Tiffany Q. Liu
April 4, 2011
CSC 270
Lab #9

# Lab #9: Observing an Endless Loop on the Oscilloscope

## Introduction

The purpose of this lab was to write and assemble an endless loop program to enter into the 6811 trainer and use the oscilloscope to observe the cycles involved.

## Materials

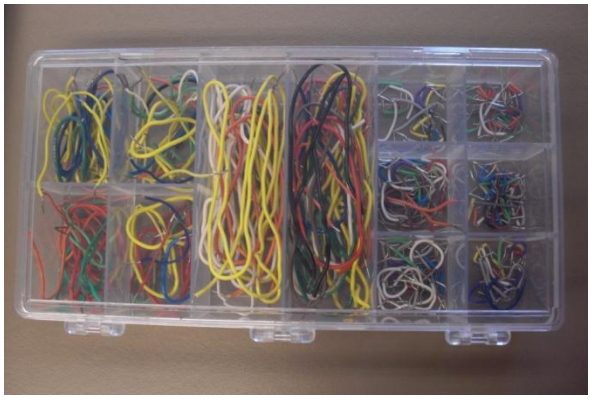

Figure 1. Wiring Kit.

Figure 2. 6811 Microprocessor Kit (Taken from D.Thiebaut).

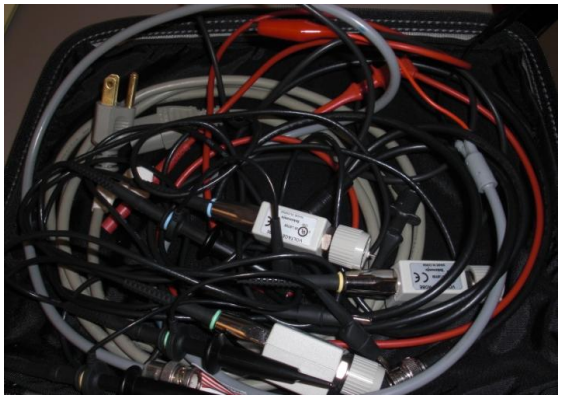

Figure 3. Oscilloscope Cables.

Figure 4. Tektronix Oscilloscope

**Part 1**

To start, we wrote and assembled a program that would increment a variable x by 1 inside an endless loop:

```
                    ;--- data section ---
                           ORG   0010
0010 00             x      FCB   0          ; 0 is stored at address 10

                    ;--- code section ---
                           ORG   0000
0000 4C             LOOP:  INCA             ; ACCA <- ACCA + 1
0001 97 10                 STAA  x          ; x <- ACCA
0003 20 FB                 BRA   LOOP       ; branch to LOOP
```

The second byte of the opcode for the BRA instruction was determined by taking the 2's complement of the next address byte, in this case 05, which gives us the result FB. We do this because when FB is added to the next address byte 05, we get 00, which is the memory address of where we want to branch to.

Next, we created a table that gave a cycle by cycle description of the execution of the instruction:

| | Cycle | Cycle Name | Address | Data | (LIR)' | LIR |
|---|---|---|---|---|---|---|
| **INCA** | 1 | IF | 0000 | 4C | 0 | 1 |
| | 2 | OF/IE | 0001? | ? | 1 | 0 |
| **STAA** | 3 | IF | 0001 | 97 | 0 | 1 |
| | 4 | OAF | 0002 | 10 | 1 | 0 |
| | 5 | OS/IE | 0010 | XX | 1 | 0 |
| **BRA** | 6 | IF | 0003 | 20 | 0 | 1 |
| | 7 | OF | 0004 | FB | 1 | 0 |
| | 8 | IE | 0005? | ? | 1 | 0 |

Then we entered the opcodes into the 6811 trainer and executed the program. Since the program we entered is just incrementing x inside an infinite loop, and we did not program the 6811 to output anything to the screen, the trainer display stays blank while the program continues to run.

**Part 2**

In order to see the cycles in real-time, we connected the 6811 trainer to the oscilloscope by connecting the grounds to GND located on the lower block and connecting one probe to the E clock signal (located on the upper block) and the other probe to the LIR signal (also located on the upper block). When we did this, we got the following screen capture:

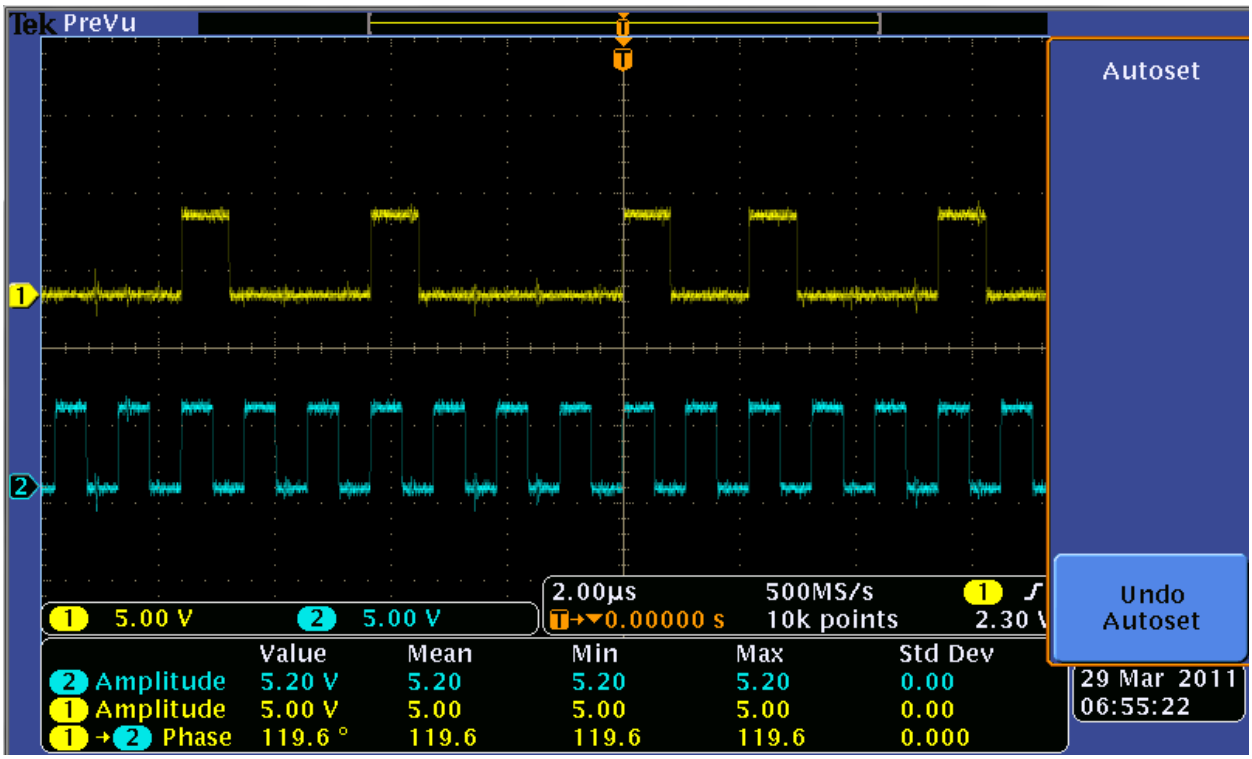


Figure 5. Screen Capture of the E Clock Signal (Blue) and the LIR Signal (Yellow).

Since (LIR)' goes low for the first opcode byte fetch, the LIR signal indicates when the first opcode byte fetch occurs when the signal goes from low to high. Thus, from the LIR signal, we can see when a new instruction begins.

To calculate the average number of instructions executed by the 6811 over time, we used the oscilloscope to measure one period of the LIR signal since we knew one period of the LIR signal represented one execution of the loop, which consists of 3 instructions. The measured period of the LIR signal was 8.65µs. Knowing this and the fact that 3 instructions were executed in that period, we determined that the average number of instructions executed by the 6811 was equal to 3 instructions divided by 8.65µs, which is 346,820 instructions per second or 0.347 MIPS.

**Part 3**

We then connected a third probe to the R/W' (read/write) signal (located on the upper block) and obtained the following screen capture:

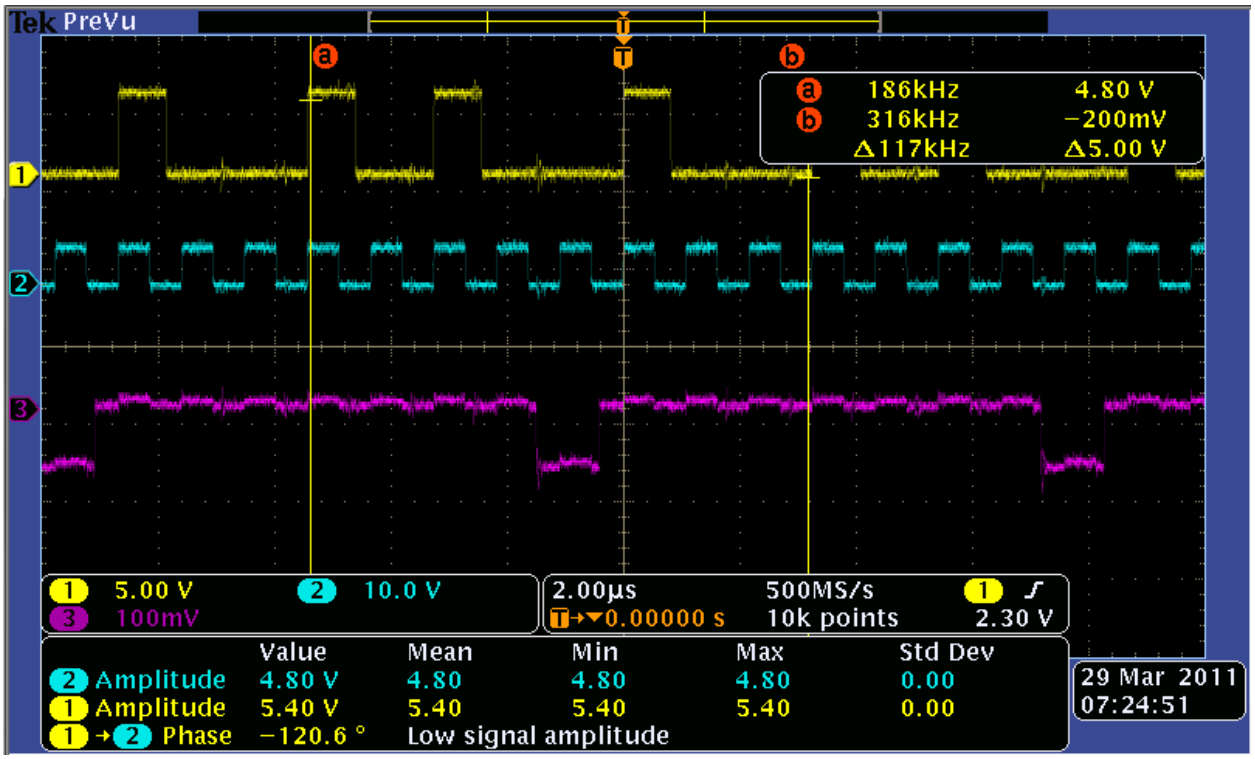


Figure 6. Screen Capture of the E Clock Signal (Blue), LIR Signal (Yellow), and the R/W' Signal (Purple).

The R/W' signal generates a 1 when the processor is reading from memory and generates a 0 when the processor is writing to memory. The signals between the two yellow cursor lines in Figure 6 represent a single iteration of the loop. The R/W' signal between the cursors stays high (1) throughout most of the loop iteration except for around the middle, where the signal goes low (0). This makes sense since that is when we programmed the processor to store the value in Accumulator A to x.

To calculate the average number of time the loop runs in one second, we measured the frequency of the LIR signal, which was 117kHz. This indicates that on average, the loop runs about 117,000 times per second.

**Part 4**

Then, we connected a fourth probe to one of the address bus lines. We chose to connect the probe to the least significant bit of the address bus (A0) since the addresses appeared to alter between an even value and an odd value fairly frequently, which would be reflected in the least significant bit changing between 0 and 1. When we connected the probe to the 6811 kit, we obtained the following:

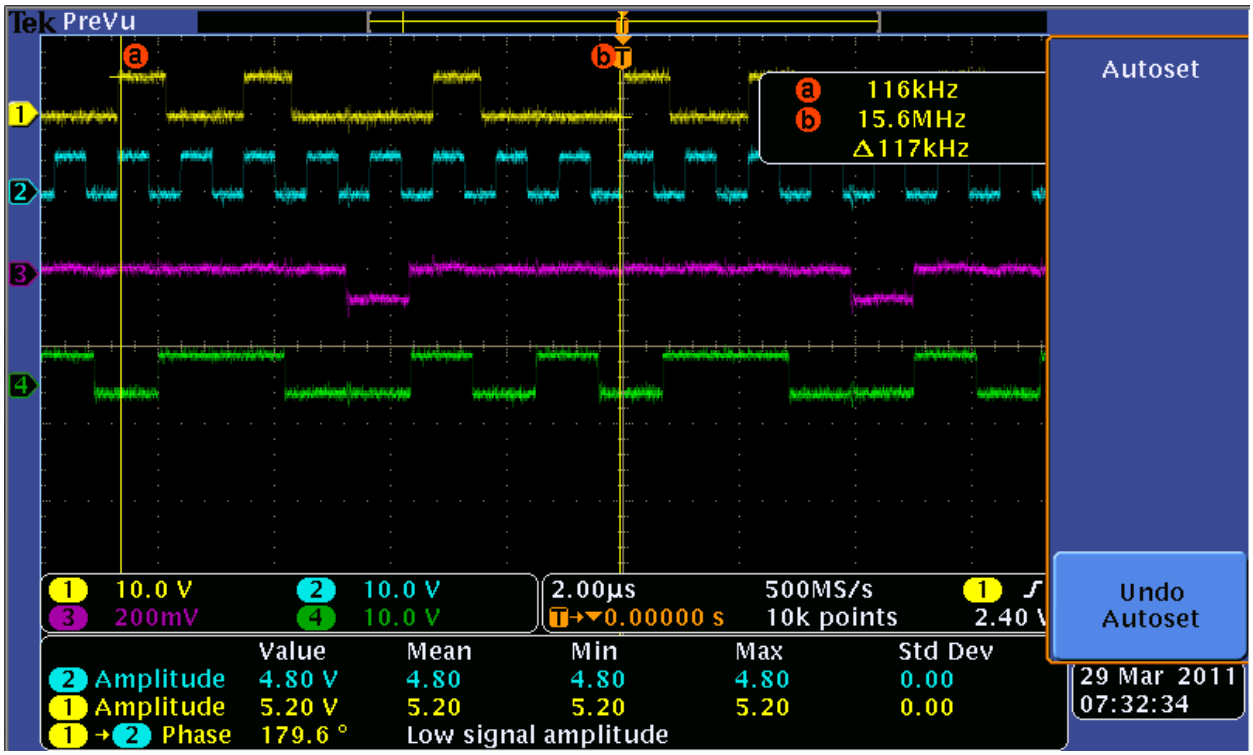


Figure 7. Screen Capture of the E Clock Signal (Blue), LIR Signal (Yellow), the R/W' Signal (Purple), and the A0 Signal (Green).

Again, the cursors in Figure 7 indicate the start and the end of one period of the LIR, which is one iteration of the loop. If we take a look at the A0 signal whenever the LIR signal goes from low to high (when a new instruction begins), we see that at the first rise of the LIR signal, the A0 signal is low. The address on the address bus at the start of an iteration of the loop should be 0000 according to our cycle-by-cycle table from Part 1. This is consistent with the idea that the address is even when the least significant bit on the address bus is 0. Next, we see that at the second rise of the LIR signal, the A0 signal is high. The address on the address bus at this point in the loop should be 0001 according to the cycle-by-cycle table. This is consistent with the idea that the address is odd when the least significant bit on the address bus is 1. Finally, if we look at the final rise of the LIR signal in this period, the A0 signal is high again. The address on the address bus at this point is 0003. Again, this is consistent with the idea of an odd-valued address having a 1 as its least significant bit.

**Part 5**

Finally, we connected the oscilloscope probes to just the R/W' signal and one of the data bus lines. Again, we chose to look at the least significant bit on the data bus since it was indicative of whether a even or an odd-valued data was being sent on the data bus.

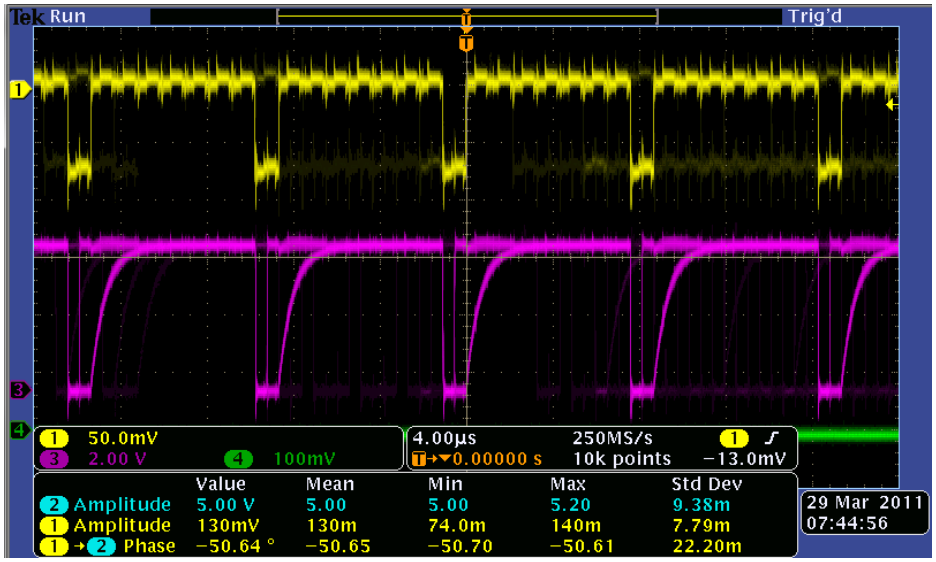


Figure 8. Screen Capture of the R/W' Signal (Yellow) and the D0 Signal (Purple).

From the screen capture in Figure 8, we noticed that in once cycle, the data bit was not stable (keeps showing 1 and 0 seemingly simultaneously). This is occurring because the processor is writing to memory, alternating between an even value and an odd value with each iteration of the loop, so quickly that the signals start to overlap.

If we take a look at one signal by pressing Single on the oscilloscope, we get the following:

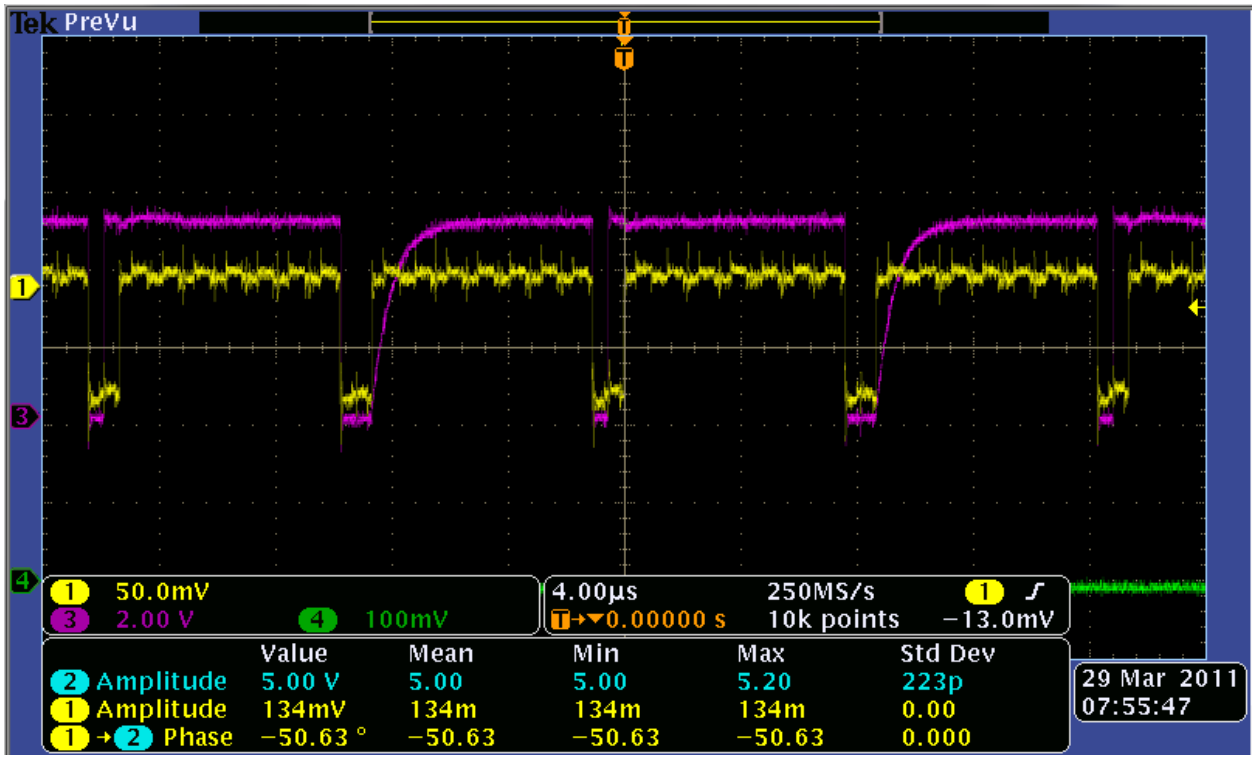


Figure 9. Screen Capture of a Single R/W' Signal (Yellow) and D0 Signal (Purple).

Knowing that the data written is actually seen during the last half of the write signal, we can see that during the first write in the screen capture, an odd-valued data (high D0) was written to memory, during the second write, an even-valued data (low D0) was written to memory, and then this pattern repeated. This is consistent with the fact that with each iteration of the loop, the value written to memory altered between being even and odd. However, the D0 signal seen in Figure 9 did not alter between being low and high while the processor was reading (high R/W'), which was inconsistent with our cycle-by-cycle table in Part 1, which suggested that the data on the data bus altered between being an even value and being an odd value more frequently than what is shown in the screen capture.