Engineering Sciences 50

Flip-Flops and Some Applications

Laboratory 3

Purpose: These exercises are meant to acquaint you with the characteristivs and typical applications of flip-flops.

Background: The exercises are divided into two main sections each with several parts. Section one introduces the basic functional characteristics of latches and flip-flops. Section two studies ripple counters and the design of a stop watch.

Please note that at the end of the exercises there are three supplementary design questions to be addressed in your lab report in addition to the questions posed throughout the exercises. Your answer should be in the form of paper designs including schematics (Of course you are welcome to actually build, debug and demonstrate working versions, if you'd like).

Engineering Sciences 50

Experiment 3: Introduction to bistable elements.

SECTION ONE

Purpose:

1. To explore the methods of storing a binary state using feedback.
2. To examine the various ways of using NAND gates to store states.
3. To understand the differences between various kinds of flip-flops
4. To understand the role of a clock in a circuit.

Parts:

2	74HCT00 NAND gate.
1	74HCT10 Triple 3 input NAND gate.
1	74HCT74 Dual D type flip-flop.
1	74HCT76 Dual J-K flip-flop.

Introduction

In all the circuits we have looked at so far, we have not been able to preserve the state of a signal after the inputs that produced it have changed. This would be useful, since we could catch the value of a signal that is changing rapidly and remember it for later use. In this lab we will introduce the basic element of memory for digital circuits -- the flip-flop. Flip-flops are circuits that have 2 stable states (output a 0 or a 1) and the ability to change between them in response to control signals. Once in a given state, they stay there until forced to change into the opposite state.

Even the most modest digital computer has hundreds of flip-flops used for information storage. In subsequent experiments, we shall see many applications of bistable circuits, but for now, we will concentrate on their basic properties.

Flip Flops

Part 1: The RS flip-flop

The simplest bistable circuit is the RS flip-flop. Using a couple of NAND gates (74HCT00), connect the circuit, as shown in **figure 3-1**.

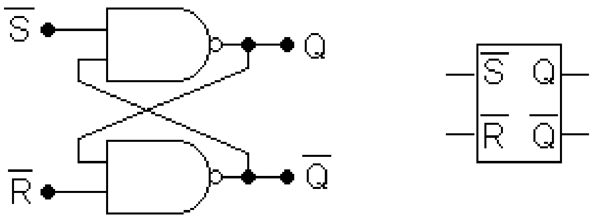


Figure 3-1

Connect the $\overline{S}$ (set) and $\overline{R}$ (reset) inputs, noting that these are active-low inputs, to the normally high terminals of the pulser switches. Connect the complementary outputs Q and $\overline{Q}$ to monitor lamps. Try various combinations of set and reset until you convince yourself of the following truth table:

The RS flip-flop

$\overline{R}$	$\overline{S}$	Q	$\overline{Q}$
0	0	Not Defined	
0	1	0	1
1	0	1	0
1	1	No Change	

The Q output can be made high or low by applying a momentary low pulse to either the $\overline{S}$ or $\overline{R}$ input respectively. The pulse must go from a 1 to 0 and back to 1 $(1 \rightarrow 0 \rightarrow 1)$. Why doesn't the opposite sequence, $0 \rightarrow 1 \rightarrow 0$ work?

Note that once the RS flip-flop is in a given state, it will stay in that state until forced into the opposite state.

The RS flip-flop could also be constructed with NOR gates, but there is a slight difference in the operation of the circuit. Without building the circuit, can you figure out what the difference is?

We shall now modify the RS flip-flop to provide a clocked input, so do not remove the circuit yet.

Part 2: The Clocked (synchronous) RS flip-flop

It is sometimes necessary to take a snapshot of a signal at a given instant or to accept the value of a digital input only at a certain time. For example, the data transfer in a computer is controlled by a series of clock pulses. The inputs to the flip-flop are only valid when the clock is high. The RS flip-flop needs two more NAND gates to allow us to chose the time at which its inputs become active. The resulting circuit is called a clocked RS flip-flop (**figure 3-2**).

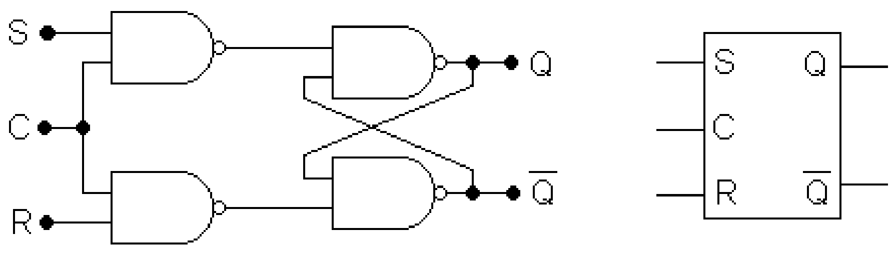


Figure 3-2

Using the 74HCT00 chip from part 1, modify the RS flip-flop to accept a Clock (C) signal. Notice that since NAND gates are used, the S and R control inputs are now active high. Connect inputs R and S to logic switches, and the C input to the normally low terminal of a pulser. Enter single clock pulses by depressing and releasing the pulser button. Confirm the following truth table. Note that in this table R_n and S_n represent the state of the inputs before the nth clock pulse, and Q_{n+1} and $\overline{Q_{n+1}}$ represent the outputs after the clock pulse.

The Clocked RS Flip-flop.

R_n	S_n	Q_{n+1}	$\overline{Q_{n+1}}$
0	0	Q_n	$\overline{Q_n}$
0	1	1	0
1	0	0	1
1	1	Undefined	

Although this is a very useful circuit, it has some deficiencies, namely, the undefined state. When the circuit is first turned on, its outputs are, as a rule, undefined. It would be extremely valuable to have some way of initializing all the flip-flops in a circuit. The next step is to modify our clocked RS flip-flop to allow asynchronous (not dependent on the clock) $\overline{\text{set}}$ (also called preset -- that's pre-set and not p-reset) and $\overline{\text{reset}}$ (also called [pre]clear) inputs. When $\overline{\text{set}}$ is brought low, the Q output will immediately be set to 1, without waiting for a clock pulse. Similarly, when $\overline{\text{reset}}$ is brought low, Q will immediately be set to 0. Modify your RS flip-flop according to **figure 3-3** and derive the truth table.

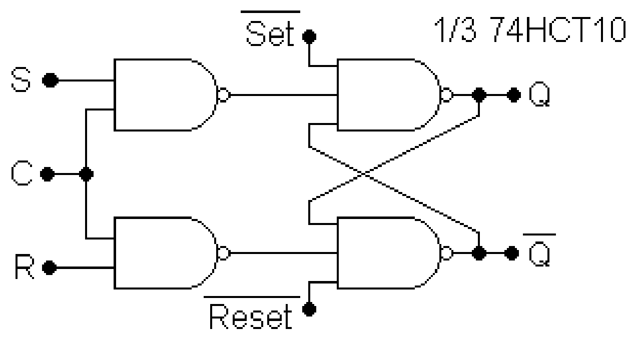


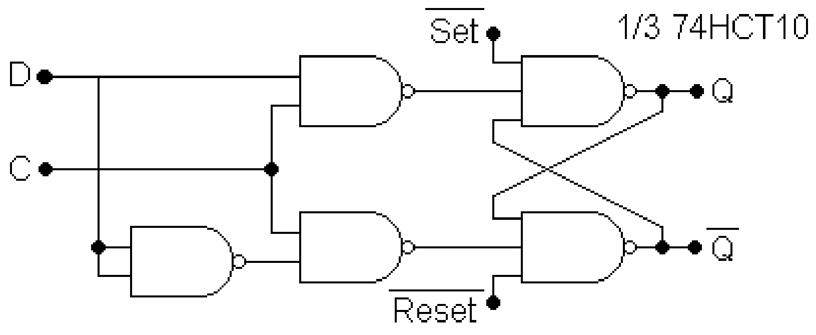
Figure 3-3

This modification solves the initialization problem, but the undefined state problem remains. Later in this lab we, will look at ways of avoiding this problem completely.

In the flip-flops studied so far, the inputs must be stable (unchanging) during the time that the clock pulse is high for the outputs to be stable. This would require an awkwardly short clock pulse if rapidly changing signals are used. Fortunately, we will be further modifying our circuit to correct this problem.

Part 3: The D flip-flop

A very common version of the RS flip-flop is the D flip-flop, also called the data latch. The undefined state is removed by forcing R and S to always be complementary. This improvement is made at the expense of the "no change" state. Despite this, the D FF is still extremely useful. Notice that the NAND gate in **figure 3-4** with both inputs tied together acts like an inverter, without having to use another chip.



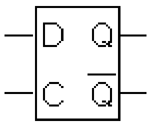


Figure 3-4

Modify your circuit to create a D flip-flop, as shown. Write a truth table for Q_{n+1} vs. D_n. At what point in the clock pulse is the data input latched at the Q output?

On a separate portion of your lab board, take a look at the 74HCT74, a dual edge-triggered D flip-flop. Wire it up using the information and pinout in your data manual. Derive the truth table of one section. Notice at what point in the clock pulse the data is latched. This is called edge triggering. Why is it advantageous?

Part 4: The "JK" Flip-flop

The "JK" flip-flop is perhaps the most powerful of the group since the previously undefined state is now transformed into something useful, a toggling state, as is shown below:

The JK flip-flop

J_n	K_n	Q_{n+1}
0	0	Q_n (no change)
0	1	0
1	0	1
1	1	$\overline{Q_n}$ (toggled)

The "toggle" state inverts the output of the flip-flop through the clock pulse transition.

There are many versions of the JK flip-flop, with positive edge-triggering, negative edge-triggering, asynchronous clear, etc. Look them up in your data book to become familiar with what options the various versions of the JK flip-flop come with. The NAND gate representation is shown in **figure 3-5**:

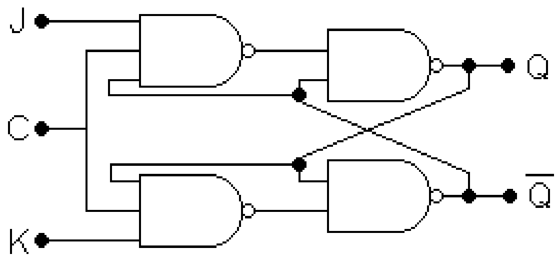


Figure 3-5

Note that this simple cross-coupling to implement toggling has a problem with “burst oscillation.

The most common type is the "master - slave" JK FF, which is a combination of two flip-flops. It is shown in **figure 3-6**:

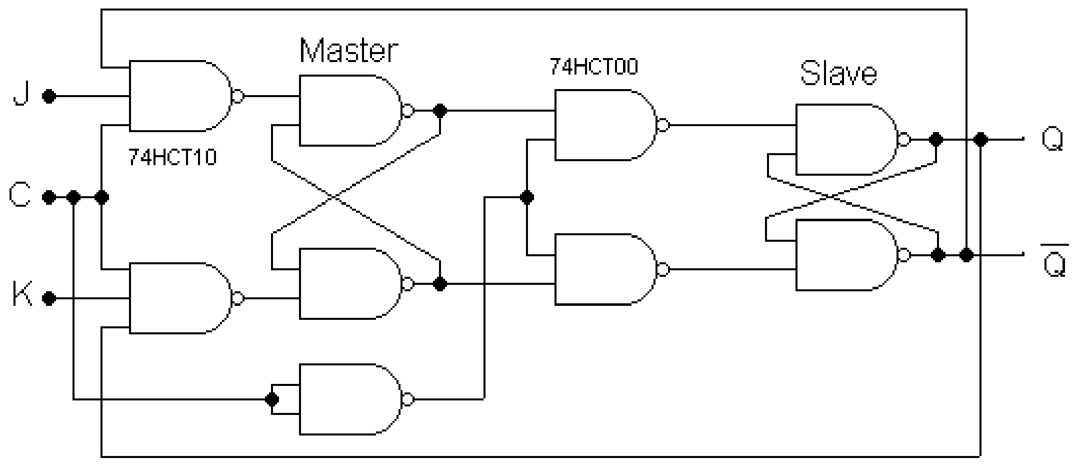


Figure 3-6

At the leading edge of a clock pulse $(0 \to 1)$ the data present at the JK inputs is transferred into the master FF and held while the clock stays high. During this time, the master and slave flip-flops are isolated from each other. On the falling edge of the clock pulse $(1 \to 0)$ the JK inputs are isolated from the outside world, and the data from the master is transferred to the slave.

Wire up the master-slave JK flip flop by modifying the D flip-flop from part 3, and verify that it obeys the truth table. Observe the sequence of data flow from the master to the slave.

The 7476 is a dual master-slave JK FF. Wire up a single section of this chip, and verify that it has the same truth table as our circuit.

Engineering Sciences 50

Experiment 3: Ripple Counters and an Electronic Stopwatch

SECTION TWO

Purpose:

1. Build and study a binary ripple up counter
2. Build and study a binary ripple down counter
3. Build and study a decimal ripple up counter
4. Design and build an electronic stopwatch

Parts:

1	74HCT00	Quad 2 input NAND gate
1	74HCT02	Quad 2 input NOR gate
3	74HCT47	Decoder/drivers
3	74LS90	Decade Counters
2	74HCT109	Dual JK flip flop
3	Seven segment display units	
3	330Ω Resistor Networks (DIP) or	
27	330Ω Resistors	

Introduction

In this section of exercises we will explore one of the major ways in which flip-flops are made use of in digital electronics: counter circuits. As we shall see in later work, nearly every significant real world application of digital techniques involves some kind of counting operation. To make sure we understand how counters work, we will first build some counter configurations from discrete flip-flop primitives. Then, to get some idea how counters are used, we will implement an electronic stopwatch configuration using MSI (medium scale integration) decimal counters as building blocks, acting as a layer of abstraction like the EXOR did.

Part 1: Ripple (asynchronous) counters

In this section we will build both a binary (with a hexadecimal display) and a decimal counter configuration from JK-FF primitives. As you have seen in lecture, the binary configuration is obtained merely by clocking a given JK-FF in the toggle mode with the output from the proceeding JK-FF. The decimal counter involves the additional logic required to cut the binary count off at BCD "9". For somewhat arbitrary reasons, you will use 74HCT109s (dual JK positive-edge-triggered flip-flops). However, negative-edge-triggered JK flip-flops would be suitable too, but as has been discussed, a slightly different circuit configuration would be required from what is presented below.

74HCT109 Function table.

$\overline{S}$	$\overline{R}$	CLK	J_n	$\overline{K_n}$	Q_n	$\overline{Q_n}$
0	1	X	X	X	1	0
1	0	X	X	X	0	1
1	1	↑	0	0	0	1
1	1	↑	0	1	Q_{n-1}	$\overline{Q_{n-1}}$
1	1	↑	1	0	$\overline{Q_{n-1}}$	Q_{n-1}
1	1	↑	1	1	1	0
1	1	0	X	X	$\overline{Q_{n-1}}$	Q_{n-1}

Note, in particular, that this FF changes state on the rising edge (0 to 1) of the clock pulse! Note also that the is $\overline{K}$ active low. The FF has two memory states and one toggle state. Can you identify them from the truth table?

Task 1:

Wire up two 74HCT109's as a four stage binary ripple up counter as shown in **figure 3(2)-1**. Connect the RESET line to one of the logic switches. Connect the outputs (Q_D, Q_C, Q_B, Q_A) to the monitor lamps. Using one of the pulsers as an input, step the circuit through its count cycle to make sure everything is working OK. Then set the clock output to 1Hz, connect it to your counter, and watch it go! Be sure you realize you are counting out a single hexadecimal digit.

Task 2:

Modify your circuit to become a four stage binary ripple down counter and test it. There are several ways to accomplish this. The easy way uses the negated outputs, $\overline{Q_D}, \overline{Q_C}, \overline{Q_B}, \overline{Q_A}$. After accomplishing this, modify the circuit to create a four stage binary ripple down counter without using the negated outputs but instead using the regular outputs Q_D, Q_C, Q_B, Q_A. This is more difficult.

Task 3:

Modify your circuit to obtain a single stage decade (decimal) up counter as shown in **figure 5-2**. Again follow the testing procedure discussed under task 1. Notice the count is no longer a hexadecimal digit, but is what ?

Part 2: An Electronic Stopwatch

In this section you will make use of the 74LS90 decade counters to build a simple electronic stopwatch (**figure 3(2)-3**). This circuit pulls together a number of things we have been talking about and should be fun. We use as the "primary time standard" the clock output. An RS-FF can be used to gate the clock output in or out as determined by the start/ stop pulses. To make the experiment interesting you should display three decimal digits. The board monitor lamps can be used to display one decimal digit. If you have two seven-segment displays, 74HCT47 decoder / drivers and appropriate resistors you can use them for the most significant digits. If you have only one seven-segment display, rig up a four

LED display like we used in Experiment 2. We have included a resistor network for you to use (see **figure 3(2)-4**)

The key element in the circuit is, of course, the 74LS90 decade counter. The 7490 is in fact a divide-by-two and a divide-by-five counter in a single package. Thus, for decade (divide-by-ten) operation, Q_0 (pin 12) must be connected to $\overline{CP_1}$. The reset/ count options are set according to the following table:

Mode Selection Function Table

Set/ Reset Inputs				Outputs			
MR_1	MR_2	MR_3	MR_4	Q_D	Q_C	Q_B	Q_A
H	H	L	X	L	L	L	L
H	H	X	L	L	L	L	L
X	X	H	H	H	L	L	H
L	X	L	X	counting			
X	L	X	L	counting			
L	X	X	L	counting			
H	L	L	X	counting			

<u>Note: A very peculiar and unusual thing: pin 5 is +5 volts and pin 10 is ground!</u>

<u>Task:</u>

Build your stopwatch, being careful to keep the wiring neat. Then test out the operation of the stopwatch at various settings of the clock output. In particular, for slow rates you may calibrate your stopwatch against an ordinary clock. Here is a question to ponder. Looking at the BCD count sequence in your data book, you will notice that the output of Q3 changes state on count 8. Q3 is also the input to the next counter (**figure 3(2)-3**). How then do these counters count to 10?

Figure 3(2)-1 (a): Binary Ripple Up Counter - Timing Diagram

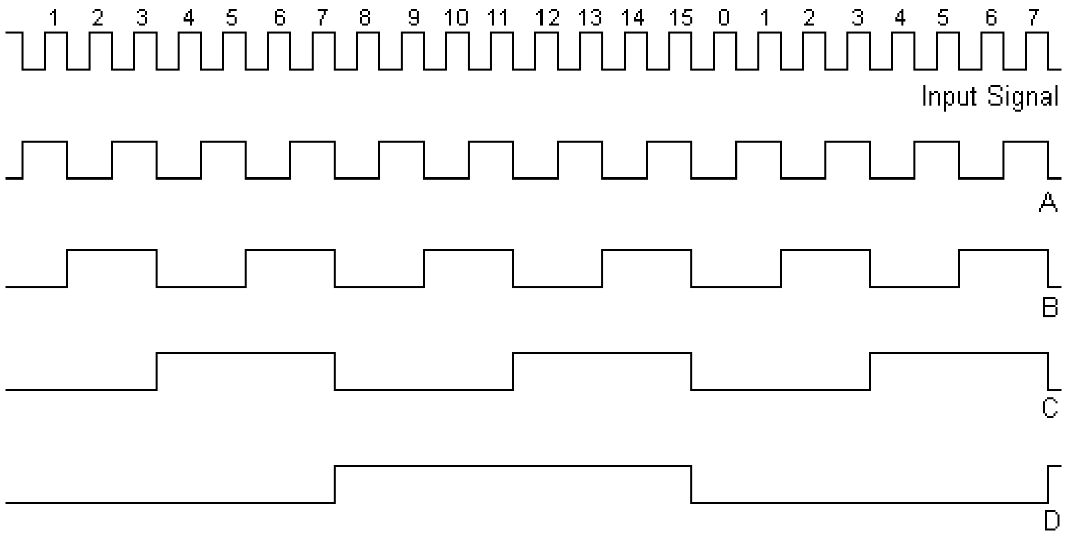


Figure 3(2)-1 (b): Binary Ripple Up Counter - Schematic Diagram

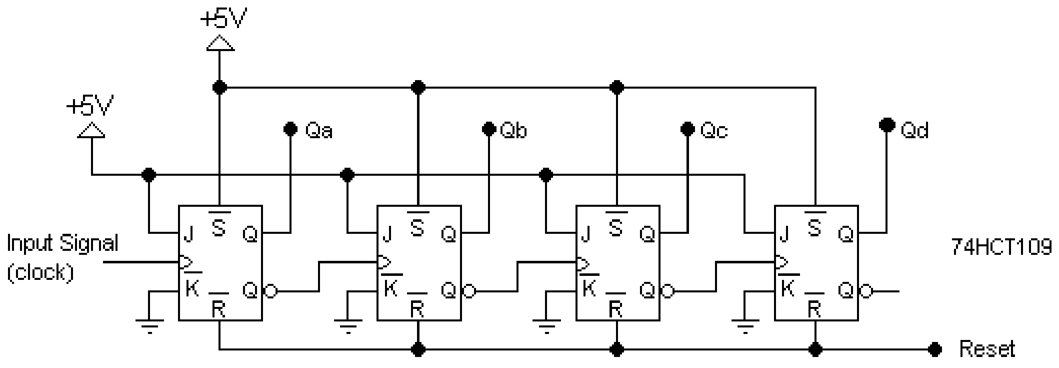


Figure 3(2)-1

Figure 3(2)-2 (a): Decade Ripple Up Counter - Timing Diagram

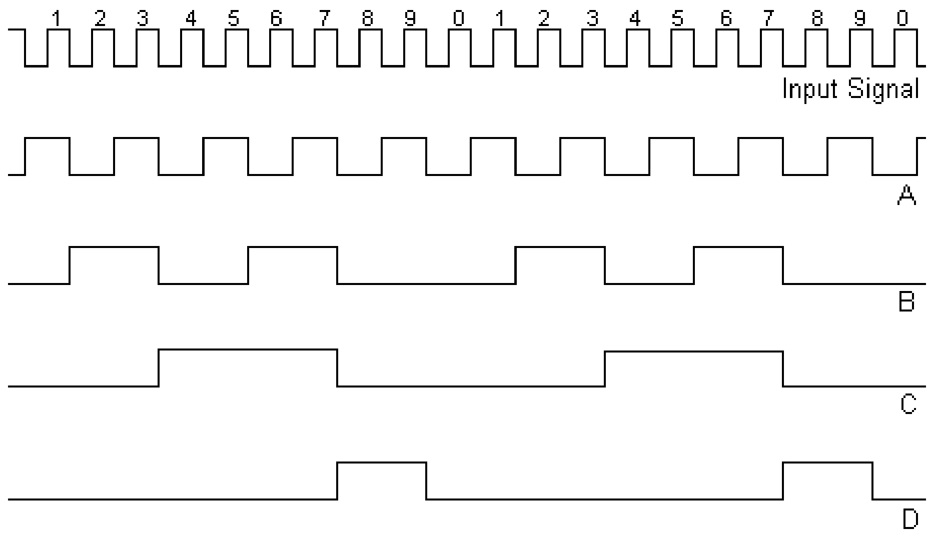


Figure 3(2)-2 (b): Decade Ripple Up Counter - Schematic Diagram

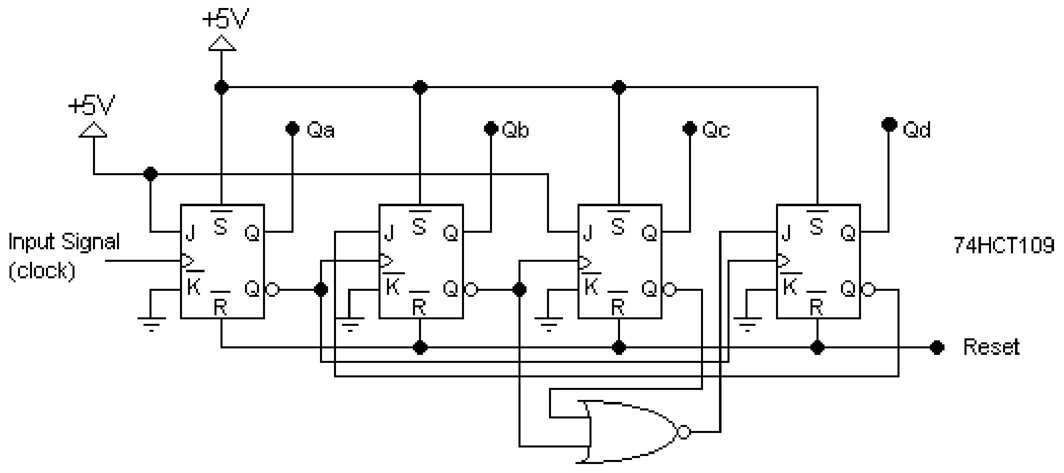


Figure 3(2)-2

Schematic Diagram of the Electronic Stopwatch

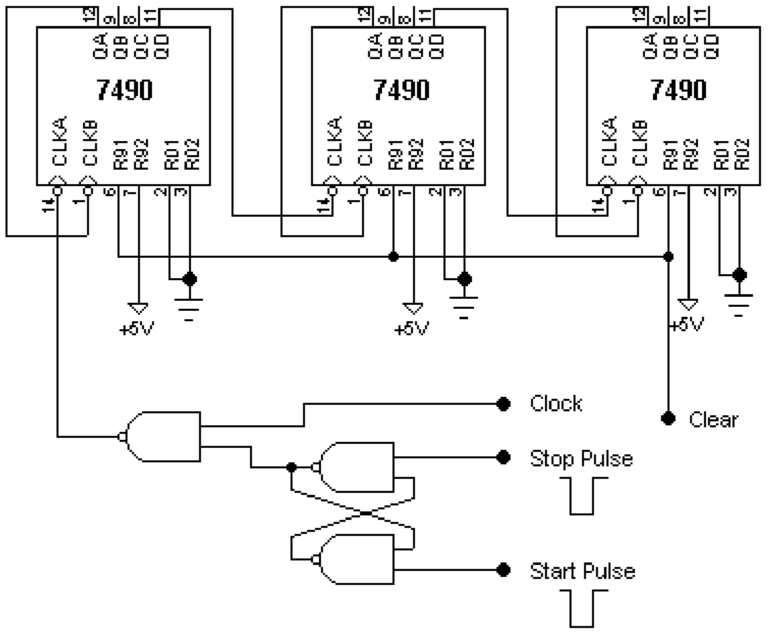


Figure 3(2)-3

330Ω Resistor Network pinout

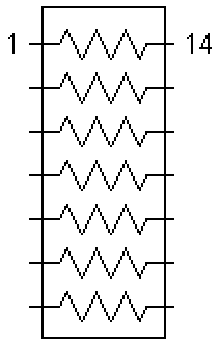


Figure 3(2)-4

Note that this resistor network is not of the same type as the others you have seen. It has 7 independent resistors and no common lead.

Supplementary Questions

1. Using appropriate logic elements, incorporate both the up and down binary ripple counters you constructed for Section one A, Tasks 1 & 2 into a single unit. Your design should include an external logic signal which causes up/down counting for a high/low signal.

2. **Figure 3(2)-1** can be used to construct a fixed modulus counter up to 16, If the CLEAR line is driven by a four input NAND gate, complete the following table of NAND inputs for counter moduli 2 through 16. A modulo-n counter can assume n unique states.

Modulus	NAND Inputs	Counter Range	Freq. Division
2	none	$0 \rightarrow 1$	$f_A = f_{in}/ 2$
3			
4			
5			
6			
7			
8	none	$0 \rightarrow 7$	$f_C = f_{in}/ 8$
9	Q_A, Q_D	$0 \rightarrow 8$	$f_D = f_{in}/ 9$
10			
11			
12			
13			
14			
15			
16	none	$0 \rightarrow 15$	$f_C = f_{in}/ 16$

3. Design a digital clock, displaying HH:MM:SS, around modulo-n counters ($n \leq 16$). Assume the clock is driven by a 120Hz source. Counter intervals are not necessary; building blocks such as ÷12 are sufficient. However, be explicit in the design of the hours digits, which should

assume the '00' state after the '12' state. Start your design by using the 7490 and 74109 chips.