

The City College of New York

Grove School of Engineering - Electrical Engineering Dept.

EE 42500 – Computer Engineering Lab

Professor J. Feng

Spring 2020

**Experiment 4: Pulse Train Synchronization**

Report due: 4/5/2020

**Prepared By**: Ben Variano

**Introduction**

This experiment deals with pulse train synchronization. It tasks us with developing assembly code in the MPLAB IDE to output specific pulses to the symbolic oscilloscope. Each task provides us with a different desired output and asks us to change our code to accommodate each output.

**Task 1**

Objective

In Task 1, we were asked to write an assembly code that generates four synchronized outputs to pins RC0 – RC3 on PORTC that would coordinate to channels 1 – 4 of the imaginary oscilloscope. The figure and table for the corresponding simulated oscilloscope output were provided and are displayed below. We then wrote our assembly code that was capable of providing the desired outputs given.

Solution

For Task 1, each of RC0, RC1, RC2, and RC3 are outputting the same pattern but on a 1 loop delay to Channel 1, 2, 3, and 4. They are each outputting high for 0.1 m/s (one time delay) and then low for the remaining 3, which is repeated continuously. After Channel 1 outputs high for 0.1 ms, Channel 2 then does the same, followed by Channel 3, then Channel 4, and finally back to Channel 1 repetitively. It’s important to note the outputs can be shifted left and right in any number of moves, but the order of the digits must remain, as well as their location/offset in relation to each other.

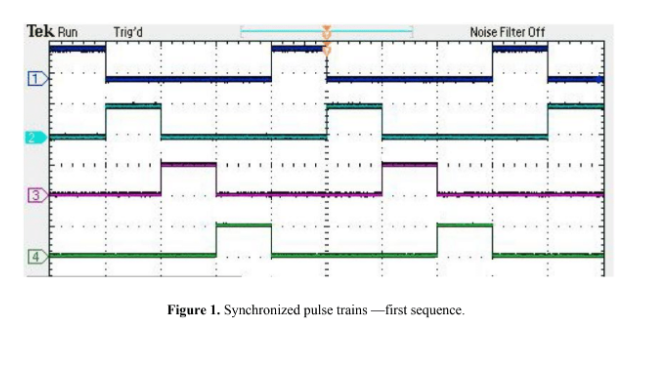
RC0 : 10001000

RC1 : 01000100

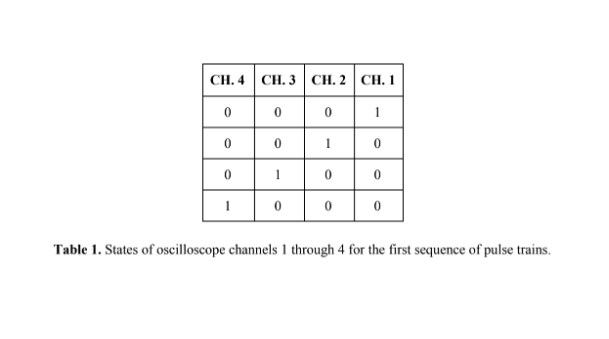
RC2 : 00100010

RC3 : 00010001

Figure



Table



**Task 2**

Objective

For Task 2, we were asked to write assembly code to generate four synchronized outputs to pins RC0 - RC3 on PORTC to correspond to channels 1 – 4 on the imaginary oscilloscope. The same task as part 1, but this time corresponding to a new set of outputs which are shown in the figure and table below. We then wrote our assembly code that was capable of providing the desired outputs given.

Solution

For Task 2 we have a similar scenario to Task 1, but some of the outputs are overlapping. Each of the channels has a high output for two time delays, 0.2 ms. Each of the channels are still only delayed by 0.1ms from the prior so that the latter half of the earlier output overlaps with the starting half of the next output. This is achieved by having each of the outputs RC0 – RC3 output high for 2 time delays and low for 2 time delays, and off putting each successive output by 0.1 ms. Looping from 4 back to 1 at the end ensures the output is the same as long as the code is running. It’s important to note the outputs can be shifted left and right in any number of moves, but the order of the digits must remain, as well as their location/offset in relation to each other.

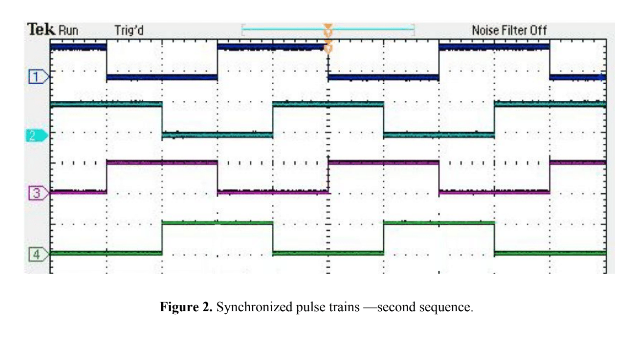
RC0 : 11001100

RC1 : 01100110 By having the outputs to the channel output as shown, with each bit being

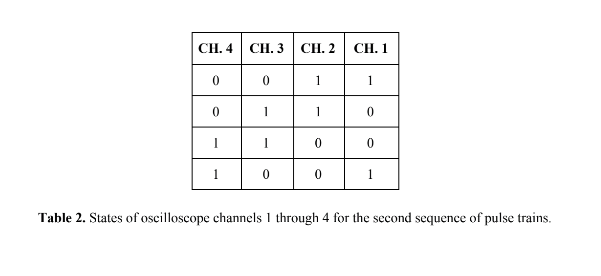
RC2 : 00110011 one time delay (0.1 ms) we achieve the desired output.

RC3 : 10011001

Figure



Table



**Task 3**

Objective

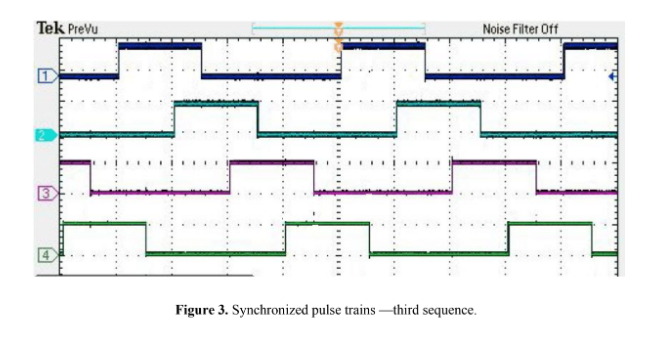
For Task 3, we were asked to write assembly code to generate four synchronized outputs to pins RC0 - RC3 on PORTC to correspond to channels 1 – 4 on the imaginary oscilloscope. The same task as part 1 and 2, but this time corresponding to another new set of outputs which are shown in the figure and table below. We then wrote our assembly code that was capable of providing the desired outputs given.

Solution

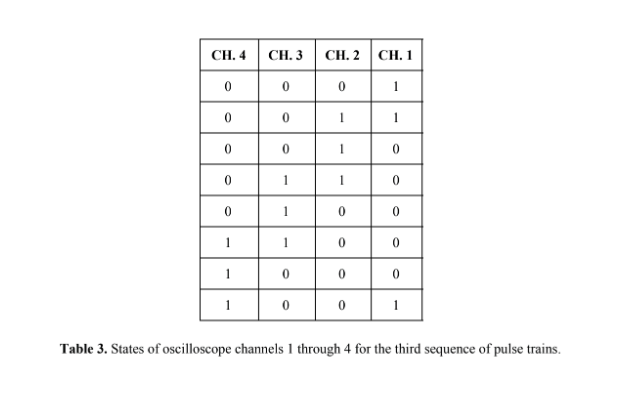
For Task 3, we have a very similar case to Task 2, but each of the outputs is high (1) for 3 time delays and it low for the consecutive 5. In addition, each of the outputs is now delayed by 2 time delays from the former, or by 0.2 ms. For tasks 1 and 2, we discussed in terms of 4 time delay intervals, but they can be viewed similarly to Task 3 by just repeating them and doubling them to 8 as shown in their RC0-RC3 outputs. It’s important to note the outputs can be shifted left and right in any number of moves, but the order of the digits must remain, as well as their location/offset in relation to each other.

RC0: 11000001 RC1: 01110000 RC2: 00011100 RC3 : 00000111

Figure



Table



**Task 4**

Objective

For Task 4, we were asked to write assembly code to generate four synchronized outputs to pins RC0 - RC3 on PORTC to correspond to channels 1 – 4 on the imaginary oscilloscope. The same task as part 1,2, and 3, but this time corresponding to a new set of outputs which are shown in only the table given below. We were asked to graph what we believed the output would look like on the oscilloscope given the data in the table. Next we wrote our assembly code that was capable of providing the desired outputs given by the table and assumed in our graph.

Solution

For Task 4, we notice that each of the channel’s output on different pulses, unlike all the others that are simply left-shifted offsets of each other. RC0, which corresponds to Channel 1 will output low (0) for 8 consecutive cycles and then high (1) for the next 8 cycles, repeating infinitely on this 16 bit pattern. RC1 performs the same but with exactly half the rate of cycles. Outputting low for 4 cycles and high for the next 4, infinitely on a 16 bit pattern as well. RC2 outputs to channel 3 at exactly half the rate of channel 2, outputting low for 2 cycles then high for 2 cycles. And RC3 outputs to channel 4 at half of that right, switching from high to low with every time delay. Note when I refer to time delay/cycle, it is 0.1 ms. It’s also important to note the outputs can be shifted left and right in any number of moves, but the order of the digits must remain, as well as their location/offset in relation to each other.

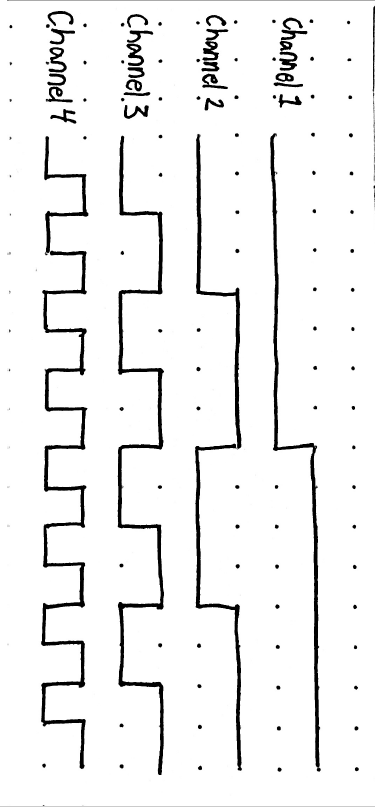
RC0 : 0000000011111111

RC1 : 0000111100001111

RC2 : 0011001100110011

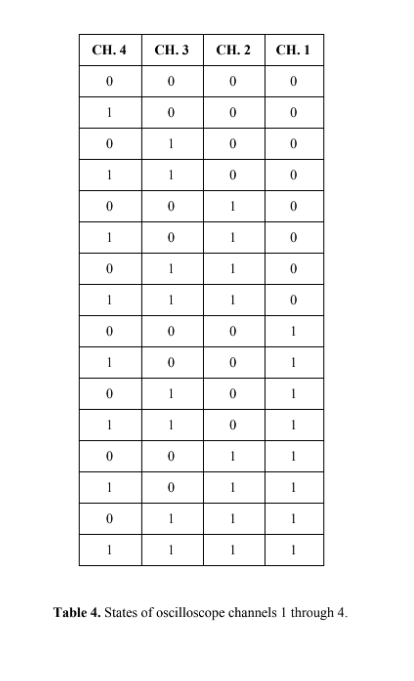
RC3 : 0101010101010101

Figure



**Figure 4.** Synchronized pulse trains – fourth sequence

Table



Conclusion

We were able to establish connections to pins RC0, RC1, RC2, and RC3 on PORTC in our assembly code to output and write to Channels 1, 2, 3, and 4 on our imaginary oscilloscope. We then were able to write to each of these ports a high (1) and a low (0) for the corresponding amount of time that was specified in the tables and figures provided. We did this using a subroutine that we developed (MySubroutine) and using the loop equation/routine (LoopTime) provided to us. We used our subroutine to properly write our outputs to the appropriate pins so that the oscilloscope would observe the outputs that the tables required. In conclusion we were able to manipulate the outputs accordingly and release them in ‘pulses’ to the oscilloscope to control whether the output to each of the channels was high or low.