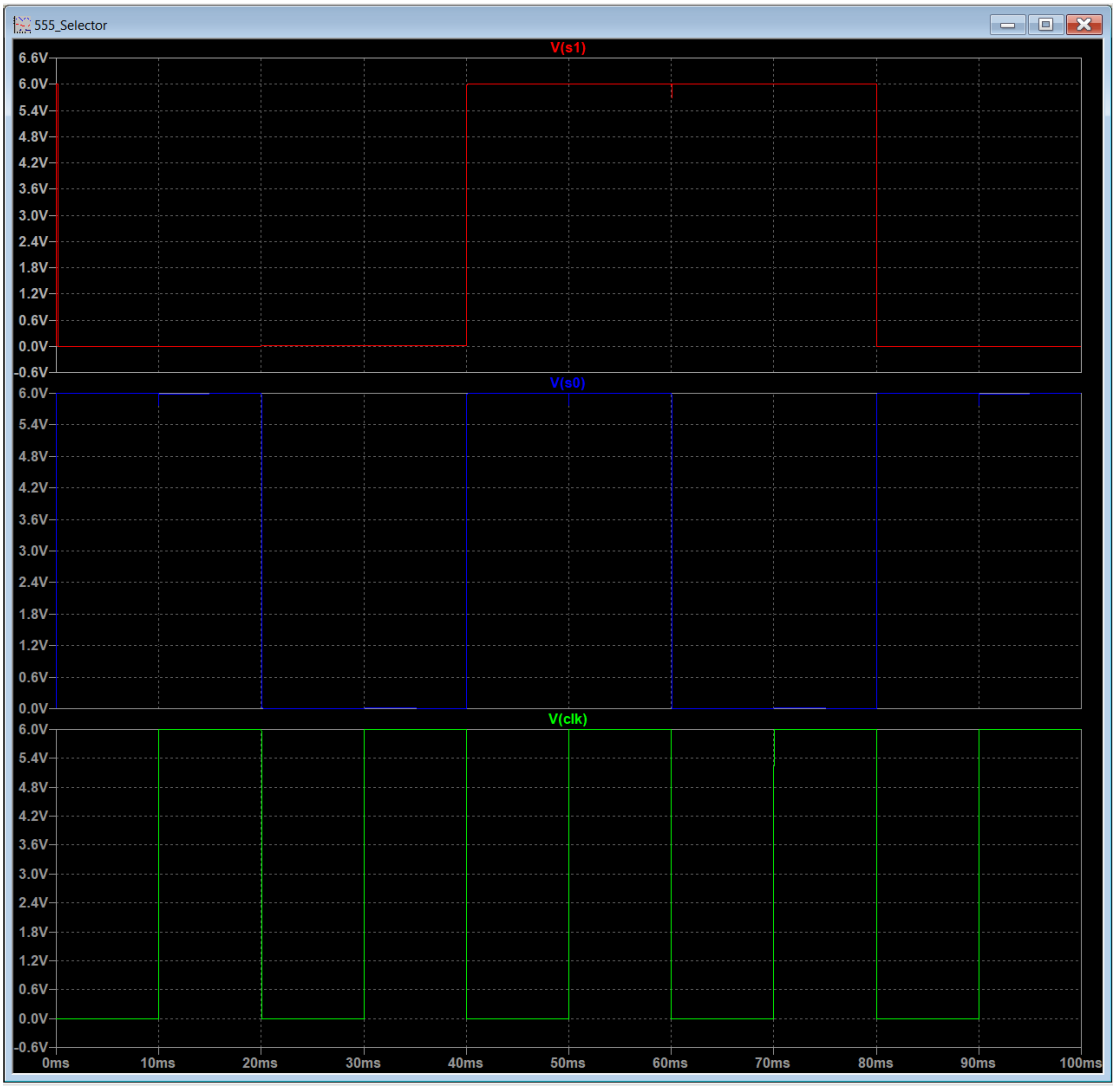


Figure 23. 555 Selector Simulation

5.4. Decoder

This is a decoder that takes in a 2-bit input and toggles 1 of 4 bits on the output. The decoder will be used to toggle the common of each digit sequentially. Any glitches in this circuit will result in intermittent flicker of the digits, therefore the output should include flip flops.

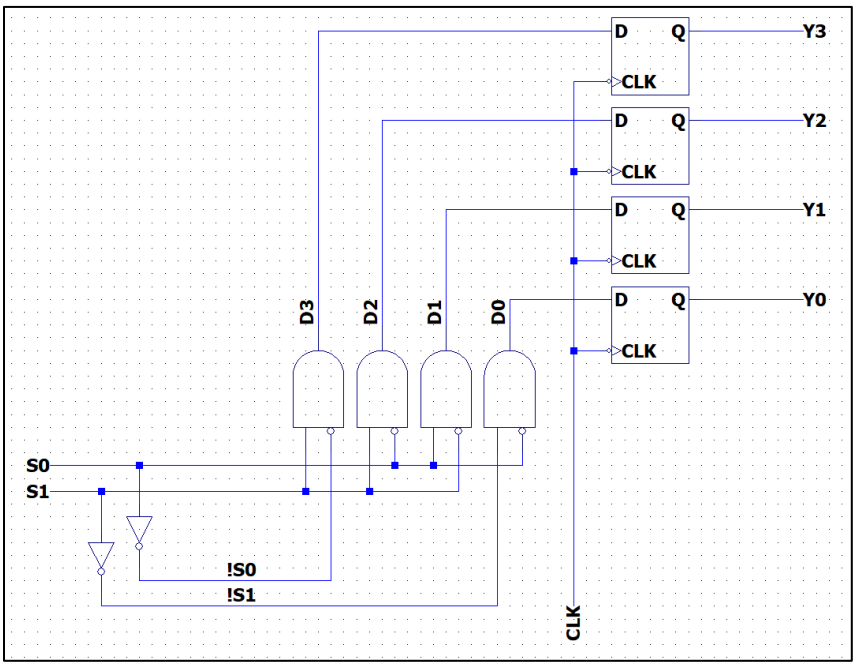


Figure 24. 555 Decoder

A decoder was implemented at the base of the MUX circuit presented in section 5.2. The widget will use the signal from the MUX's decoder, to help reduce the overall component count.

5.5. Clock A

Clock A will be an astable 555 configured for ~700Hz (or greater). The circuit shown in section 4.1 needs to be slightly modified to achieve a 50% duty cycle. Figure 25 shows the modification to lock the circuit at ~50% duty cycle. Notice that the charge and discharge of capacitor C1 is almost entirely dictated by R1 and C1. Charging will also be slightly affected by R2, but if this value is large enough it should be negligible. For reference, the breadboarded circuit is shown in Figure 26. To determine the frequency, see equation 5 [2].

$$f \approx 1/(0.693 * 2 * R_1 * C_1) \qquad \text{(eq.5)}$$

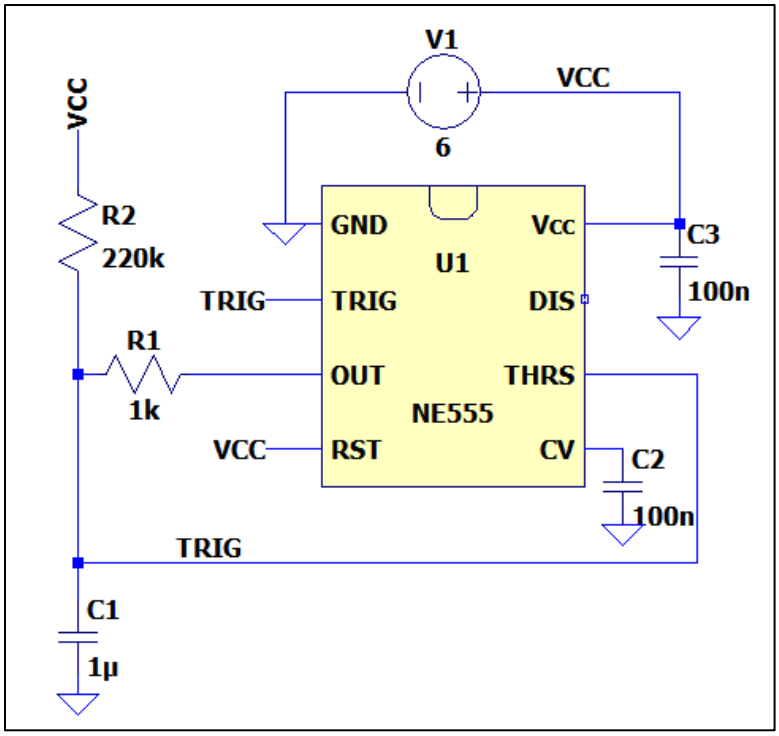


Figure 25. Clock A Configured for ~700Hz

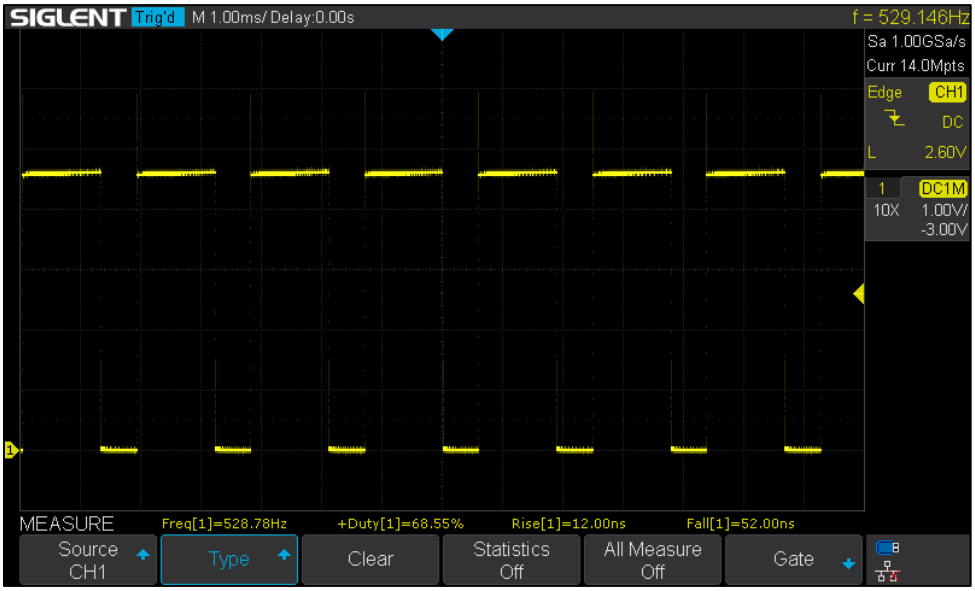


Figure 26. Clock A Breadboard

5.6. Clock B

Unlike Clock A, Clock B will target a very large duty cycle (short off time) using the astable multi-vibrator configuration. A short off time will allow users to inject clock pulses during normal operation. If the clock is high for 59s and low for 1s, a user can inject a pulse by pulling the output low via a push button. The final circuit should include several DNS components in the charge circuit to allow for future tweaking.

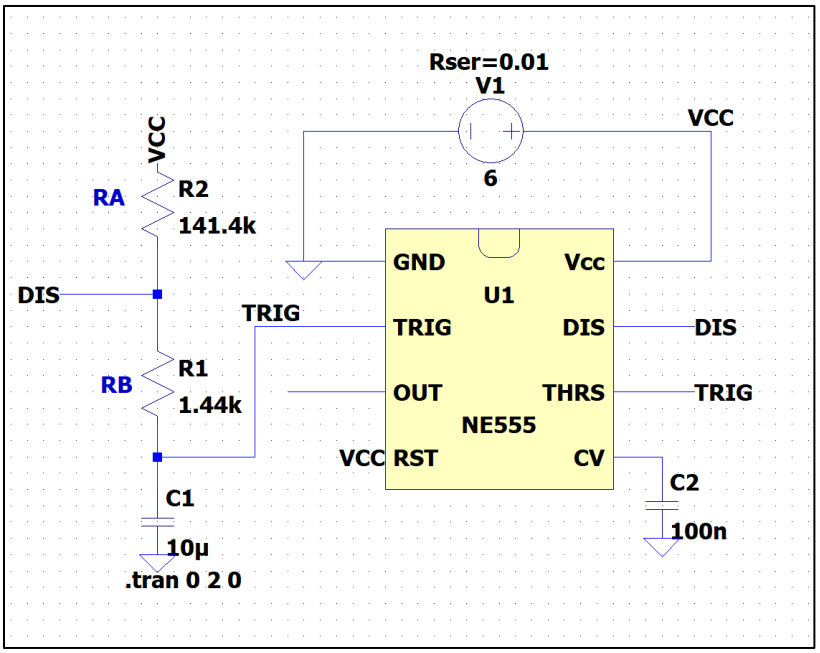


Figure 27. Clock B ~1 minute period

5.7. Counter

To simplify the counter decoder block presented in section 3, the counter will be implemented in a base 10 scheme using binary coded decimal (BCD). Although this will make the counter block less efficient so to speak, it will dramatically reduce the complexity of the counter decoding down the road.

The counter will take clock B as an input. With each clock pulse the contents of the counter need to be decremented by 1. There will be 4, 4-bit registers that contain the base 10 value for each 7-segment digit. Meaning if the counter is displaying 2355, the registers should contain the following [0010/0011/0101/0101]. After the next clock pulse, the contents will be decremented, and registers will now contain [0010/0011/0101/0100]. The final counter circuit is shown in Figure 28, and a brief explanation will be provided below.

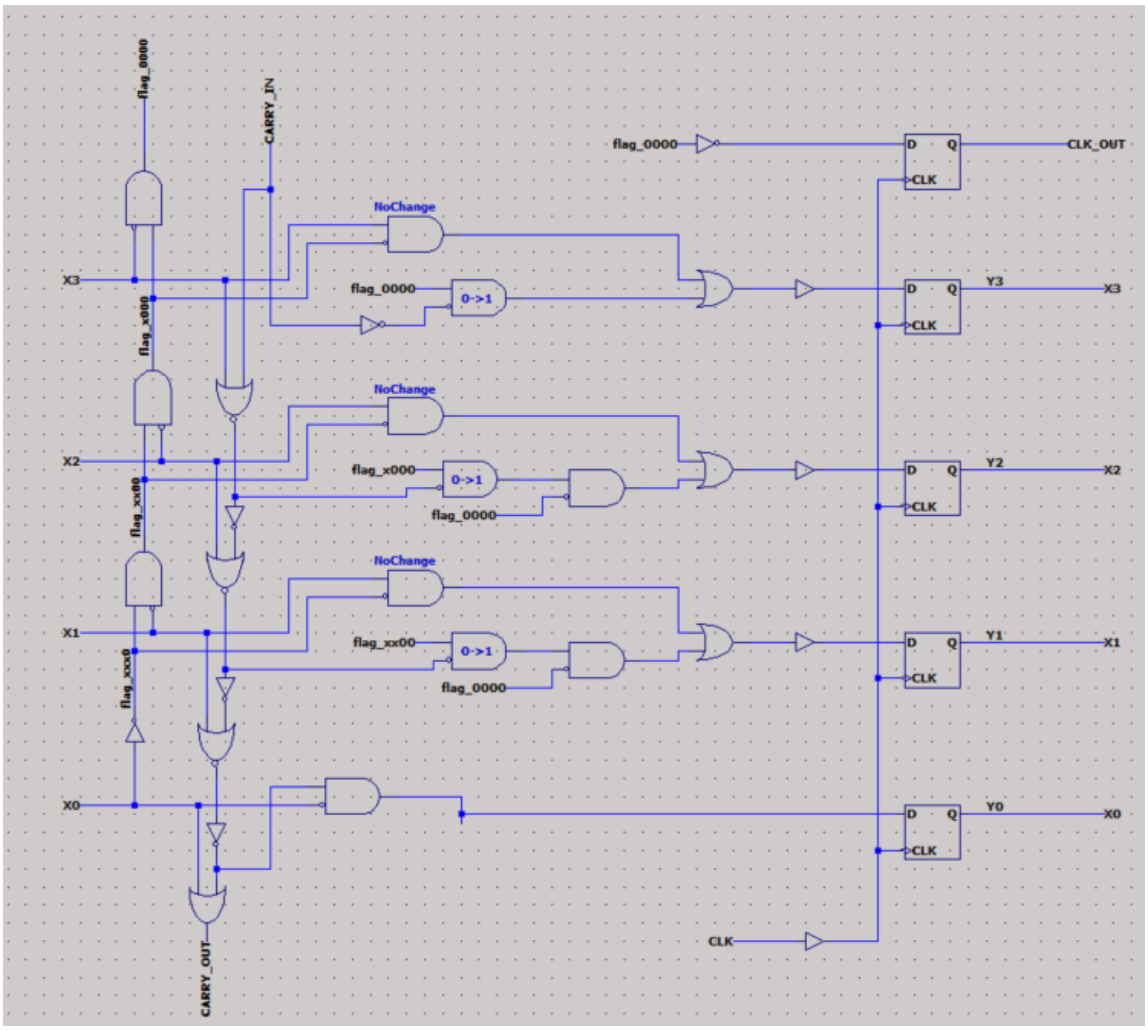


Figure 28. BCD Count Down Block

The logic for each bit is relatively symmetric, excluding X0, so the logic will be explained for X3 and the reader should be able to make sense of the subsequent logic. It may be helpful to refer to Table 1.

Table 1. BCD Counter States

NEXT STATE	Bit 3	Bit 2	Bit 1	Bit 0
↓	1	0	0	1
↓	1	0	0	0
↓	0	1	1	1
↓	0	1	1	0
↓	0	1	0	1
↓	0	1	0	0
↓	0	0	1	1
↓	0	0	1	0
↓	0	0	0	1
↓	0	0	0	0

For a transition from 1 to 0 requires that all lower bits be 0. To transition from 0 to 1 requires that all lower bits be 0 and there is a 1 in any of the upper bits. This “look ahead” and “look back” logic is achieved by the branch of AND and NOR gates on the left hand side of the circuit. For the highest bit X3, this is pulled from the next BCD block using CARRY_IN. Similarly, when bit X3 transitions from 0 to 1 the next BCD block is given a clock pulse via CLOCK_OUT. Note that when the counter loops it begins at 1001 (9_{10}).

It is difficult to simulate this circuit in its entirety but a short snippet is shown in Figure 29. This capture shows all of registers 0 and 1 and a portion of register 3 (where each register holds 1 digit of the 7 segment). The transitions are as expected, this screen capture shows a 0->9 transition, notice each counter resets to 1001 and not 1111.

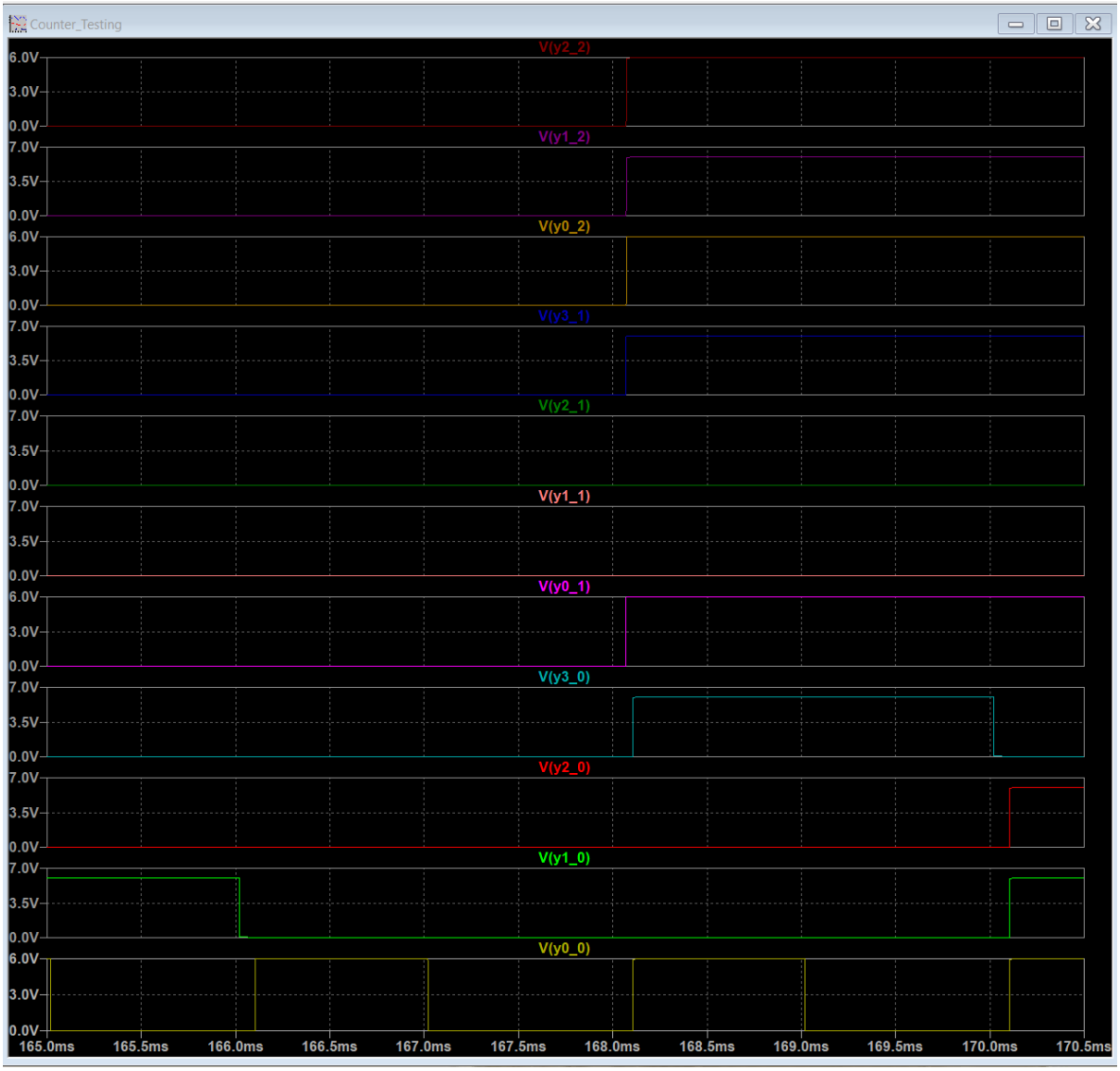


Figure 29. BCD Counter Simulation

5.8. Buzzer

When all the bits of the counter are 0, the buzzer will be momentarily enabled. The CLK_OUT from the last BCD counter shown in section 5.7 is 0 when every bit is 0. Therefore, this signal can trigger the BUZZER circuit. Note that the buzzer will artificially trigger when the user is cycling through the timer unless CARRY_IN is pulled high. This circuit is shown with simulation results in Figure 30.

A basic explanation of its operation is as follows. When enable goes high the signal is inverted by the inverter, attenuated by a voltage divider, and then high pass filtered by the coupling capacitor C5. This creates a negative pulse with a peak less than or equal ½ VCC. Since TRIG is held at ½ VCC this pulse causes TRIG to reach approximately 0 volts momentarily enabling the output of the first 555 (see section 4.2 for monostable oscillator). Since the first 555 is configured as a monostable oscillator this sends a positive pulse to the next 555's RESET pin. For the duration of this pulse the second 555 will be enabled and it will output a pwm signal for the speaker.

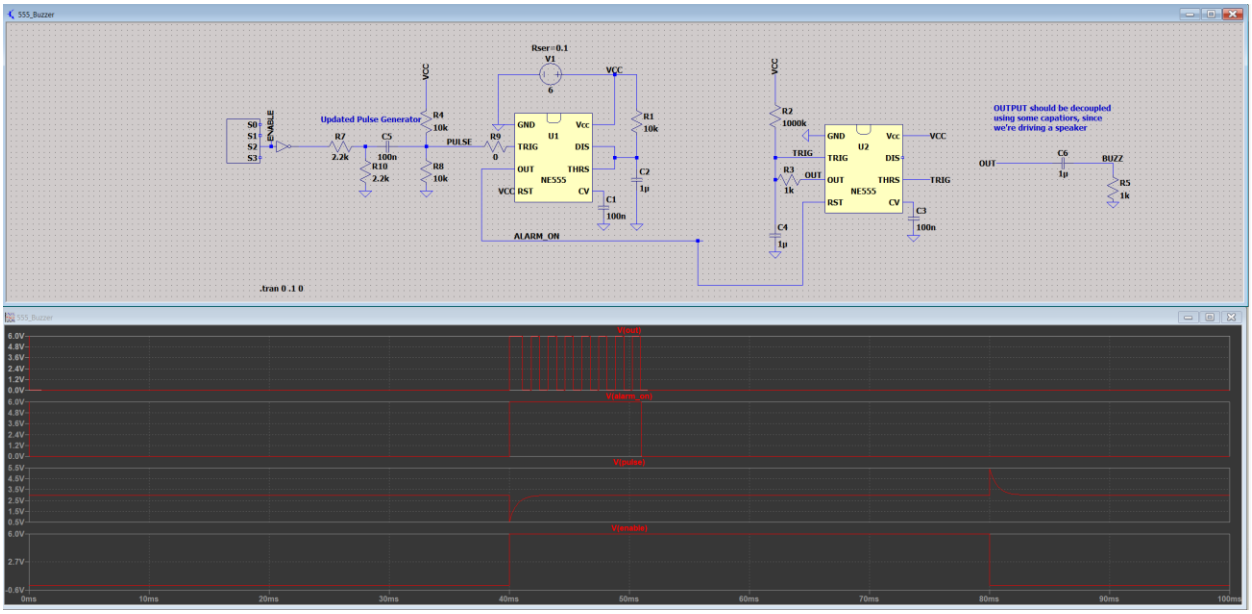


Figure 30. 555 Buzzer Circuit

5.9. User Inputs

The last block that needs to be implemented is the user input. The user will be able to (a) inject clock pulses into each 4bit counter, (b) toggle CARRY_IN high or low to allow the counter to loop, and (c) disable the Clock_B to pause the timer.

To allow for clock pulse injections the clock will be buffered at the input of each block. Then a push button will be placed at the input of the buffer, and a high value resistor will isolate the push button from the rest of the circuit. The CARRY_IN bit for the BCD counter can be included via a single DIP switch on the most significant 4-bit counter. While the CARRY_IN is held at VCC the timer will be able to loop from 0000 to 9999 without sounding the buzzer. Lastly, a disable will be added to Clock_B via the reset pin of its 555 IC.

6. Component Selection

The following section is a brief overview of some of the critical components for this circuit. Now that the entire circuit has been established each block in section 5 will be revisited and components will be selected. For a list of components refer to Table 2. Component selection was primarily driven by availability and price.

Table 2. Widget Component List

Manufacturer Part Number	Manufacturer Name	Description	Datasheet
NE555PWR	Texas Instruments	IC OSC SGL TIMER 100KHZ 8TSSOP	https://www.ti.com/general/docs/suppproductinfo.tsp?distId=10&gotoUrl=https%3A%2F%2Fwww.ti.com%2Flit%2Fgpn%2Fse555
MMST3904-7-F	Diodes Incorporated	TRANS NPN 40V 0.2A SC70-3	https://www.diodes.com/assets/Datasheets/MMST3904.pdf
1SS404,H3F	Toshiba Semiconductor and Storage	DIODE SCHOTTKY 20V 300MA USC	https://toshiba.semicon-storage.com/info/docget.jsp?did=3400&prodName=1SS404
TDCR1050M	Vishay Semiconductor Opto Division	DISP 7SEG 0.39" QUAD RED 16DIP	https://www.vishay.com/docs/83180/tdcx10x0m.pdf

6.1. 555 IC: NE555PWR

At the time of writing only single circuit 555's were economically viable for this project. Ideally, the design would have used 556/558s to reduce both the footprint and cost of the design. Most 555's are capable of sinking/sourcing up to 200 mA, and can switch faster than ~100KHz. The NE555PWR from TI provides similar performance at a relatively competitive price [4].

6.2. Diode: 1SS404

The OR gate will use Schottky diodes due to their low forward voltage drop and relatively faster switching times. The exact switching speed of the Schottky is not of much concern since the cycling rate of the display is only ~700Hz.

One of the cheaper diodes that fit this requirement was the 1SS404 from Toshiba. This diode features low reverse leakage, adequate forward current, and most importantly was one of the cheaper options.

The maximum reverse current for this diode is 50uA, and 4 of these are put in parallel to create a 4 input OR gate. A pull-down resistor of 2.2kΩ would introduce a maximum offset of 4*50uA * 2.2kΩ = 0.44V. Since the logic level transitions occur at 1/3 VCC and 2/3 VCC, where VCC is 6 V, this voltage offset is tolerable in the current system.

6.3. NPN Transistor: MMST3904

The MMST3904 [6] is an npn bjt that will be used for the flip flop circuits. This component was chosen since it mirrors the classic 2N3904 used in prototyping. Additionally, it was one of the cheaper options.

7. Schematic Capture

This section contains a brief overview of the schematic developed in KiCad. The circuit was developed using a modular approach to reduce the overall footprint of the design. In doing so each block was developed as a PCB that plugs into a main board. The full list of boards is as follows: (1) Main Board, (2) 4x BCD Counter, (3) MUX, (4) BCD to 7 Segment, (5) 4 Bit Register, (6) Buzzer, (7) Clock Sources, and (8) Power. All schematics are presented as is, and any issues encountered will be discussed in section 9.0.

See appendix A for the top-level schematic of each board. For the full schematic see attached documentation (if on Hackaday see document files).

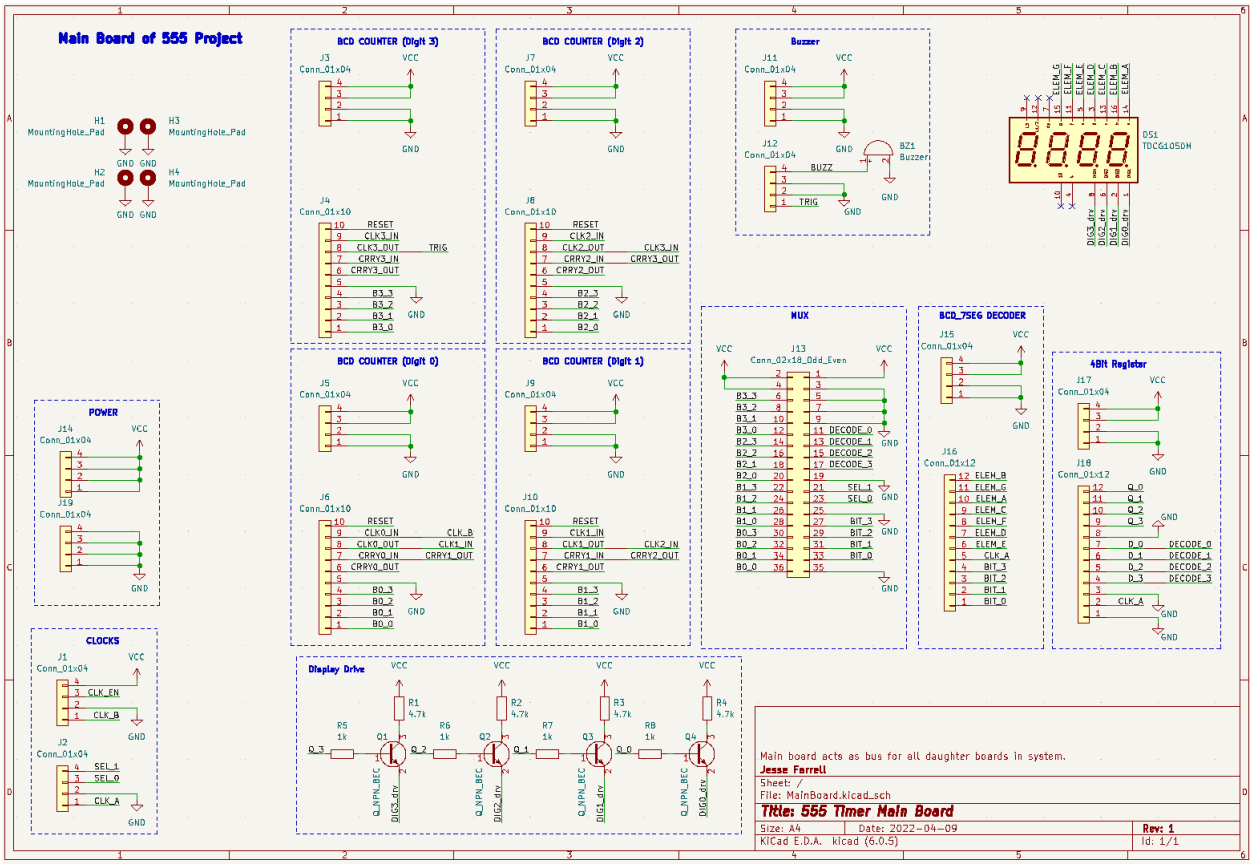


Figure 31. Main Board Schematic

8. PCB Layout

Where possible all PCBs were designed using 2-layers with a common interconnect. All daughter boards will interface with the main board via 2.54mm male headers. Where possible power and ground pins will be separate from the I/O by 1 x 2.54mm slot. See Figure 32 for an illustration. All I/O will be labeled to assist in debugging. Any unused I/O will be connected to GND.

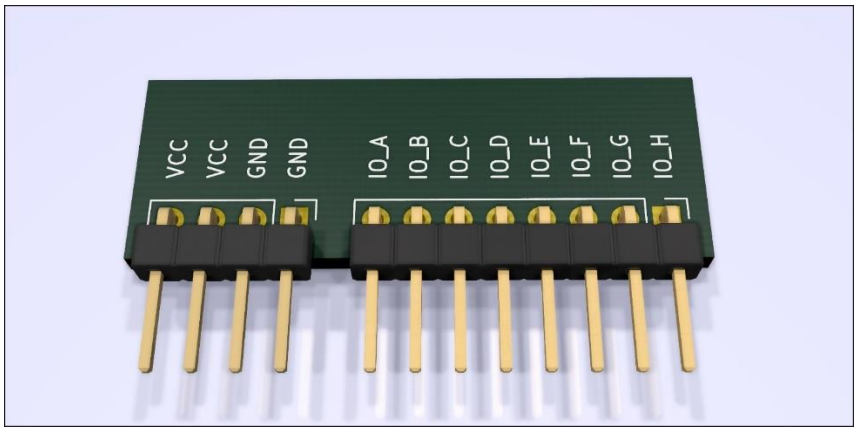


Figure 32. PCB Interconnect Template

8.1. Main Board

The main board schematic is shown in Figure 33. All user inputs, 5 push buttons, and 2 dip switches, are labeled using naming conventions discussed in this report. To help resolve unforeseen logical errors, 2 14SOIC footprints were included on the main board with all I/O exposed. If needed 74 series IC's will populate these footprints and be "bodged" into the system.

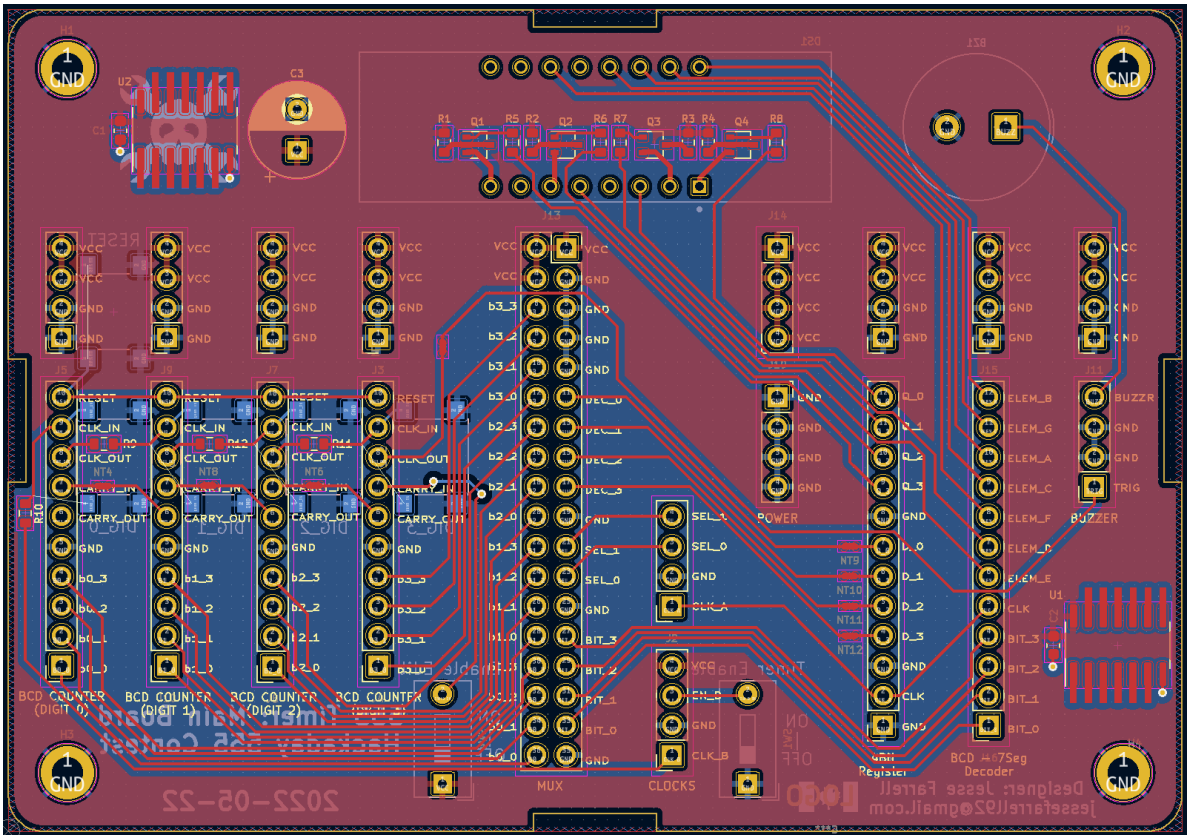


Figure 33. Main Board PCB Layout

8.2. BCD 7Segment Decoder Layout

Due to the number of interconnects in the BCD to 7 Segment decoder, a 4-layer board was used. The inner layers include a solid ground and power plane that helped ease the layout process. Although a 2-layer board would be feasible in this design given the signals are not high speed, doing so would have complicated layout, also its only a couple extra dollars from JLC.

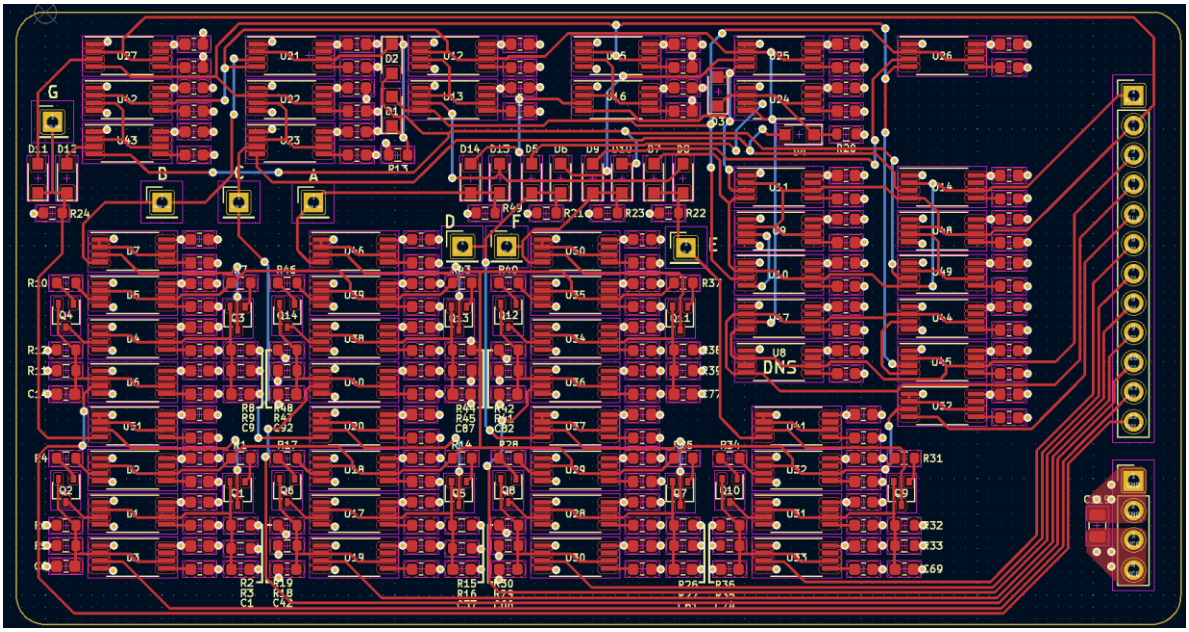


Figure 34. BCD to 7Segment PCB

8.3. MUX Layout

For the MUX PCB, due to the number of I/O, a 2x18 header was used to connect to the main board. As a result, this board does not follow the PCB interconnect template (Figure 32). Other then this issue, the board was relatively simple to layout due to its lower component count.

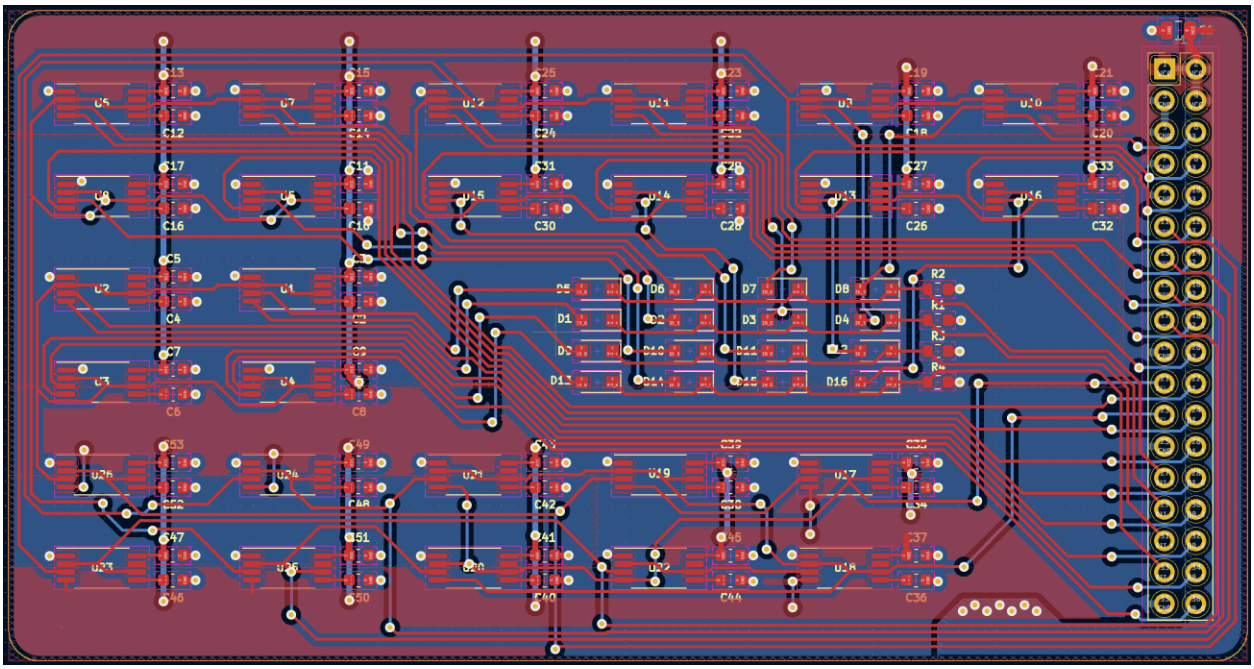


Figure 35. Multiplexer PCB

8.4. Selector & Clocks Layout

Figure 36 shows the PCB for the clock sources of the widget. All sources are configurable via potentiometers and labeled to help with debugging.

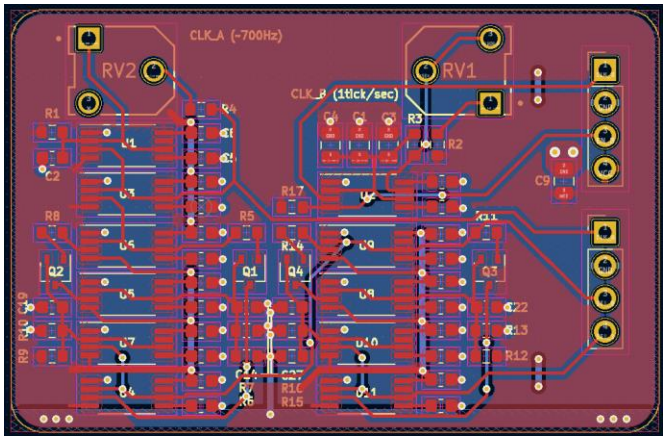


Figure 36. Selector and Clocks PCB

8.5. BCD Counter Layout

Due to the number of interconnects and feedback networks in the BCD counter, a 4-layer board was used to simplify layout. The inner layers include a solid ground and power plane that helped ease the layout process.

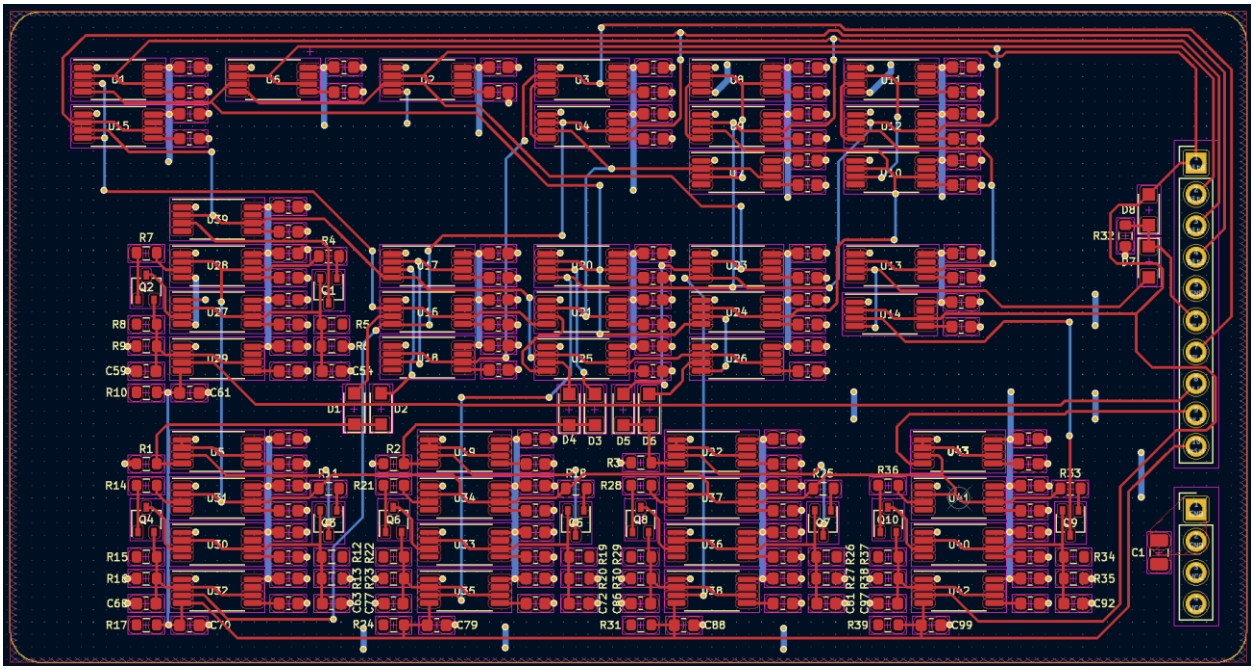


Figure 37. BCD Counter PCB

8.6. Buzzer Layout

The buzzer PCB is shown in Figure 38. It includes two labeled potentiometers to adjust the frequency of the PWM as well as the speaker on time.

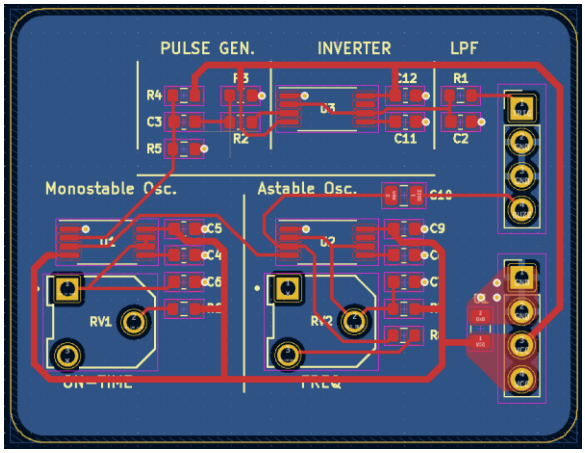


Figure 38. Buzzer PCB

8.7. 4-Bit Register Layout

The last board is the 4-bit register (Figure 39). This board uses a 2-layer PCB and includes no user inputs. The main board interconnects follow the interconnect template outlined at the beginning of this section.

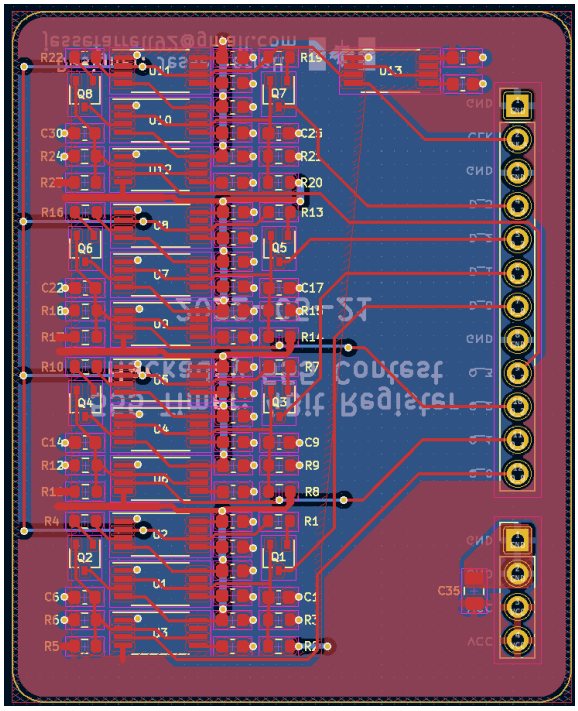


Figure 39. 4-Bit Register PCB

8.8. Power Layout

No power board was created, since powering this widget via battery will not be realistic. This board can be added in a later date using the pinout defined by the main board; or the user can power the device from an external power supply.

9. Final Testing & Validation

This section covers the testing and validation of the various modules of the timer widget. Each board will be evaluated individual to verify its behavior and to characterize its propagation delay. Due to the number of tests required to evaluate all boards, the documentation was recorded in a separate OneNote. In the future, the individual board validation may be added to this section, but for now only the final system validation will be documented.

9.1. System Validation

After validating the main add-in boards of the system such as the BCD counter, BCD to 7-segment decoder, and the clock sources, the system was assembled shown in Figure 40.

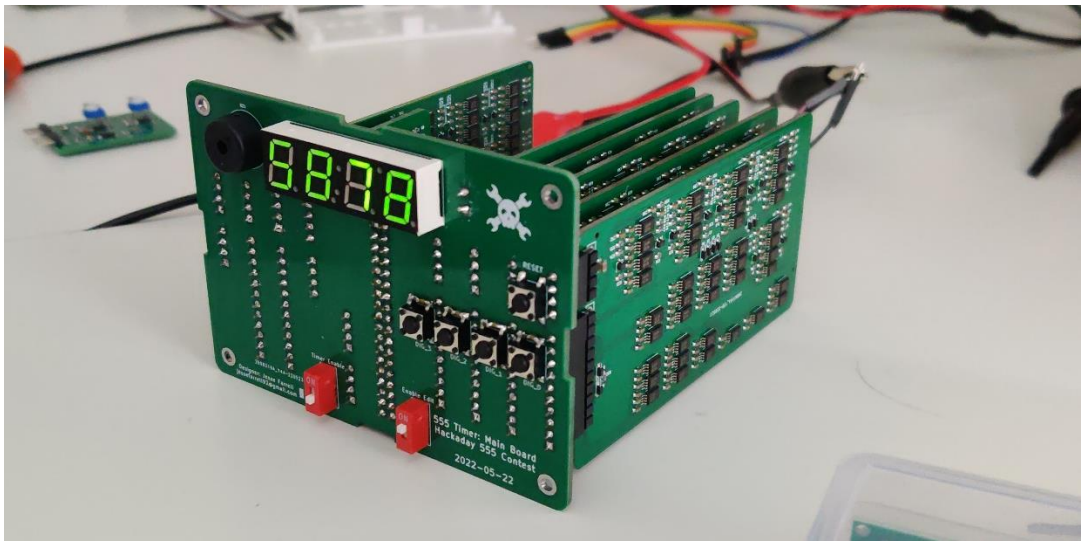


Figure 40. Initial System Validation

Initially, the push buttons and counting circuits seemed non-functional. Digits were displaying, but the system was not counting down. After some debugging the issue was found to be a design error in interfacing with the 7-segement display. During schematic capture, the various BCD digits were labeled expecting that the highest digit was the most significant. However, the 7-segment uses the opposite convention (Figure 41). As a result, digit 0 was incorrectly routed to D1, digit 1 to D2, and so on.

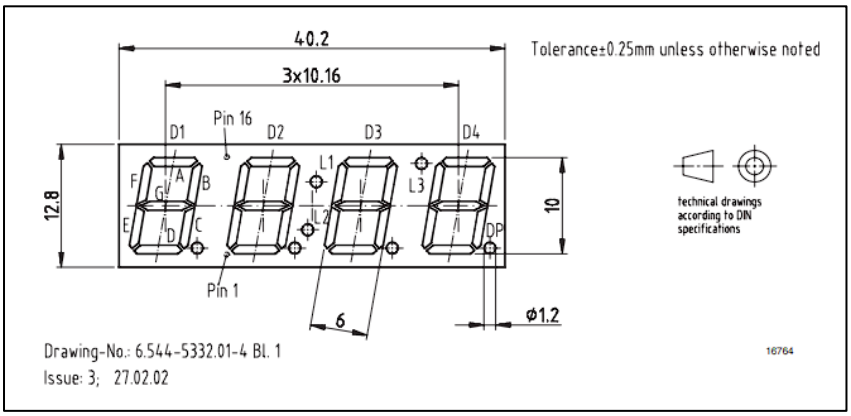


Figure 41. Most Significant Digits Illustration [7]

To correct this error, the 4 output traces of the 4-bit register were cut, and new connections were soldered in place. These new connections shorted the original output to the correct digit (Figure 42).

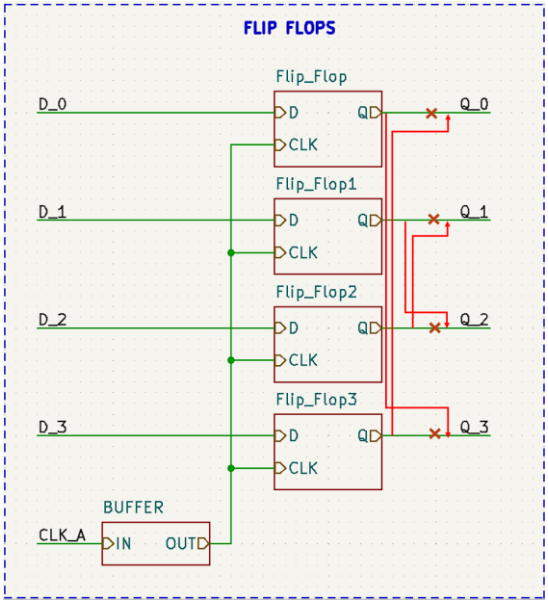


Figure 42. 4 Bit Register Corrected Schematic

The next error to resolve was the buzzer, which sounded when the most significant digit transitioned from 9 to 8 rather than 0 to 9. This behavior was caused by the design of the buzzer circuit. It was designed to trigger on the rising edge of an enable signal. The current circuit is triggered by the output clock of the most significant BCD counter circuit; this clock signal is pulled low when the digit transitions from 0 to 9 and is released on the next transition. As a result, the current circuit triggers on the wrong transition. This could be resolved by changing the inverter for a buffer (Figure 30).

Once this error was corrected the widget worked with some minor caveats covered in section 10, as well as some minor intermittent behavior caused by poor contact between the add in cards and the main board. A video of the final system can be found here. Note that in this demonstration video there is aliasing of the display caused by my camera's shutter rate.

10. Recommendations

This section briefly details the recommendations for future changes, most of which are a result of minor bugs and errors found in section 9.

The two boards that need to be modified for the widget to work are the 4-bit register, and the buzzer circuits. Both boards can either be fixed via "bodge" wires and by cutting existing traces on the PCB, or by revising the PCB designs. For the 4-bit register the flip-flop outputs need to be rearranged (Figure 42), and for the buzzer the inverter needs to be swapped for a buffer.

While testing the system it was noted that the counter is not very consistent and depends on the voltage that is applied to the widget. This bug is the result DC biasing of the charge/discharge capacitors, and possibly also a result of changing leakage currents of the 555. To resolve this the clock circuit should use an LDO to drop the input voltage to a known value. The system can then be configured for that voltage and would remain consistent over the input voltage range. Note that

logic level shifting can be accommodated by adding a 555 buffer to the output, since the buffer defined in section 4.3.3 has a trigger point at 0.9 volts.

A minor issue was a lack a debouncing on the user inputs. Debouncing capacitors can easily be added by handholding 100nF ceramics between the upper pins of each push button.

Another minor issue was inconsistent LED brightness. Instead of current limiting each digit, every element should have included a current limiting resistor. This guarantees a consistent current flow through each element rather than through each digit. In the current circuit a “1” is provided roughly the same current as an “8”.

In the current design the power board was left unpopulated. In the future a board should be fabricated to connect the widget to power either via a barrel jack or battery.

As mentioned earlier some intermittent behavior was noted at times. The issues were always resolved by shifting the various add-in cards, suggesting various intermittent connections. Ideally, better contact headers should be used or the add in cards can be soldered directly to the main board.

References

[1] Diodes Incorporated, “NE555_SA555_NA555,” 555 datasheet, Feb. 2021. [Online]. Available: https://www.diodes.com/assets/Datasheets/NE555_SA555_NA555.pdf. [Accessed: 08-Aug-2022].

[2] “555 oscillator tutorial - the astable multivibrator,” Basic Electronics Tutorials, 06-Sep-2013. [Online]. Available: https://www.electronics-tutorials.ws/waveforms/555_oscillator.html. [Accessed: 08-Aug-2022].

~~[3] REMOVED~~

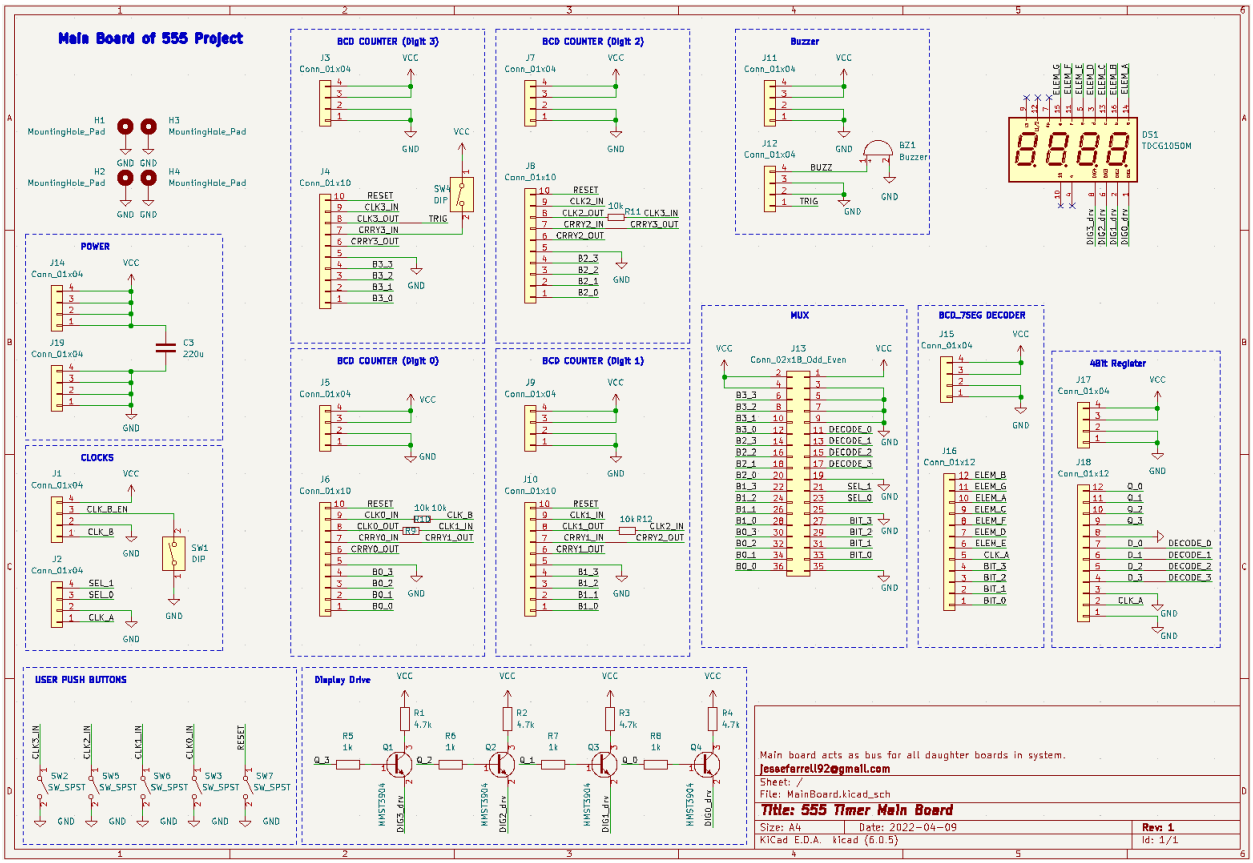
[4] Texas Instruments, “xx555 Precision Timers,” 555 datasheet, Sept. 2014. [Online]. Available: https://www.ti.com/general/docs/suppproductinfo.tsp?distId=10&gotoUrl=https%3A%2F%2Fwww.ti.com%2Flit%2Fgpn%2Fse555. [Accessed: 08-Aug-2022].

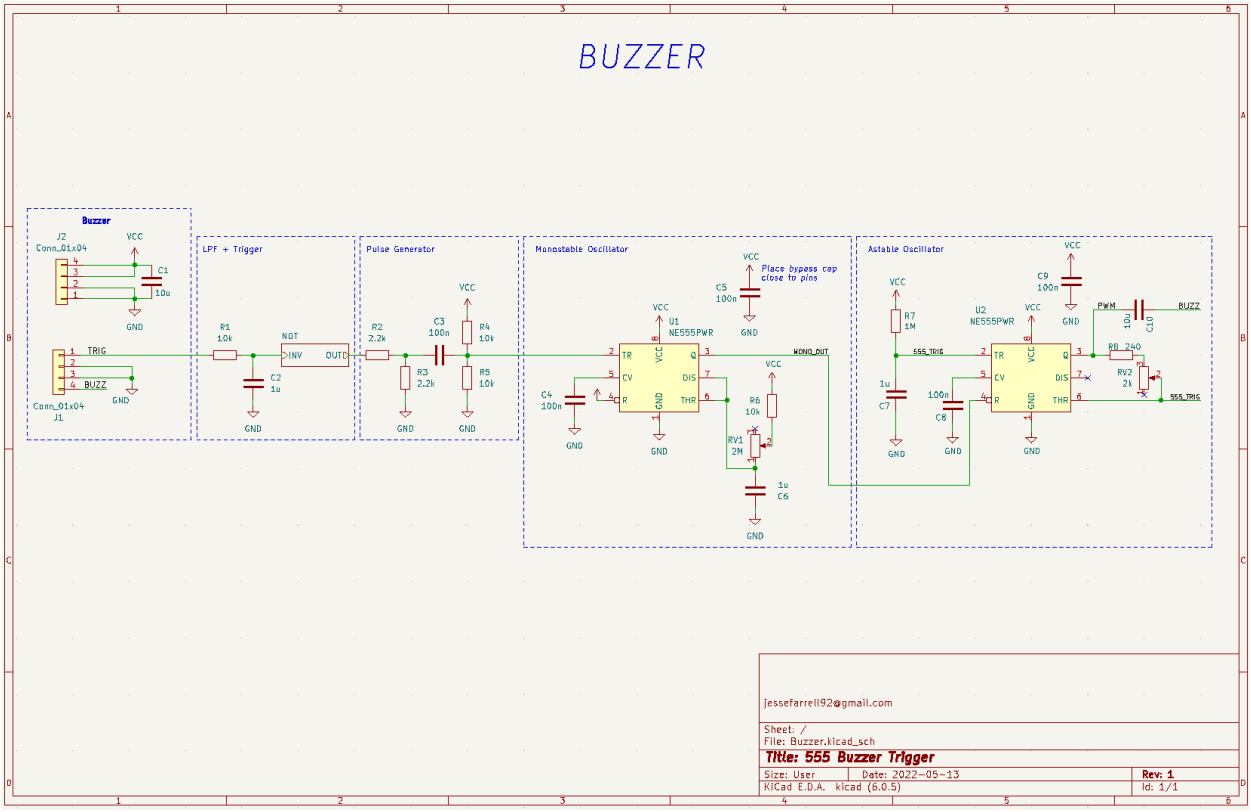
[5] Toshiba, “TOSHIBA diode Silicon Epitaxial Schottky Barrier Type – 1SS404,” diode datasheet, Mar. 2014. [Online]. Available: https://toshiba.semicon-storage.com/info/docget.jsp?did=3400&prodName=1SS404

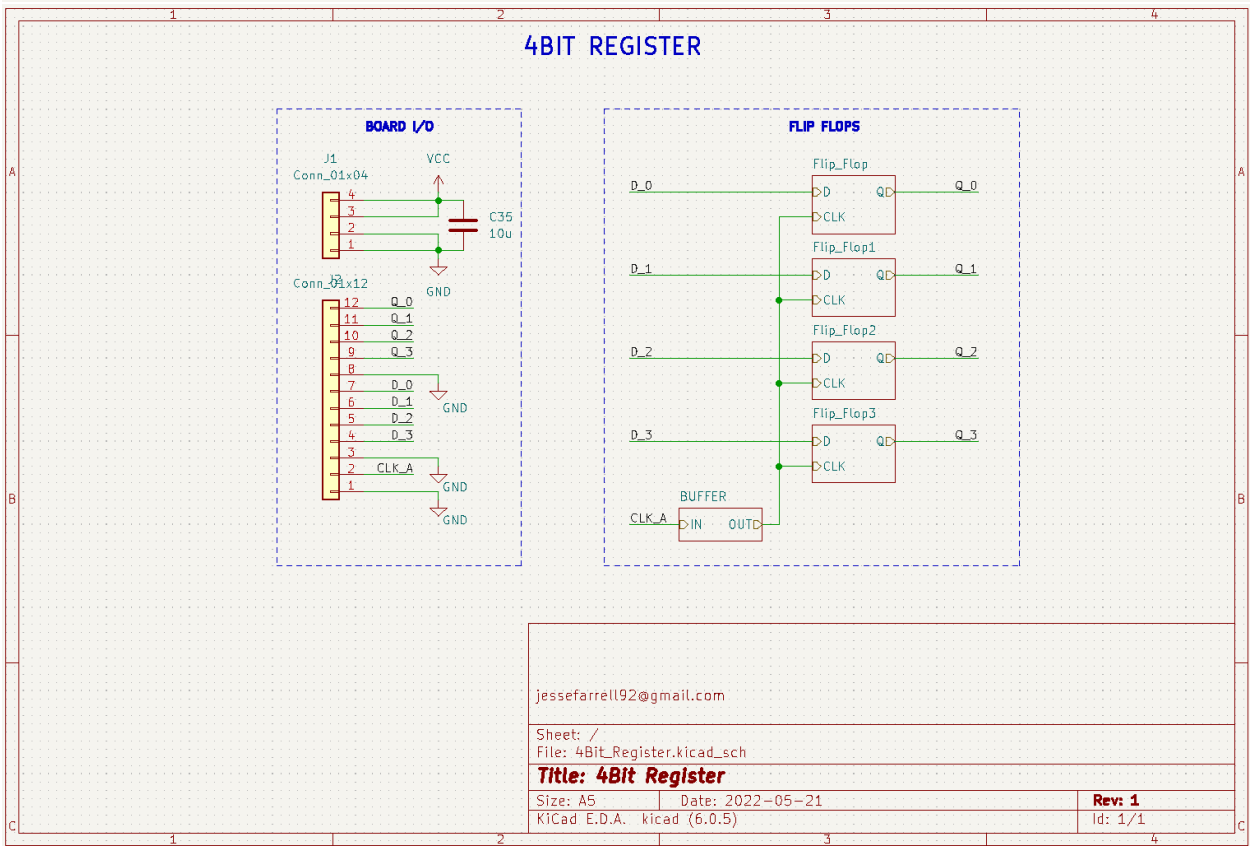
[6] Diodes Incorporated, “40V NPN Small Signal transistor,” bjt datasheet, Sept. 2019. [Online]. Available: https://www.diodes.com/assets/Datasheets/MMST3904.pdf

[7] Vishay, “Clock Display – TDCG1050M”, seven segment datasheet, Feb. 2018. [Online]. Available: https://www.vishay.com/docs/83180/tdcx10x0m.pdf

Appendix A – Top Level Schematics



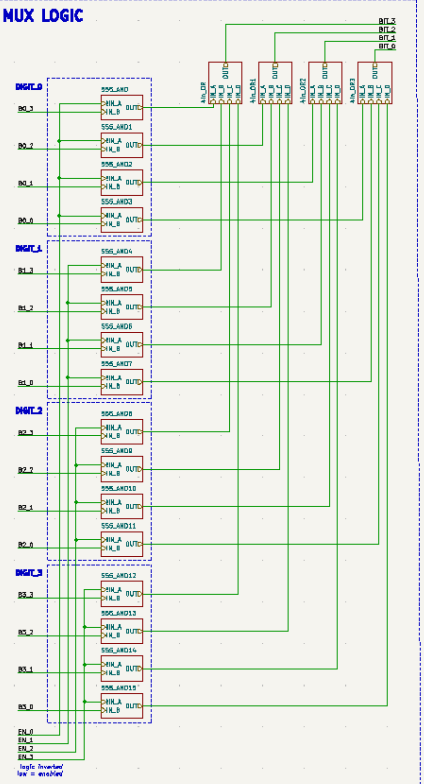

BUZZER
Buzzer
J2
Conn_01x04
VCC
C1
10u
GND
TRIG
BUZZ
Conn_01x04
J1
LPF + Trigger
R1
10k
NOT
INV
OUTD
C2
1u
Pulse Generator
R2
2.2k
C3
100n
R4
10k
R3
2.2k
R5
10k
Monostable Oscillator
U1
NE555PWR
TR
CV
R
VCC
GND
Q
DIS
THR
C5
100n
Place bypass cap close to pins
C4
100n
R6
10k
RV1
2M
C6
1u
MONO_OUT
Astable Oscillator
R7
1M
555_TRIG
1u
C7
100n
C8
U2
NE555PWR
C9
100n
PWM
10u
C10
BUZZ
RB 240
RV2
2k
jessefarrell92@gmail.com
Sheet: /
File: Buzzer.kicad_sch
Title: 555 Buzzer Trigger
Size: User
Date: 2022-05-13
Rev: 1
KiCad E.D.A. kicad (6.0.5)
Id: 1/1


4BIT REGISTER
BOARD I/O
J1
Conn_01x04
VCC
C35
10u
GND
J2
Conn_01x12
Q_0
Q_1
Q_2
Q_3
D_0
D_1
D_2
D_3
CLK_A
GND
FLIP FLOPS
Flip_Flop
Flip_Flop1
Flip_Flop2
Flip_Flop3
D
CLK
QD
BUFFER
IN
OUTD
jessefarrell92@gmail.com
Sheet: /
File: 4Bit_Register.kicad_sch
Title: 4Bit Register
Size: A5
Date: 2022-05-21
Rev: 1
KiCad E.D.A. kicad (6.0.5)
Id: 1/1

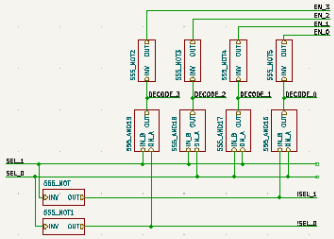
555 MUX

HEADER I/O (+PWR)

MUX LOGIC



MUX Decoder

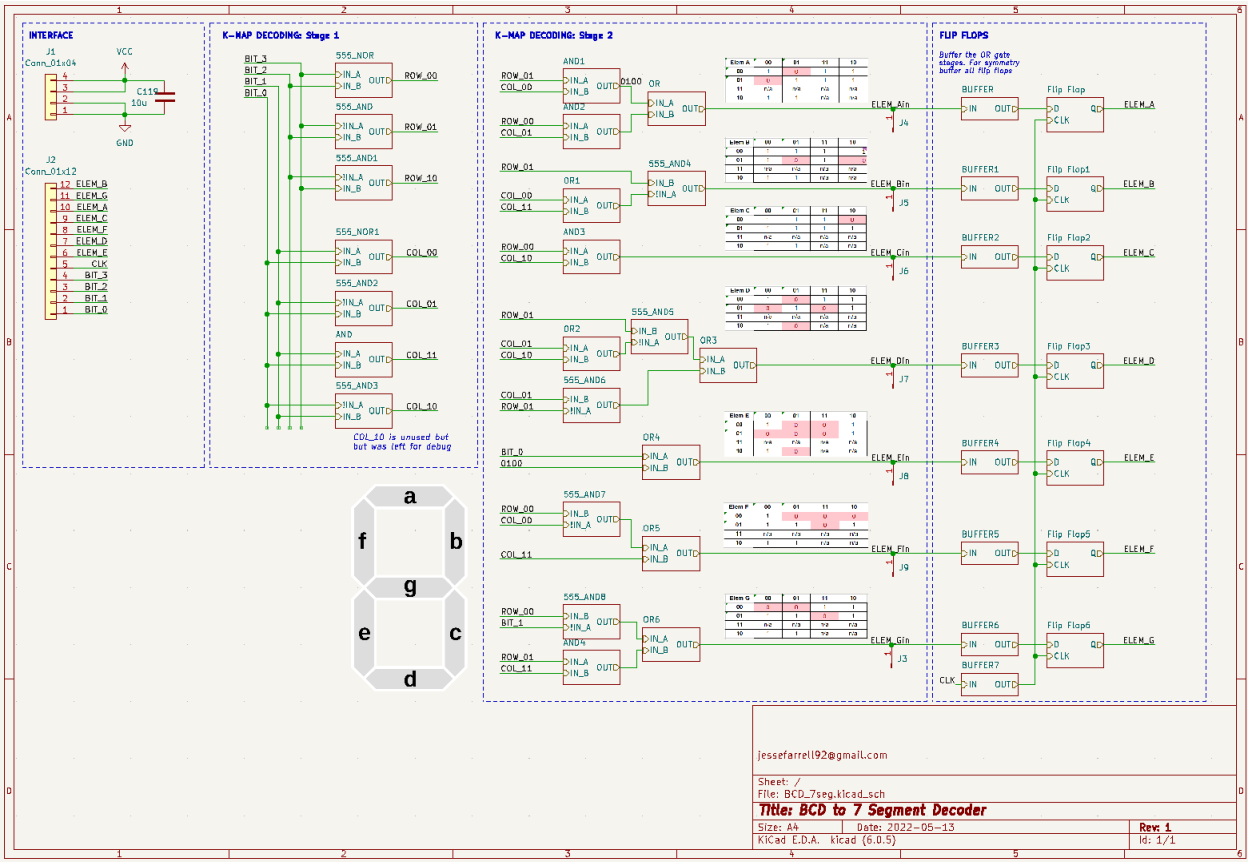


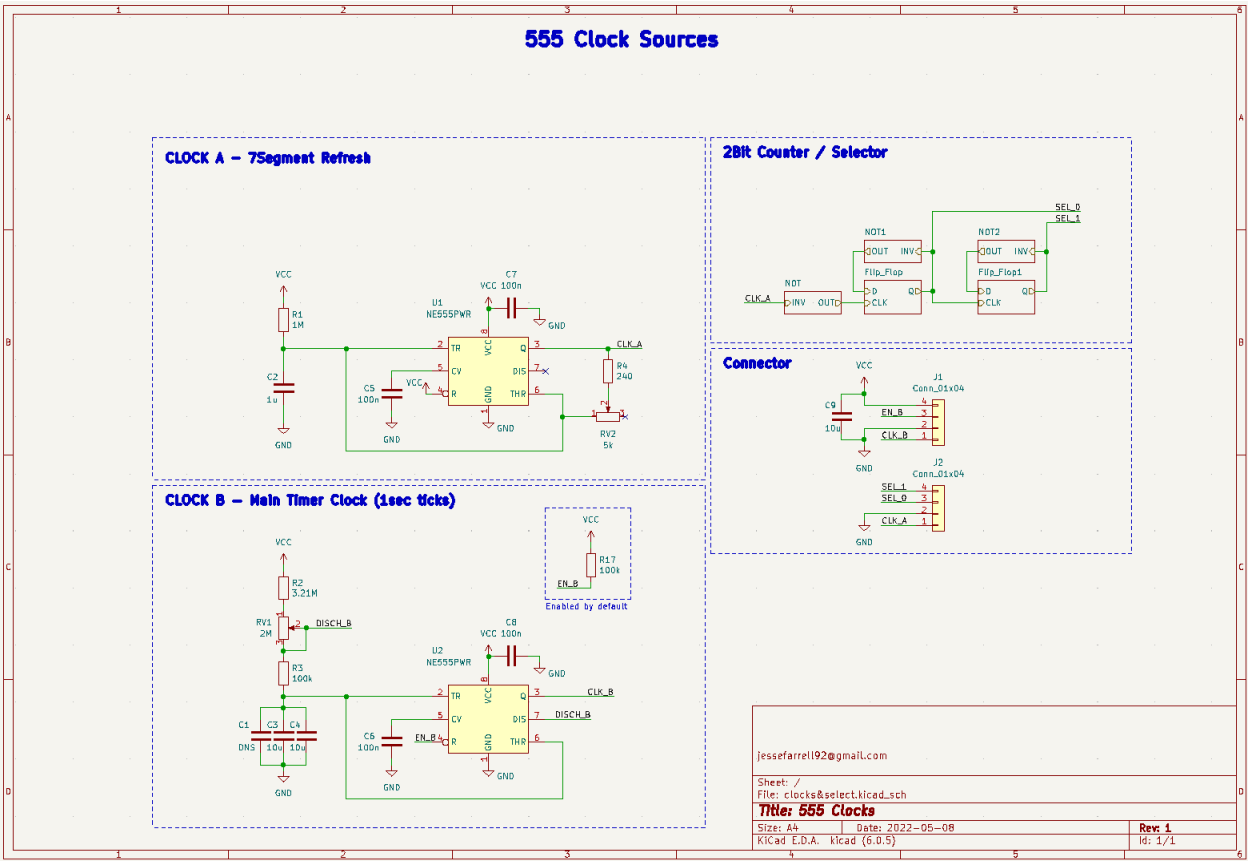
jesseferrell92@gmail.com

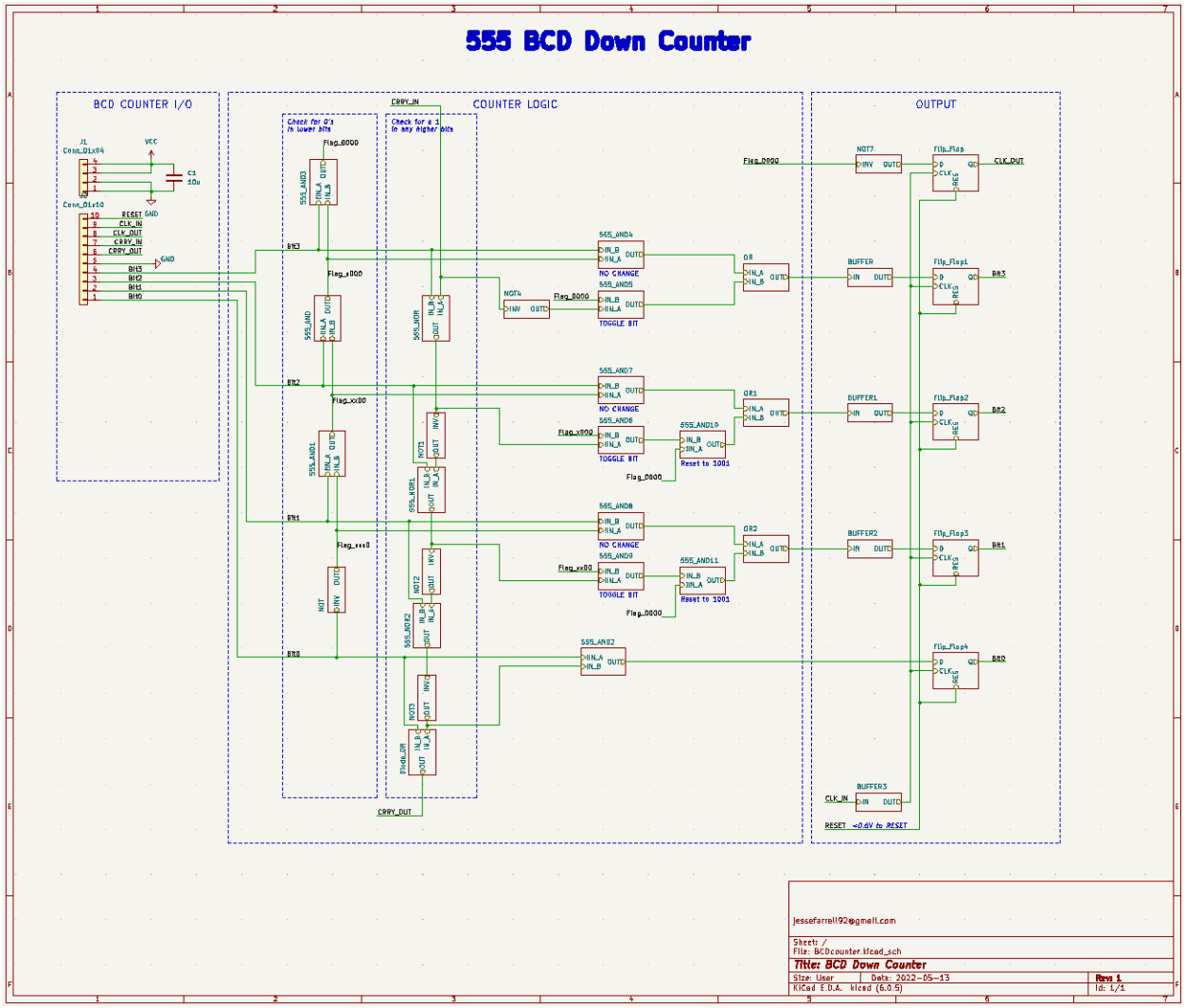
Sheet: /
File: MUX.kicad_sch

Title: 555 MUX

Size: User	Date: 2022-05-13	Rev: 0
KiCad E.D.A. Kicad (6.0.5)		Id: 1/1


INTERFACE
K-MAP DECODING: Stage 1
K-MAP DECODING: Stage 2
FLIP FLOPS
COL_10 is unused but but was left for debug
jessefarrell92@gmail.com
Sheet: /
File: BCD_7seg.kicad_sch
Title: BCD to 7 Segment Decoder
Size: A4
Date: 2022-05-13
Rev: 1
KiCad E.D.A. kicad (6.0.5)
Id: 1/1


555 Clock Sources
CLOCK A – 7Segment Refresh
2Bit Counter / Selector
Connector
CLOCK B – Main Timer Clock (1sec ticks)
Enabled by default
jessefarrell92@gmail.com
Sheet: /
File: clocks&select.kicad_sch
Title: 555 Clocks
Size: A4
Date: 2022-05-08
Rev: 1
KiCad E.D.A. kicad (6.0.5)
Id: 1/1


555 BCD Down Counter
BCD COUNTER I/O
COUNTER LOGIC
OUTPUT
CRRY_IN
CRRY_OUT
Check for 0's in lower bits
Check for a 1 in any higher bits
Flag_0000
Flag_x000
Flag_xx00
Flag_xxx0
Flag_xx00
555_AND3
555_AND
555_NOR
555_AND1
555_NOR1
555_AND2
555_AND4
555_AND5
555_AND7
555_AND6
555_AND8
555_AND9
555_AND10
555_AND11
555_AND12
NOT4
NOT5
NOT6
NOT7
NOT
NOT2
NOT3
OR
OR1
OR2
BUFFER
BUFFER1
BUFFER2
BUFFER3
Flip_Flop
Flip_Flop1
Flip_Flop2
Flip_Flop3
Flip_Flop4
NO CHANGE
TOGGLE BIT
Reset to 1001
Bit3
Bit2
Bit1
Bit0
CLK_IN
CLK_OUT
RESET
VCC
GND
C1
10u
RESET <0.6V to RESET
jessefarrell92@gmail.com
Sheet: /
File: BCDcounter.kicad_sch
Title: BCD Down Counter
Size: User
Date: 2022-05-13
Rev: 1
KiCad E.D.A. kicad (6.0.5)
Id: 1/1