An Implementation of "Etch-A-Sketch" Using Digital Logic

By Casey He
&
Arthur Christopher Watkins

The Cooper Union for the Advancement of Science & Art
ECE-150, Digital Logic Design
Prof. Harwayne-Gidansky

3 January 2016

Contents

List of Figures

1 Abstract

Etch-A-Sketch is a wildly popular children's drawing toy released in the early 1960s by the Ohio Art Company. The toy itself provides two knobs as mechanical inputs to the device. The user can turn the knobs with variable intensity at different points in time to move the pen and produce a picture, which can then be erased by vigorously shaking the toy. This project explores an implementation of this toy using digital logic. Instead of two knobs that can be manipulated by the user to produce a picture, four mechanical button switches are used to select the direction of pen movement: left, right, up, and down. Three other mechanical button switches are used to advance the pen in the selected direction of movement, to place a mark at the current pen position, and to erase the entire picture. All picture information is stored in memory and output to a 14 LED wide by 10 LED long LED matrix display. Work to be completed for future implementations may include features such as a more realistic mechanical system for erasing the picture (e.g. physically shaking an analog control) or a more realistic mechanical knob system using potentiometers and a circuit that measures rate of resistance change for speed of pen movement.

2 Design Considerations

During the design phase, a number of considerations were accounted for, regarding the display, the method of memory access, component interfacing with memory, the method of input, and the something else. Here these considerations are described in detail.

2.1 The Display

The display needed to be large enough such that a large picture could be drawn yet small enough such that a protective border could allow for the detection of invalid pen positions. It was decided that a 14 LED by 10 LED matrix was to be constructed. The final constructed display can be seen below in Figure 2.1.1 as well as in use in Figure 2.1.2.

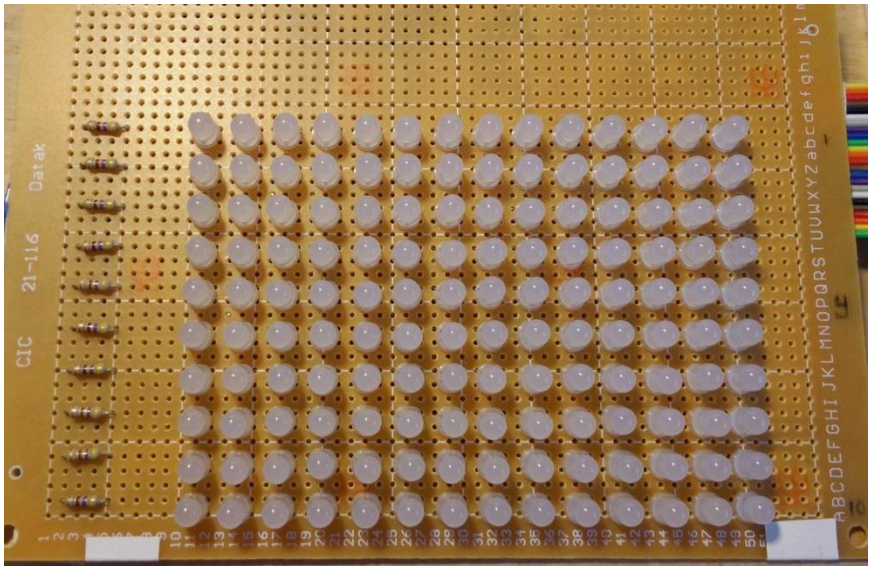

Fig. 2.1.1: 140 LED display

Because the display needed to be large, it was determined that not all LEDs could emit light simultaneously due to a wastefully substantial draw of power. Therefore, it was decided that only one LED should emit light at a time. In addition to this, creating an LED

matrix that could allow a single LED to be addressed at a time made display construction much easier because only the anodes of each row of LEDs had to be soldered together, and the cathodes of each column of LEDs had to be soldered together.

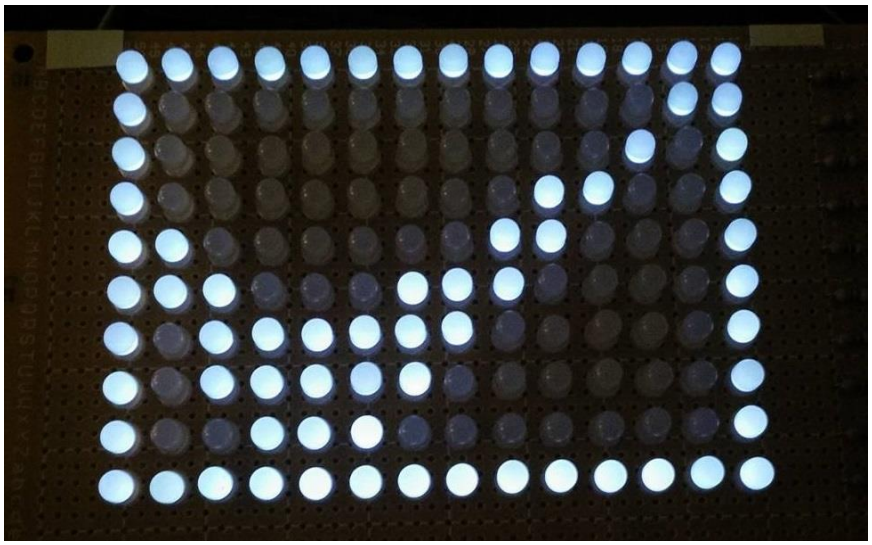

Fig. 2.1.2: 140 LED display used to display a checkmark

These rows and columns of LEDs interface with an X demultiplexer, which takes an X address and an on/off signal, as well as a Y demultiplexer, which takes a Y address and an on/off signal. When the on/off signal of the X demultiplexer is logic HIGH and the on/off signal of the Y demultiplexer is logic LOW, the corresponding LED will turn on. However, when the on/off signals are input in any other combination, the LED will not turn on.

2.2 The Memory Cycle

Cycling through 140 addresses of memory was necessary to displaying all 140 "pixels" of the display. In order to accomplish this memory cycle, two up-counters were constructed using logic gates and JK flip flops. One up-counter progressed from 1 to 14 (corresponding to the X coordinate of an LED or memory address), while the other up-counter progressed from 1 to 10 (corresponding to the Y coordinate of an LED or memory address). The finite state machine diagrams for each counter are shown below:

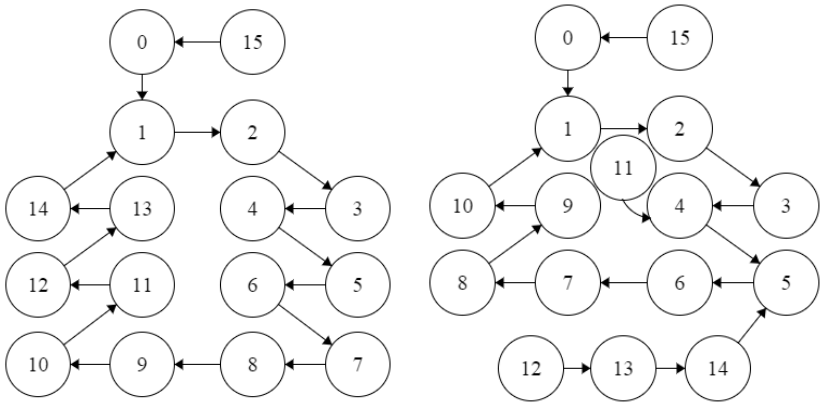

Fig. 2.2.1: 1–14 counter (left), 1–10 counter (right) finite state machines

In order for the display to refresh at a rate such that each LED could seem to be on without any flickering, the entire display needed to be updated at least 60 times per second, meaning that 140 LEDs needed to be updated in less than one-sixtieth of a second. This meant that the timer controlling the 1-14 counter and 1-10 counter X and Y address cycles needed to operate at a frequency greater than 8400Hz, as shown in the calculation below:

$$140 \cdot 60\text{Hz} = 8400\text{Hz}$$

A 555 timer set up in astable mode was selected such that two 220Ω resistors and a 0.1µF capacitor were used. This resulted in the following calculation for memory cycle rate:

$$\text{f} = (\ln(2) \cdot 0.1\mu\text{F} \cdot (220\Omega + 2 \cdot 220\Omega))^{-1} \approx 21{,}859\text{Hz}$$

This resultant frequency of memory cycling corresponds to a display refresh rate of approximately one-hundred-fortieth of this value:

$$\text{f} \approx 21{,}859\text{Hz} \div 140 \approx 156\text{Hz}$$

Because both the memory and display operate at such high frequencies, multiple decoupling capacitors were placed between the power and ground rails of the circuit to eliminate noise.

2.3 The Memory Bus

In order to properly interface the circuit components with memory, the construction of an address bus was necessary because different addresses were being accessed by different components at different times. Multiple circuit components must be able to interface with memory, but they cannot interfere with each other. Therefore, tri-state buffers were used to create an address bus. One tri-state buffer was always in the high-impedance state while the other was in the low-impedance, true output state. The outputs of the tri-state buffers were tied together and input to the address pins of memory respectively.

2.4 The Directional Input

Because the method of directional input is asynchronous to the timing of the rest of the synchronous circuit, a direction register was necessary to store the asynchronous input and provide it synchronously to the rest of the circuit. The input is selected using a tri-state buffer which outputs one of four binary numbers represented by logic values of LOW or HIGH: 00 (left), 01 (up), 11 (right), and 10 (down). Once these values have been input into two SR latches, the SR latch register is available for the rest of the circuit to the use the direction.

2.5 The Boundary Logic

In order to prevent invalid pen positions, boundary logic had to be implemented. When the pen's X position is 0 or 15, the pen's position is reset back to the (1, 1) position. When the pen's Y position is 0 or 11, the pen's position is reset back to the (1, 1) position. This is accomplished using a variety of 2-input and 4-input AND and OR gate integrated circuits.

3 Functional Block Diagram

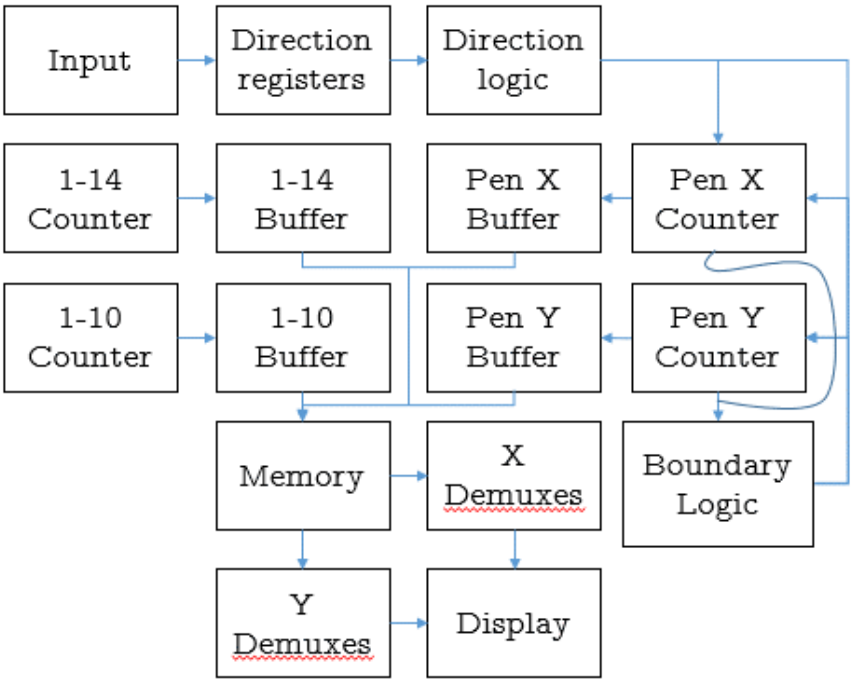

Fig. 3.1: Functional block diagram

4 Logic Diagrams

In order to control the buffers appropriately and interface the inputs with the other circuit components, vital digital logic had to be implemented. Below are some important logic diagrams used within the digital logic implementation of "Etch-a-Sketch".

In Figure 4.1, the "Pen" input, normally logic LOW, outputs *true* to the display address buffer disable pin "DispBuff" and *complement* to the pen address buffer disable pin "PenBuff". Upon the depression of the button, Pen goes logic HIGH and switches the active address buffer to the pen address buffer.

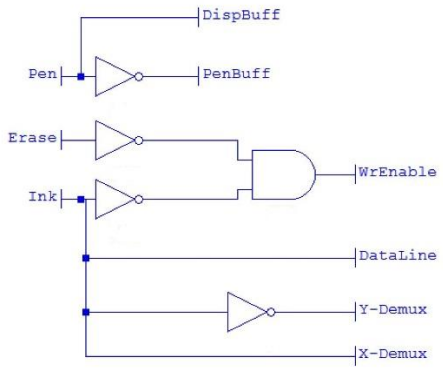

Fig. 4.1: Draw, erase, and buffer control logic

Upon the depression of the "Erase" button, normally logic LOW, Erase goes HIGH thus outputting a LOW to the memory write enable pin, which enables the memory to be written. Upon the depression of the "Ink" button, normally logic LOW, Ink goes HIGH thus outputting a logic HIGH value to the data bit pin of the memory, as well as a logic HIGH value to the X demultiplexer signal "X-demux". Logic LOW values are output to the write enable pin of memory as

well as the Y demultiplexer signal “Y-demux”. This writes a HIGH value to the memory.

In Figure 4.2, the “Pen” input, normally logic LOW, outputs logic LOW values to “XPenCLK” and “YPenCLK”, the clock pins of the pen’s X and Y up/down counters. Upon depression of the Pen button, Pen goes logic HIGH and outputs a clock pulse to the X or Y pen up/down counter clock, depending on the direction of movement of the pen (given by the 2 bit binary number BA, B being the most significant bit and A being the least significant bit).

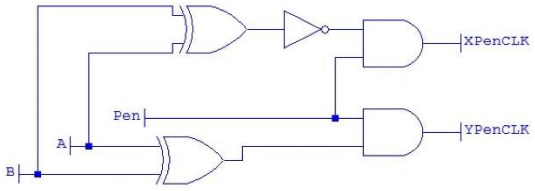

Fig. 4.2: Pen counter control logic

When BA = 00 or BA = 11, the X pen up/down counter clock goes HIGH. When BA = 01 or BA = 10, the Y pen up/down counter clock goes HIGH. B also goes to the up/down pin of both counters. When B is 0, either the X or the Y up/down counter is decremented. When B is 1, either the X or the Y up/down counter is incremented.

In order to ensure that invalid pen positions remain unexpressed, boundary logic is implemented to reset the pen position back to X=1, Y=1. This logic is illustrated in Figure 4.3 below:

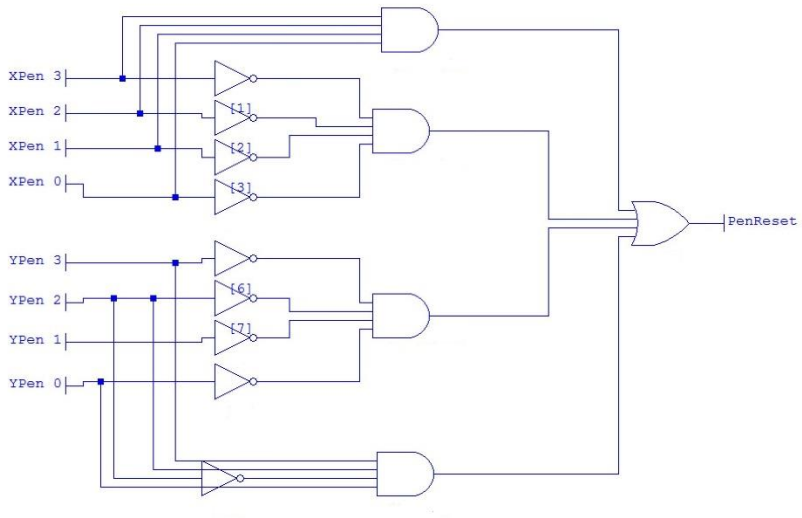

Fig. 4.3: Pen counter reset logic

When all 4 bits of the pen's X or Y counter are logic LOW, the pen's location is reset, i.e. "PenReset" goes logic HIGH (corresponding to X=0 and Y=0). When all 4 bits of the pen's X counter are logic HIGH, the pen's location is reset (corresponding to X=15). When all 4 bits, except for the second least significant bit, of the pen's Y counter are logic HIGH, the pen's location is reset (corresponding to Y=11).

5 Annotated Circuits

Pictured below are annotated pictures of all circuitry.

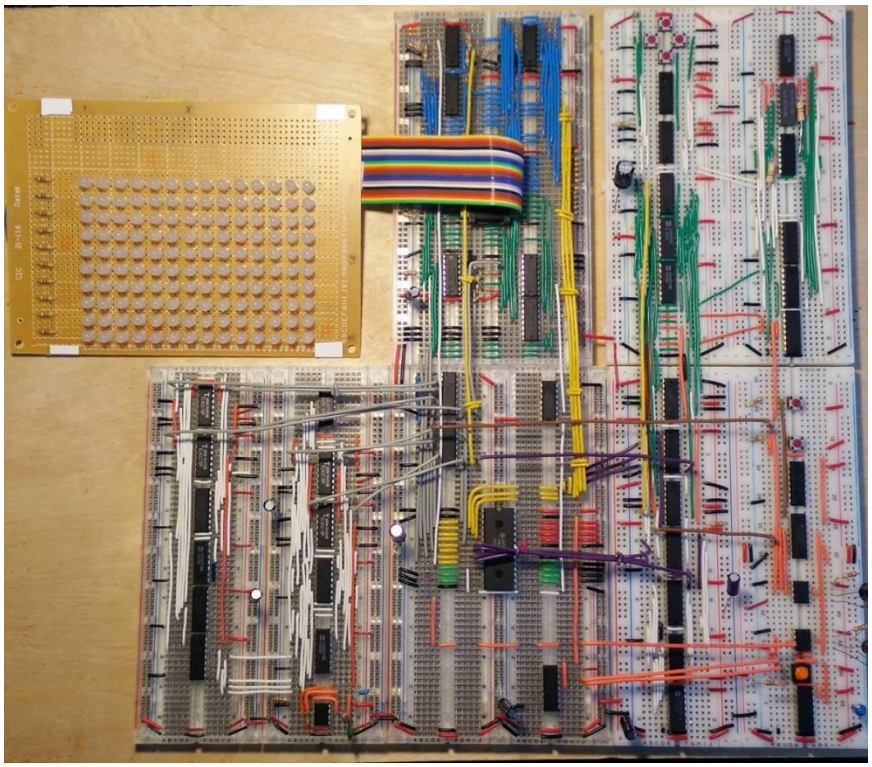

Fig. 5.1: Full implementation circuitry

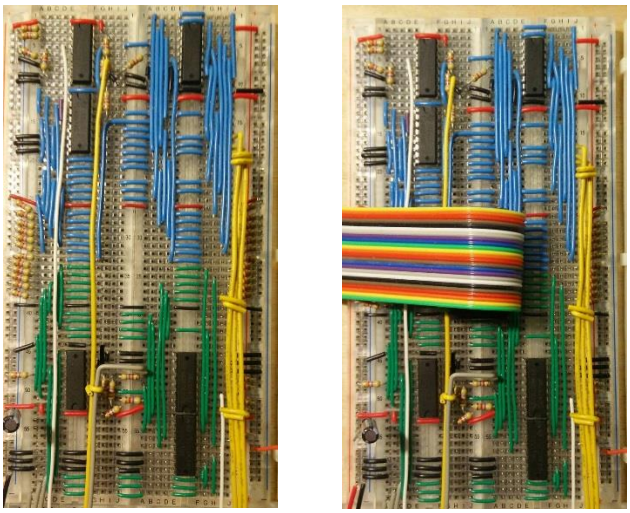

Fig. 5.2: Display X demultiplexers (*top left of both*), display Y demultiplexers (*top right of both*) with and without ribbon connector

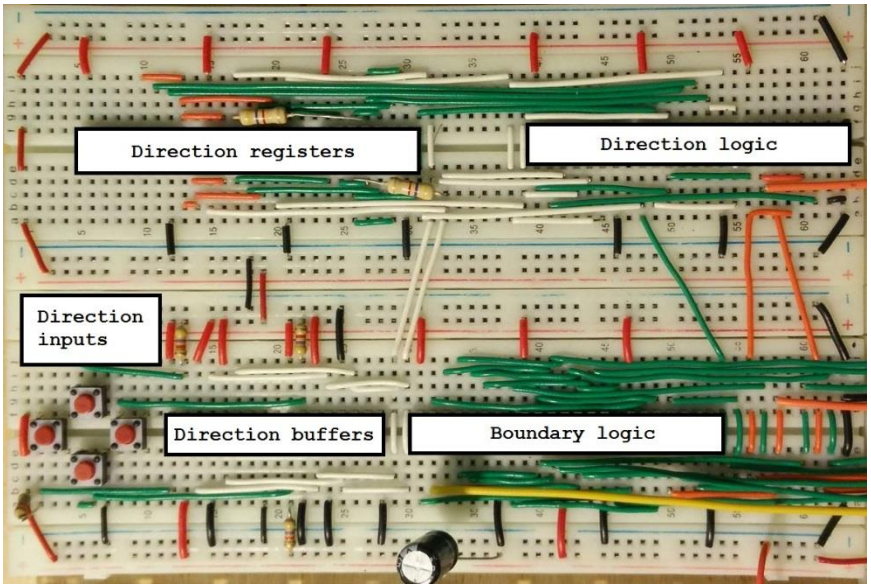

Fig. 5.3: Direction inputs, direction registers, direction logic, and boundary logic

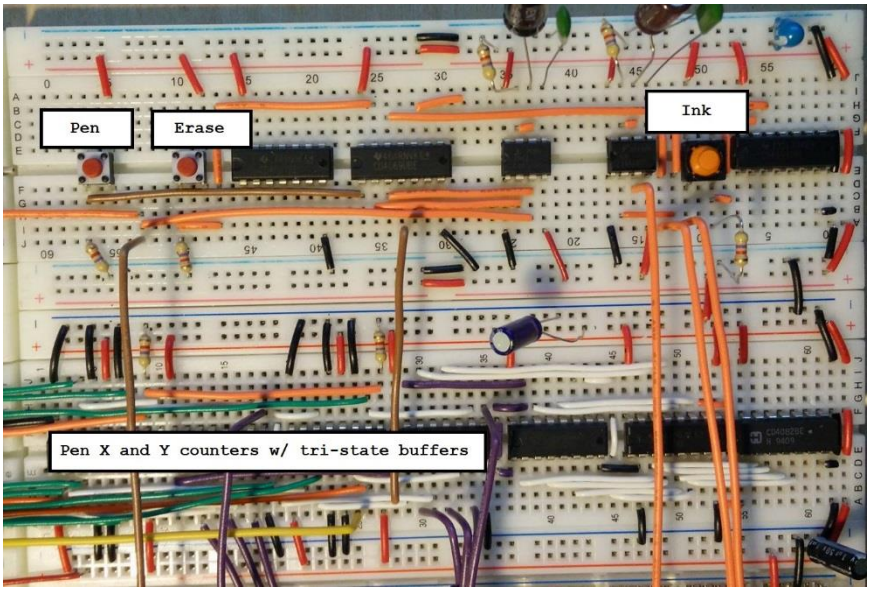

Fig. 5.4: Mechanical button switch inputs (debounced using monostable 555 timer configuration) with pen counters and tri-state buffers

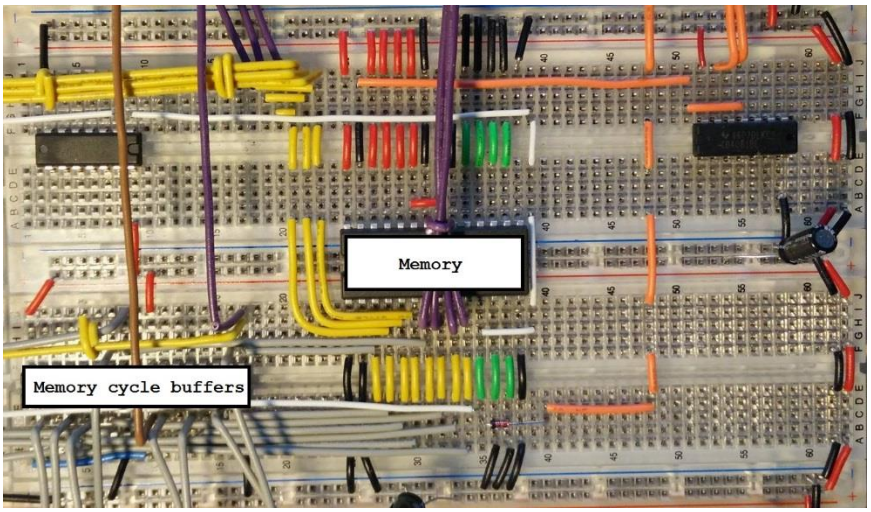

Fig. 5.5: Memory cycle buffers (inputs from 1–14 counter and 1–10 counter) and memory

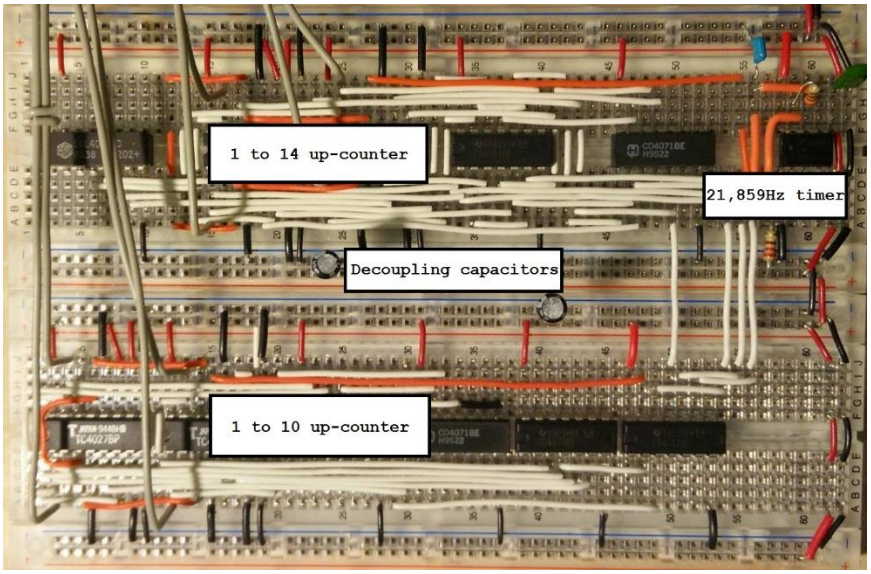

Fig. 5.6: 1–14 counter, 1–10 counter, decoupling capacitors, and timer clocking the 1–14 counter (clocks the 1–10 counter every 14 counts)