

Hardware Lab 3: Latches, Flip-Flops and Counters

Prerequisites: Before beginning this laboratory experiment you must be able to:

- Communicate with the Arduino board using the Firmata Test application. Refer to the instructional videos for Week 2 on how to download and install the Arduino IDE, upload the firmware, download and test Firmata.
- Use a prototyping board.
- Interpret the function definition table of a D and J-K flip-flop.
- Describe the binary up-counter counting sequence.

Equipment: Personal computer, Arduino Uno (or equivalent) with Firmata firmware loaded, USB cable, breadboard, breadboarding wire bundle.

Integrated Circuits: You will need the following IC to complete this lab:

- (1) 7400 (Quad 2-Input NAND)
- (1) 7402 (Quad NOR gates)
- (1) 7404 (Hex Inverters)
- (1) 7408 (Quad 2-Input AND gates)
- (1) 7474 (Dual D Flip Flops)
- (1) 74112 (Dual J-K Flip Flops)
- (2) 1000 Ohm Resistors

Objective: The objective of these laboratory assignments is to allow you to gain some experience in building and using latches, flip-flops and registers. You will also learn to apply your knowledge of the operations of these devices by building a 2-bit binary up-counter.

Outcomes: When you have completed the tasks in this experiment you will be able to:

- Realize and describe the operation of an S-R and D latch.
- Describe the difference between active-high and active-low inputs.
- Use a D and J-K flip-flop in a circuit.
- Realize a two-bit binary up-counter using J-K flip-flops.

Introduction

In this laboratory exercise you will build so-called sequential circuit. Up to this laboratory all of our circuits were combinational circuits, which had the characteristic that changing the input conditions directly affected the output of the logic circuit. Sequential circuits are different in the sense that they maintain their output even if one or more inputs change – they store the output. With every task in this lab you will design circuits that exhibit more and more of this “independence” of the output on the input. Eventually, you will work with flip flops that are “non-transparent”, which means that the output is only influenced by the input at a particular point in time. This allows us to use the output signal and feed it back to use it as an input signal. We can design circuits that work completely autonomously, for example a binary counter which keeps on incrementing its value until it reaches the highest value when it resets and starts over again.

Task 3-1: Build an Active-High S-R Latch.

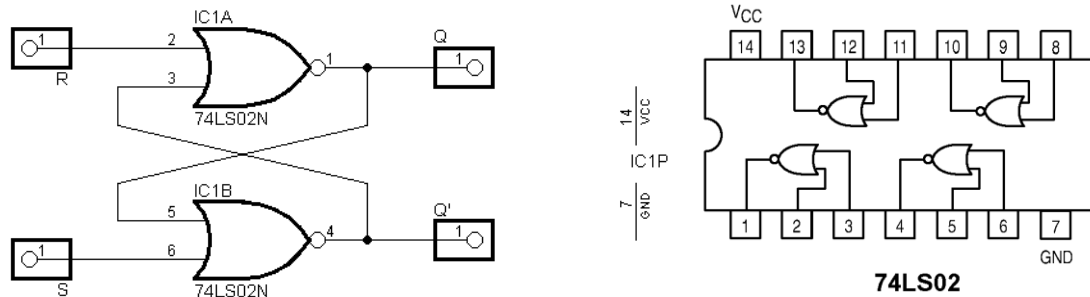


Figure 1: Schematic diagram of the active-high S-R latch using NOR gates. The pinout of the 74LS02 IC is shown on the right. Note the different gate layout from the other ICs. (Pinout diagram copyright of Motorola. Used with permission.)

The first circuit you will build is an active-high S-R latch. The schematic diagram, including suggested pin numbers is shown in Figure 1. Set up the Arduino so that you have two logic outputs available, for example Arduino Pins 2 and 3, and two logic inputs, for example Arduino Pins 7 and 8. The Arduino logic outputs will be used to drive the R and S inputs of the latch and the two Arduino logic inputs will monitor Q and Q'. Do not forget to connect the 5V supply to IC Pin 14 and GND to IC Pin 7. When wiring the circuit, watch out for the "reversed" layout of the 74LS02.

When testing the circuit, you should follow a particular sequence. Start off with both inputs set to logic low. Then switch S to high, leaving R low. Next, switch S back to low. The output should not change once it is set to 1, which is the whole point of this latch. This is the suggested sequence to follow: S,R: 0,0 -> 1,0 -> 0,0 -> 0,1 -> 0,0 -> 1,1 -> 0,0.

Observe what happens to the outputs if you switch both inputs to true. When making the transition from 1,1 to 0,0 do you remember which output you switched first? Switch both inputs back to logic true and then back to false, paying attention to which input you switch first. Repeat this task switching the other input to false. Report your observation on the lab template.

Task 3-2: Build a D Latch

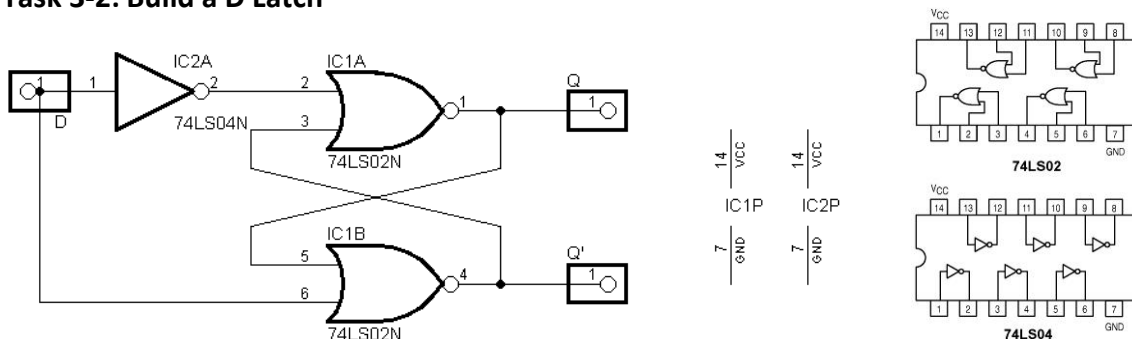


Figure 2: Schematic diagram of a D latch. The pinouts of the 74LS02 and 74LS04 ICs is shown on the right. (Pinout diagrams copyright of Motorola. Used with permission.)

In this task you will convert the S-R latch into a D latch by adding a NOT gate in front of the R input of the R-S latch as shown in Figure 2. So do not rip apart your circuit from Task 3-1, rather add the NOT gate and re-connect the Arduino output Pin 2 to the D input of your latch. The remaining Arduino output can stay disconnected for this task. Test the circuit by changing the value of the signal driving the D input and observe the two output signals. Record your findings in the lab report template. .

Task 3-3: Build a D Latch with Enable

One of the limitations of the D latch is that it has no real storage mode; the input at D will always re-program the latch. This limitation can be eliminated by using the D latch with enable shown Figure 3.

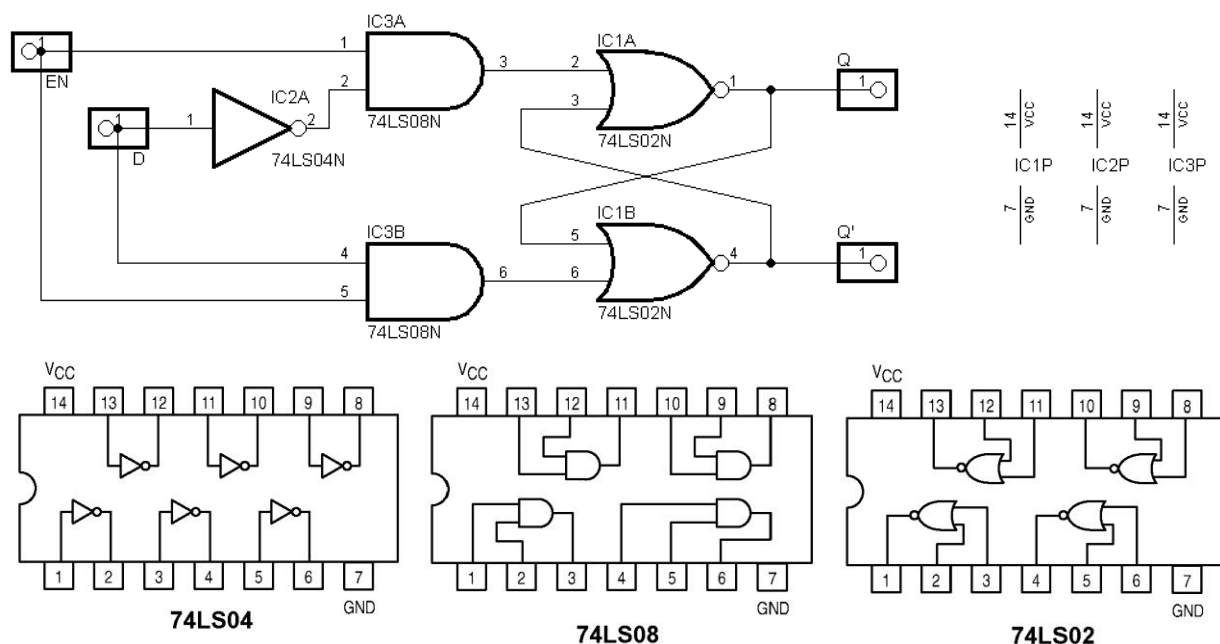


Figure 3: Schematic of a D latch with Enable. The pinouts of the 74LS02, 74LS04 and 74LS08 ICs is shown at the bottom. (Pinout diagrams copyright of Motorola. Used with permission.)

Using the circuit from Task 3-2, remove the connection between the output of the NOT gate and the R input of the latch and insert an AND gate. Additionally, insert an AND gate in front of the S input of the latch as shown in Figure 3. To test the circuit, set Enable to logic high and test the D latch the same way you did in Task 3-2. Record your results in the lab report template. Then, set the output Q to be logic low. Disable the latch by setting Enable to logic low and switch D to logic high. Now enable the latch and observe the output Q. Finally, disable the latch and switch D back to logic low. Does the latch switch its output when the Enable signal is logic low? Does it matter what the output Q is? Report your result in the truth table on the lab report template.

Task 3-4: Build an Active-Low S-R Latch

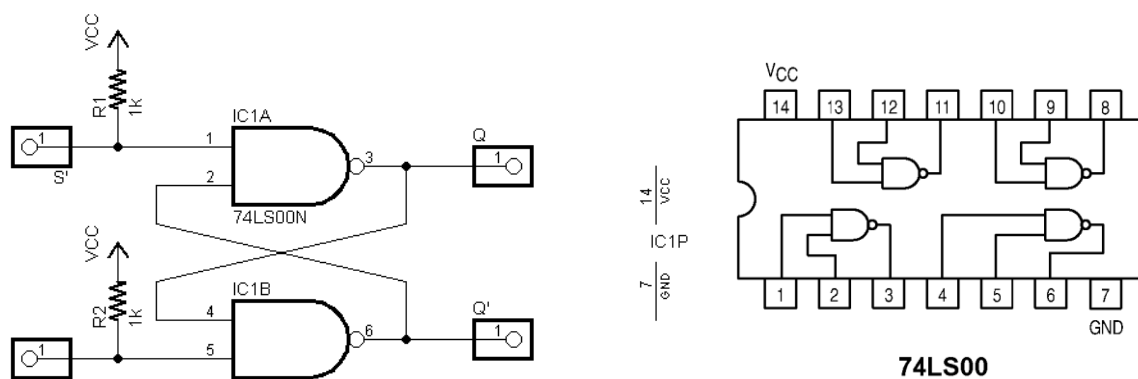


Figure 4: Schematic diagram of an active-low NAND S-R latch. The pinout of the 74LS00 IC is shown on the right. (Pinout diagram copyright of Motorola. Used with permission.)

In this task, you will build an S-R latch similar to the one in Task 3-1. The difference between the latch you built in Task 3-1 and the one in Task 3-4 is that you will be using NAND gates instead of NOR gates. Because of the different layout of the two ICs, it makes most sense to completely rebuild this circuit from scratch to avoid wiring mistakes. Figure 4 shows the schematic of the circuit as well as the pinout diagram of the IC. By using NAND gates

for the S-R latch, we changed the input behavior of the latch to be active-low. This means that the “storage mode” is now active if both inputs are at logic high and that the latch is set or reset by switching one of the inputs to logic low. To ensure that our circuit will be held in “storage mode”, you should connect two 1k Ω resistors from Vcc to the IC inputs for S’ and R’. To demonstrate the “latching” ability of the circuit you should connect the two output signals from the IC to the Arduino inputs Pin 7 and Pin 8. Rather than connecting the input signals S’ and R’ to the Arduino, connect a wire to GND and touch the S’ and the R’ input with the other end of the wire. This re-creates the function of a momentary push button. Observe what the output signal does when you touch one of the inputs and when you remove the GND wire from the input. Does the output logic value change when you remove the GND wire? What happens if you repeatedly connect the GND wire to the same input? Report your findings in the lab report template. To test the case in which both S’ and R’ inputs are logic low, connect two wires, one from S’ to GND and another from R’ to GND and record the logic for both outputs Q and Q’.

The circuit that you just built and tested shows how a latch can be used to re-create the functionality of a two-position (mechanical) switch that switches between logic low and high. Now you can use momentary push buttons or other circuits that output a momentary logic low, like a wire to GND or a circuit with an open collector output. The circuit is also often called a “switch debouncer”, because it will maintain its output no matter how often you push the same input button. Even if that button only makes intermittent contact, it is still enough to set or reset the latch. Your wire to GND is a “worst case scenario” of a worn-out pushbutton.

Task 3-5: Verify the Function Definition Tables for D & J-K Flip-flop

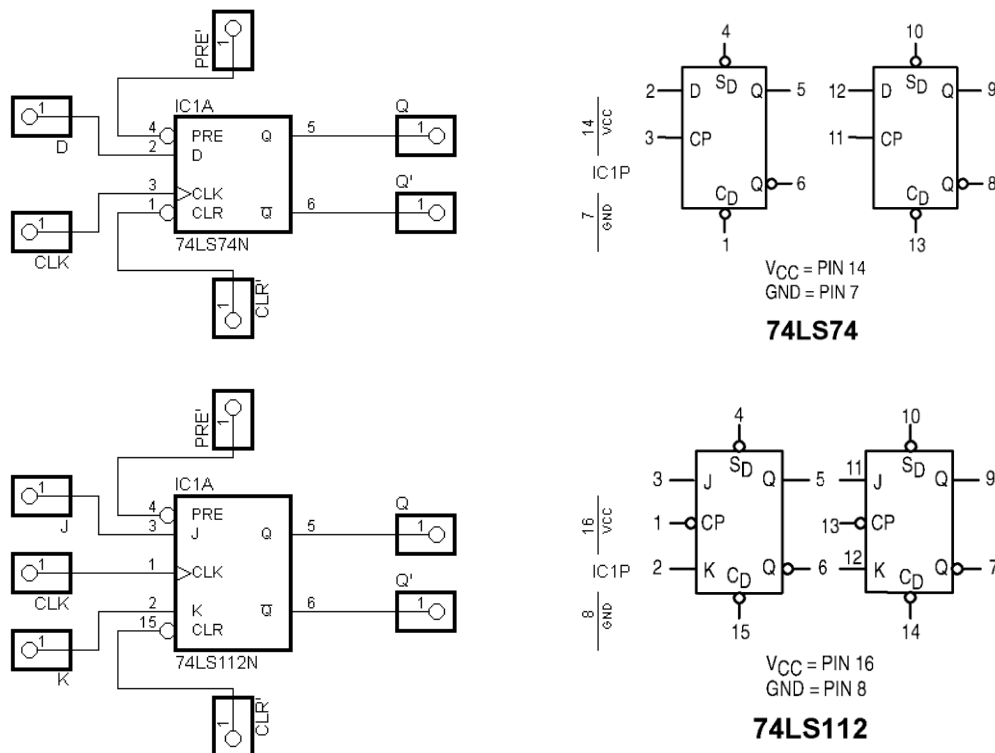


Figure 5: Testing setup for the D flip flop and the J-K flip flop. The pinouts of the 74LS74 and 74LS112 ICs are shown on the right. (Pinout diagrams copyright of Motorola. Used with permission.)

Moving from the level-triggered latches, like the one you built in Task 3-3, to edge-triggered flip flops, you are supposed to verify the function definition tables given in Tables 1 and 2, using the TTL 74LS74 (D) and 74LS112 (J-K) flip-flops. For this task, disconnect everything you previously built and start by connecting the D flip flop on your breadboard according to the wiring diagram shown in Figure 5. You have to use four Arduino output pins (for example Pins 2-5) and two Arduino input pins (for example Pins 7 and 8). You have the freedom of choice which

pin performs which function, but to keep track of the pins, the table in the lab report template has a line where you can write down the pin number used. Don't forget to connect GND and +5V as well. You should be able to read and make sense of the function definition table and develop your own testing protocol, asking yourself how would you verify the operation if signals have been labeled as don't cares.

Note the difference between the edge trigger behavior between the D flip flop 74LS 74 and the J-K flip flop 74LS112. While the D flip flop is positive-edge triggered, the inverter bubble before the triangle in case of the J-K flip flop indicates that this one is negative-edge triggered (see Figure 5). This means that the device will change state only when a high to low (1 to 0) transition of the clock occurs.

Also note that the 74LS112 is a 16-Pin device. GND and +5V are still at the corner pins, but they are shifted, so it makes most sense to build the second testing circuit from scratch.







Table 1				
D flip-flop function table				
PRE'	CLR'	Clk	D	Q ⁺
1	1		0	0
1	1		1	1
0	1	X	X	1
1	0	X	X	0

Table 2					
J-K flip flop function table					
PRE'	CLR'	Clk	J	K	Q ⁺
1	1		0	0	Last Q
1	1		0	1	0
1	1		1	0	1
1	1		1	1	Last Q'
0	1	X	X	X	1
1	0	X	X	X	0

Task 3-6: Build and Test a Controlled Two-Bit Binary Up-Counter

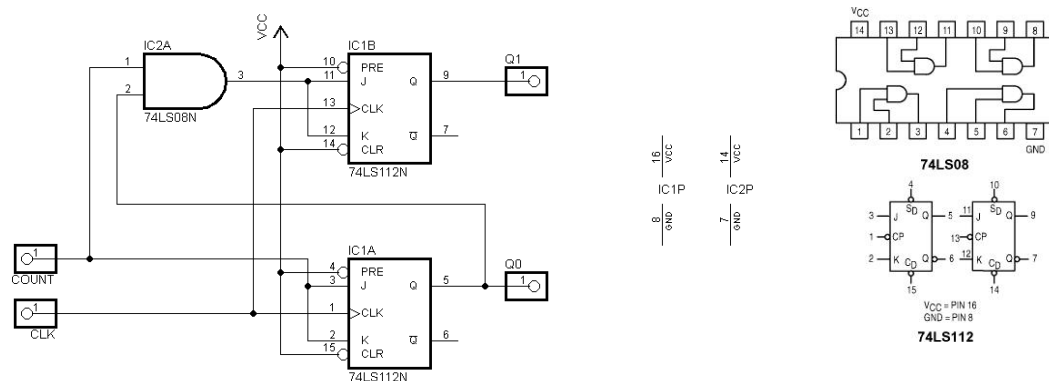


Figure 6: Schematic diagram for the controlled two-bit up counter using J-K flip flops wired as T flip flops. . The pinouts of the 74LS08 and 74LS112 ICs are shown on the right. (Pinout diagrams copyright of Motorola. Used with permission.)

For the final task of this lab exercise you are supposed to build a 2-bit binary up counter. You can leave the 74LS112 chip on your breadboard, because it already has two J-K flip flops built in, you just have to re-wire the input and output connections. Both J-K flip flops have been wired to act as T flip flops as shown in Figure 6 by connecting the J and K inputs for each flip flop together. In addition to the 2-bit counter shown on the lecture slides (see Figure 7), this counter features an active high count enable input. If the input is logic low, both flip flops are supposed to store their last value and the clock will not advance the count. When the control input is logic high, the counter should behave like the counter shown on the lecture slides. This can be accomplished by connecting the flip flop input for the lowest bit to the count enable input directly and connecting the output of that lowest bit flip flop via an AND gate to the input of the higher bit flip flop. The AND gate will ensure that the flip flop will be in

storage mode while the count enable input is logic low and will pass the signal through when the count enable input is logic high.

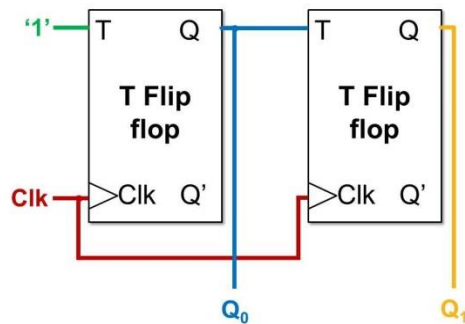


Figure 7: Schematic diagram of the permanently enabled 2-bit binary up counter as shown on the lecture slides. The flip flop for the lowest bit is permanently toggling while the higher bit only toggles whenever Q_0 is 1.

Use two of the Arduino output pins, for example Pins 2 and 3 to control the count enable and the clock. Use two of the Arduino input pins, for example Pins 7 and 8 to monitor the outputs Q_0 and Q_1 . You do not need to monitor the complemented outputs. Be careful to connect all asynchronous flip-flop PRE' and CLR' inputs to +5 V.

After you connect GND and +5V to your circuit, what is the initial count value at the outputs? Record this in your lab report template. Test your circuit by controlling the Arduino pins with one pin acting as a manual clock signal and record the count values in the truth table on the lab report template.

After finishing the tests to complete the truth table, disconnect the +5V from the breadboard and reconnect it again. What is your count output value now? Repeat this several times and record the values on the lab report template. Does the counter always have the same value when the power is turned on? What circuitry would you add to bring your circuit to a known count value? You do not have to build this, but you should mention it on your lab report template.

Task 3-7: Take a Photo of your Completed Circuit

Before you rip everything apart, take a photo of your completed circuit and attach it to the lab template. This will serve as a replacement for the "attendance stamp" that the in-person students have to get on their paper lab submission.