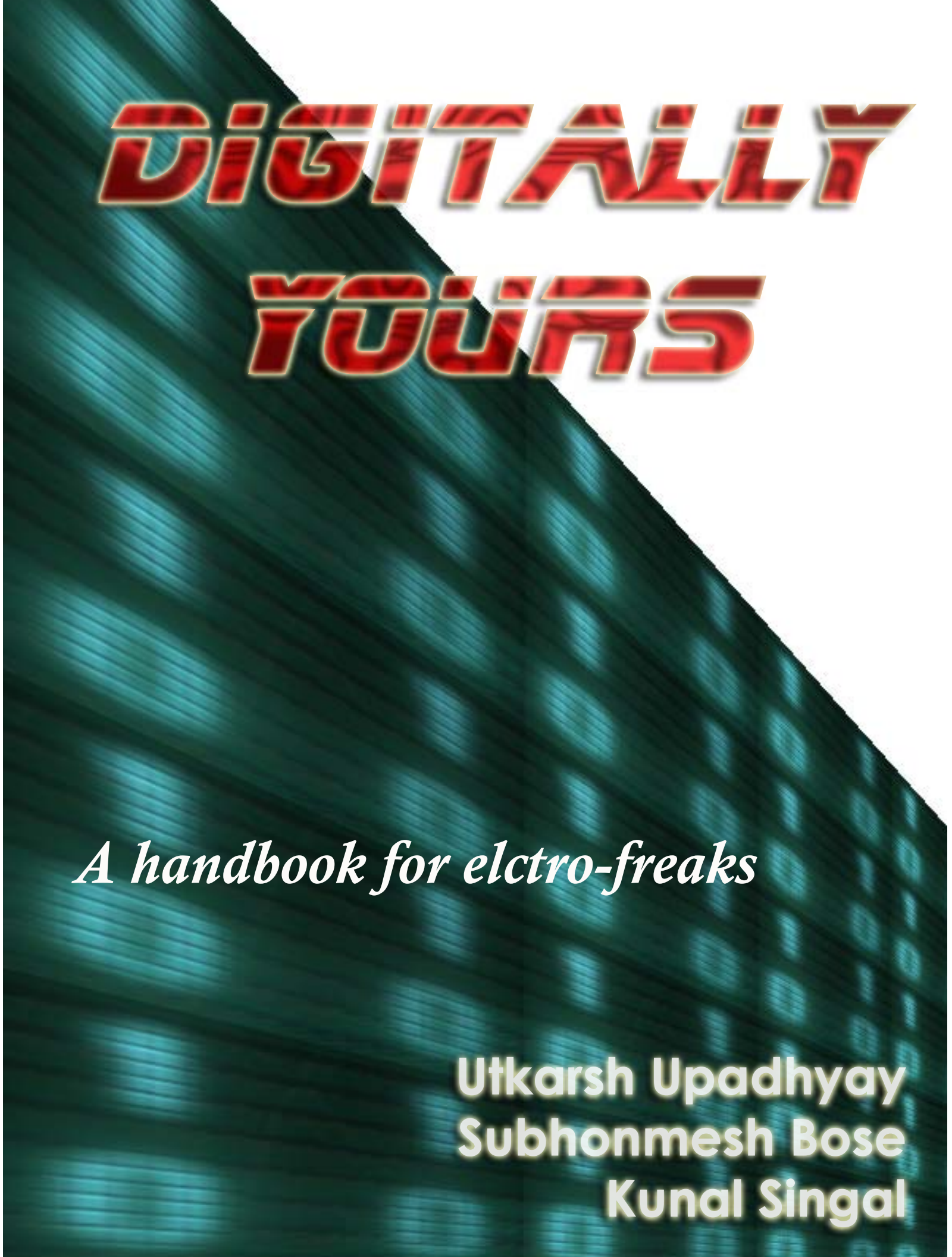

DIGITALLY
YOURS
A handbook for elctro-freaks
Utkarsh Upadhyay
Subhonmesh Bose
Kunal Singal

Table of Contents

We give an approach to digital circuitry in this article.

Why digital?

Introduction

Digital Electronics uses just two states of each input/output. The logic is binary in nature. The standard results of Boolean algebra can thus be easily implemented using just the two states. With chips already available to do the basic operations, the complicated and unreliable nature of analog electronics can be bypassed. Moreover, the world's going digital. Thus we can build circuits using simple sequential logic using the building blocks that incorporate the complications inside them. But we can safely bypass their internal workings and use them as black-boxes.

Basic binary arithmetic and Boolean Algebra.

We develop a simple algebra with each variable having only two states, namely 0 and 1. The interpretation of this 0 and 1 is completely on us. Let's take a decision for instance:

*If **(mom's back home)**,*
I will study,
else
I will watch TV.
Let x be a Boolean variable. x=is mom home?

Now we can associate 0 as the state (mom's not home) and 1 as the state (mom's at home). We would build a framework and show how we can grow out of such a framework and handle complicated problems based on this simple digital framework.

How we denote them in circuits?

A voltage of 5V is usually regarded as 1 and 0V as 0 in most digital circuits. Thus the voltage or potential level has the *logical* equivalent of 0/1. Now what about voltages between 0 and 5V? Small deviations about this value do not matter at all. Thus margin of error with the voltages is much more than in analog circuits. But the voltages in between, like 2V or 3V may be regarded as 1 or 0, depending on the chip in operation.

Use of gates

In this algebra we have some operations defined which work on the variables given. The basic operations are AND, OR and NOT. We denote AND by (.) and OR by (+).

- P1: X = 0 or X = 1
- P2: 0 . 0 = 0
- P3: 1 + 1 = 1
- P4: 0 + 0 = 0
- P5: 1 . 1 = 1
- P6: 1 . 0 = 0 . 1 = 0
- P7: 1 + 0 = 0 + 1 = 1

The knowledge of the gates is essential in understanding digital circuitry. The approach we take in this article assumes the knowledge of the gates on Boolean variables.
***Link*:** Refer to the following site for a revision:
http://educ.queensu.ca/~compsci/units/BoolLogic/summary.html

Chips mainly used

AND – Chip no. 4081
OR – Chip no. 4071
NOT – Chip 4069

4025……Triple 3-Input NOR Gate
4075……Triple 3-Input OR Gate
4082……Dual 4-Input AND Gate
4070 7486…...……Quad 2-Input Exclusive OR Gate
7266……Quad 2-Input Exclusive NOR Gate
4001 7402……...Quad 2-Input NOR Gate
4011 7400……...Quad 2-Input NAND Gate
4071 7432……...Quad 2-Input OR Gate
4073 7411……....Triple 3-Input AND Gate
4081 7408……...Quad 2-Input AND Gate
4049 4069 7404……..Hex Inverter (NOT Gate)
Triple, Quad, Hex means that one chip contains 3, 4, 6 such gates respectively.

Note: It is always desirable to minimize the number of chips required for a particular Boolean function required. Use a K-map to deduce the minimum function required to deduce the functions.

***Link*:** Refer to http://www.ee.surrey.ac.uk/Projects/Labview/minimisation/karnaugh.html for a detailed discussion on K-maps.

Exercise: Suppose you have 3 bits to represent numbers from 1 to 6 on a display panel of LED's such that number of LED's lit up equals the no. represented by the 3 bits. What is the minimum number of gates required to realize this?
Ans: Zero.
Hint: Try thinking of the LED's being arranged in the position of dots on the face of a dice. You can suitably connect the 3 bits so that the output is as required.

Multiplexer

Introduction

A **multiplexer** or **mux** is a device that selects one of many data-sources and outputs that source into a single channel. In electronics, multiplexers function as multiple-input, single-output switches. A multiplexer has multiple inputs and a selector that connects a specific input to the single output. The selection wire(s) determine which input to pass onto the output.

Where to use MUX?

Whenever we need to change a parameter to select among more than one value, we use MUX.

Example: Suppose we want to change the frequency of operation of a counter which works on a clock. How do we achieve it? Attach 4 different clocks of different frequencies in four inputs of a 4x1 MUX. Change the selection lines accordingly and let the output pass to the counter. Effectively, the output is that of 4 different clocks of 4 different frequencies.

Inputs to a MUX:

1. The inputs I_0, I_1, etc.
2. Selection Lines S_0,S_1, etc.
3. Enabled Pin

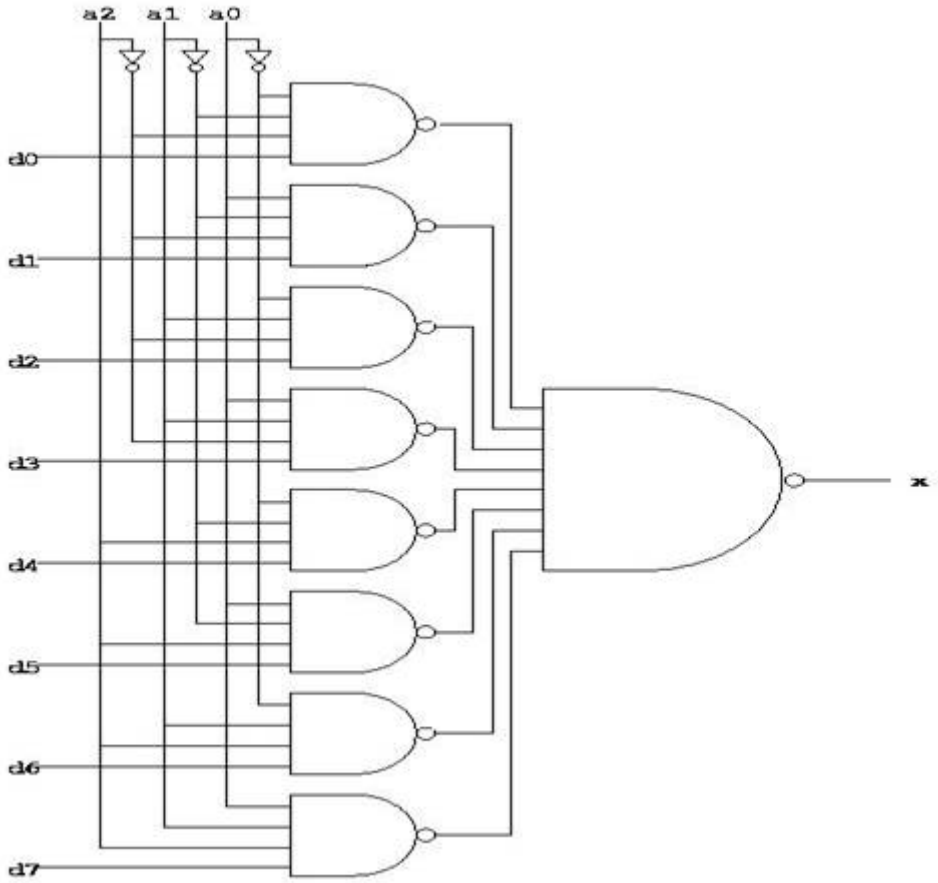
The Enabled Pin is a pin that can control if the MUX will output anything at all or not. This can thus be used if we do not want an output to be generated at all.

Suppose you are making a game where 2 players can play the game and while the game is on and a shot has been played, the players cannot fiddle with the input controls. We can achieve this in this manner.

Is Game On?
    Yes:
        Make E=1 (Enable the MUX)
        Is it player 1's turn?
            Yes: Make S=0
            No: Make S=1
    No:
        Make E=0

Now this decision tree is more complicated than the one given previously. But we see that the main theme of digital electronics runs through the proposition logic given here too, though the decision tree is becoming more complex.

Note: Sometimes the pins are active low and written with a bar on the top of the variable. The meaning is pretty simple. Let us say a RESET pin is marked active low. Now we explain the meaning what this means. When we set a pin HIGH, we intuitively mean that we want that action to be taken. For a RESET pin we think in the manner that when it is set to 1, we are asking the chip to reset and 0 means no reset. But, active low means that we should



go for the opposite here. Thus, here 0 would reset the chip and 1 would mean not to reset. The enabled pin in almost all MUXs is active low.

How selection lines determine which output to choose: Let's take a_2 a_1 a_0 as 1, 1, 0. Now $110_2=6_{10}$. Therefore the MUX will choose d_6 and pass it to the output. Refer to the adjoint diagram to figure out how the 'chip' knows which input to pass through to the output.

The MUX chips

4051…….8 x 1 Analog Multiplexer
4052…….4 x 2 Analog Multiplexer
4067……16 x 1 Analog Multiplexer
4053……Triple 2 x 1 Analog Multiplexer
74157…..Quad 2 Input Multiplexer

Note: Suppose you are working with analog signals and need to choose between 4 different inputs. You can still use these multiplexers. These simply pass the input to the output irrespective of the nature of the signal.

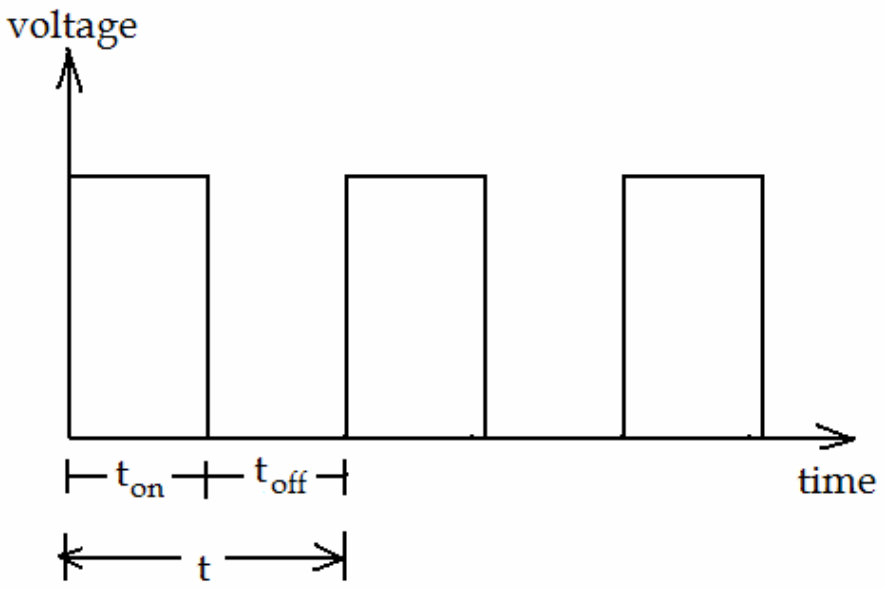
Now some circuiting basics

IC

– Integrated circuit – is a small chip which when given input, performs some specific operation on the inputs & gives some output. It is basically made of millions of transistors in such a way that the desired operation is performed.

Clock

From then on we have been telling about a clock. But what exactly is a clock?
Looking electronically, a clock is simply a square wave i.e. alternate high and low states. Each alternate high-low forms a clock cycle with a specific frequency and duty cycle.



Frequency = 1/t
Duty Cycle = t_{on}/t_{off}

Capacitors

We prefer to use electrolytic capacitors over ceramic as they are more accurate. The important point to be noted is that electrolytic capacitors must be put in the circuit with the **correct polarity**; otherwise, it bursts or gets swollen up. You can try it once to appreciate the aesthetics of it, but from a managerial viewpoint, we strongly discourage it, although we have all tried doing it once with passion. ;)

Breadboards

These are the bases on which circuits are built which already have certain holes interconnected. ***Link*:** Refer to http://www.iguanalabs.com/breadboard.htm to know the details of the interconnections.

Resistors

These are the resistors whose value can be identified from the color codes.

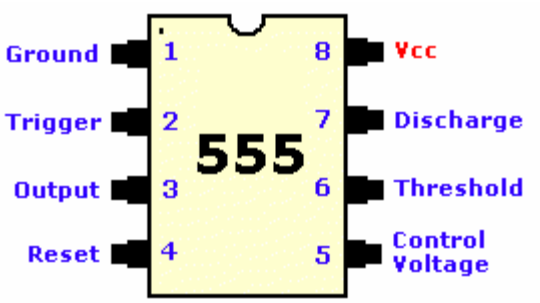
Color	1st band	2nd band	3rd band (multiplier)	4th band (tolerance)	Temp. Coefficient
Black	0	0	$\times 10^0$		
Brown	1	1	$\times 10^1$	±1% (F)	100 ppm
Red	2	2	$\times 10^2$	±2% (G)	50 ppm
Orange	3	3	$\times 10^3$		15 ppm
Yellow	4	4	$\times 10^4$		25 ppm
Green	5	5	$\times 10^5$	±0.5% (D)	
Blue	6	6	$\times 10^6$	±0.25% (C)	
Violet	7	7	$\times 10^7$	±0.1% (B)	
Gray	8	8	$\times 10^8$	±0.05% (A)	
White	9	9	$\times 10^9$		
Gold			×0.1	±5% (J)	
Silver			×0.01	±10% (K)	
None				±20% (M)	

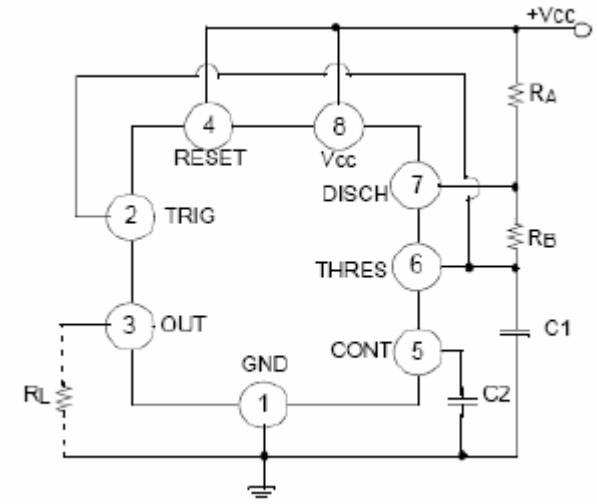
555

Introduction

This is an 8-pin IC that can be used in two well-known modes to get three different kinds of operations:

1. Astable mode
2. Monostable mode
3. Bi-stable mode (less common)





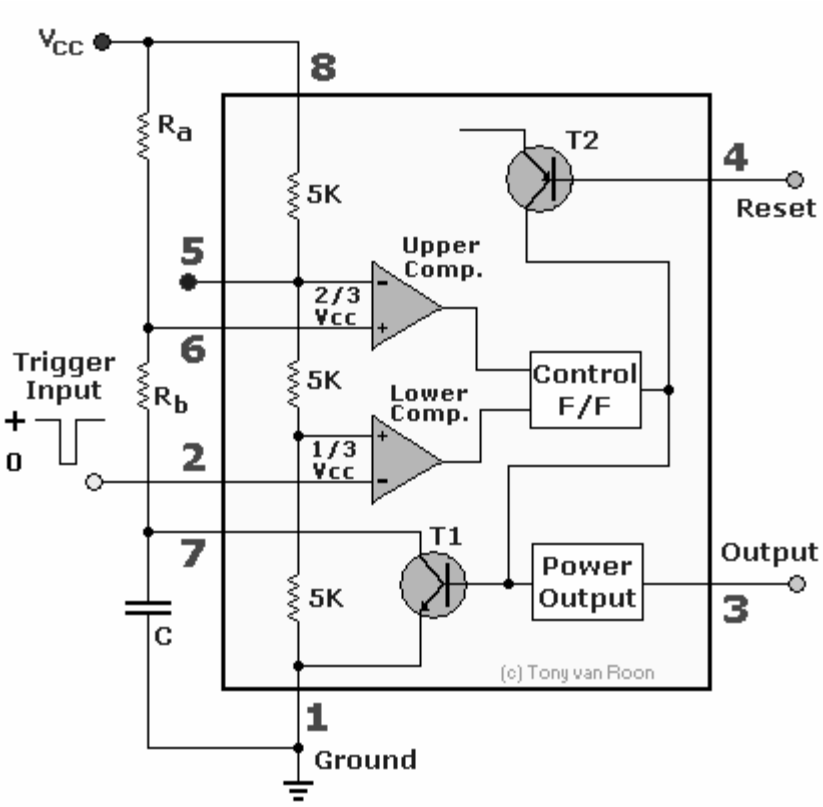
What is trigger?

Trigger is a pin that instructs the timer IC to change the state from a 0 to a 1. The trigger pin is an active low input. Refer above to know the definition of active low inputs. Whenever there is a momentary transition of the 555 from 1 to a 0, it triggers the IC to go to the 1 state. In astable mode, the IC is self triggered. Therefore it generates a periodic pulse.

What is the use of C_2?

It is usually a noise filter. Capacitors are often used as noise filters. We explain why it is so. Capacitors have impedance $Z_C = \frac{1}{j\omega C}$. Therefore, if a signal with a very high frequency is passed, the capacitor becomes a low impedance path and lets the current to flow through the branch with little voltage drop. Thus noise voltage does not get passed onto the output. Now noise signals are almost invariably high frequency signals. Thus, they get filtered.

The working of the 555 timer can be explained in terms of a simple model that we show here. The V_{c1} is the driving signal connected at PIN 6, which is the THRESHOLD PIN. As soon as V_{c1} reaches $\frac{2}{3}V_{cc}$, the output at PIN 3 goes low, and the capacitor starts discharging via PIN 7 with R_b as the resistor and ground as the other terminal. (Refer to the figure to understand it. When it reaches $\frac{1}{3}$Vcc, the output at PIN 3 goes high, and the DISCHARGE PIN's connection with the ground is broken. The capacitor again starts charging, and the cycle is repeated. The working may look a bit cumbersome but with a little experience of op-amps, it can be easily tackled. This graph gives the OUTPUT at PIN 3 and the input voltage at PIN 7.



This circuit can be conceptualized by the schematic diagram shown alongside. The working as previously explained looks fairly straightforward if you are familiar with op-amps. Work it out yourself to discover the working.

Why the strange name 555?

The chip derives its name from the three R's that are connected to the chip. These are all 5 kΩ resistors and hence the name 555.

Now to some practical stuffs

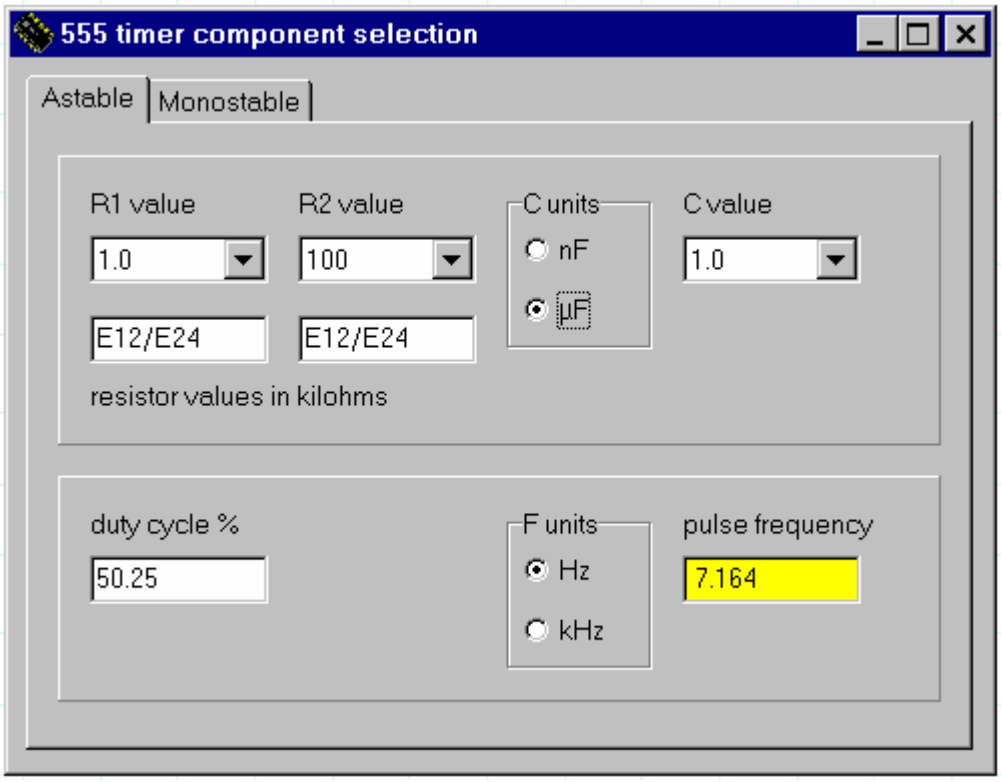
How do we get a clock of desired frequency and duty cycle?

A combination of RA, RB and C holds the key. It can be derived that:

$$\text{Frequency} = \frac{1.44}{(R_A + 2R_B)\,C_1}$$

$$\text{Duty Cycle} = \frac{100(R_A + R_B)}{(R_A + 2R_B)}\ \%$$

There exists a very useful software which uses these two equations to give you the frequency and duty cycle, given the capacitor and resistor values. It also has a few other features. Do use it to save yourself from a lot of computation.



PS: This site also provides an excellent applet based interface for calculating the needed values of resistors, etc. for various configurations of 555.
***Link*:** http://home.cogeco.ca/~rpaisley4/LM555.html

There are more details about how the monostable state of the 555 works later.

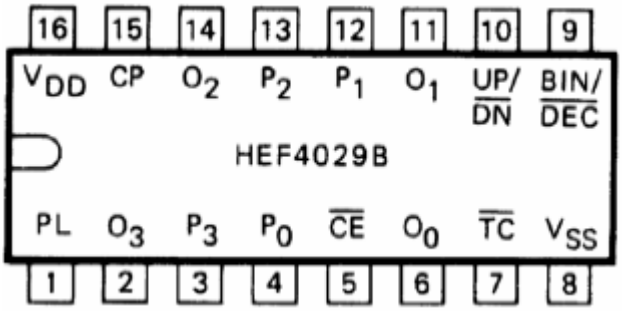
The Chip 4029

Introduction

4029 can be uses as:

- ❖ counter
- ❖ memory

This chip has 4 bits to operate on and the various pins just manipulate these 4 bits. Let's look at the work of the various pins to understand the chip better.



The pins can be divided into the following categories:

1. **CP (Clock pulse)**: This sets the tempo of counting. If we work in the counter mode the frequency of generation of numbers is achieved using the frequency of the clock pulse.
   Exercise: Try devising a counter using JK flip/flops only.
2. **CE (Clock Enabled):** This input defines whether to work in the memory mode or counter mode. Use your brain to analyze when to use this.
   An example: Suppose we want to stop counting at some moment and retain the output. What do we do? We set the clock enabled and let the 4029 count to a desired number and then simply disable the clock pulse by inverting the CE pin, which results in stopping the 4029 from responding to the clock pulse anymore. Thus we actually devise a basic level memory! But there is a small catch. Figure it out after reading the next section on parallel loads.
3. **Parallel loads:** These pins let the 4 bits to be set independently. When the 4029 is working in the memory mode (Which pin sets this? We have already answered!), these bits are simply passed onto the output, i.e., output pins are correspondingly set to the values of the parallel loads.
4. **BIN/HEX:** The counter operates on 4 bits. What is the maximum number that can be represented by 4 bits? It's obviously 15. So it can count from 0 to 15. Now in certain practical applications we want a counter to count in decimal mode, i.e., from 0 to 9. This can be achieved through this pin. This pin tells 4029 whether to operate in decimal mode or binary mode. For exact settings refer to the datasheet.
   Exercise: Try devising a strategy by which the counter will count in the decimal mode even if the 4029 is instructed to go into binary mode. *Hint:* Try to flush the 4 bits to 0000 as soon as the value 10 appears, through parallel load.
5. **UP/DOWN:** This asks the 4029 whether to count up from 0 onwards or to count down from 15 or 9 as the case maybe.
6. **TC:** This is a pin that goes HIGH once the first cycle is over. This can be used in various applications where counting once has a special significance.
7. **Output pins:** These are the output bits that give the status of the 4 bits in operation.

8. **Some common pins:** The ground/V_{CC} pins are common to all ICs. Remember to properly connect these. This is where maximum mistakes are committed. So beware.

Exercise: Implement the entire 4029 using flip/flops and other gates. Refer to the next section for a hint.

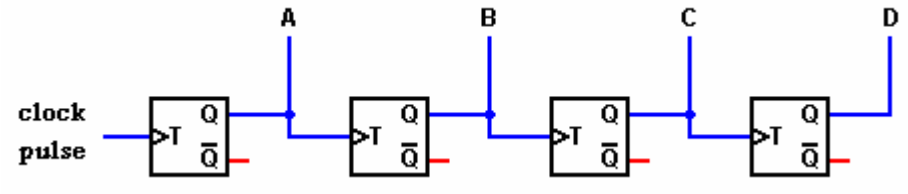
Counters

Generally counters are of two kinds:

- Synchronous
- Asynchronous

We seldom use the synchronous counter in developing circuits. For the sake of completeness we present a link for your reference on synchronous counter:
http://www.allaboutcircuits.com/vol_4/chpt_11/3.html

Now let us develop a simple asynchronous counter using a single clock pulse.



This is an example of a 4-bit asynchronous counter where the input is a single clock pulse and the four output bits A, B, C, D form the nibble representation of the number to be output. You can easily remove one T flip/flop from here and still get a similar counter. Can you see how?

Note: Nibble = a sequence of 4 bits.

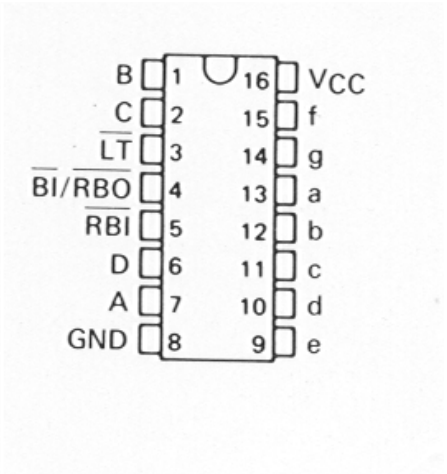
This circuit can easily be realized using a J-K flip/flop. How do you make a T-flip/flop using a J-K flip/flop? Refer to the section of flip/flops to know it.

Working With BCD to 7 Segment Decoder 7447

Introduction:

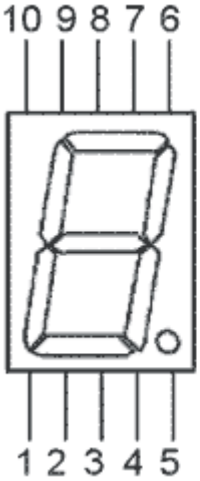
The BCD to 7 Segment Decoder **IC7447** takes in the 4 output bits and decodes it such that the corresponding digit in decimal form can be shown on the **7 Segment Display**. The output here are the pins numbered 9 to 15 named a to g as shown in the pin out diagram. These 7 bits are to be connected to the corresponding pins (to be identified by the alphabets) via a resistor, preferably a 220Ω or 330Ω. (Think about the reason resistor is used). The pin connections in this chip are:

- **Pin 1,2, 6,7** – Input pins
- **Pin 8** – Ground
- **Pin 16** – 5 V
- **Pin 9 to 15** – To 7 Segment Display
- **Pin 3** – To test whether the 7 segment is defective. If it is high, all the 7 segments must glow. It has to be connected to Ground while working with the chip.
- **Pin 4 , 5** – Ground



7 Segment Display

The 7 Segment Display consists of 7 LEDs which can be glowed in some particular fashion using the 7 different input pins. For example, if A, B, G, E, D glow, numeral 2 is represented. Cathode has to be connected to 5 V. The decimal is optional as it is used to glow the decimal point in bottom right of the 7 segment display.



1	Cathode E
2	Cathode D
3	Com. Anode
4	Cathode C
5	Cathode D.P.
6	Cathode B
7	Cathode A
8	Com. Anode
9	Cathode F
10	Cathode G

Important Tips While Circuiting

• Interconnect the horizontal lines & use them for ground & 5V purposes so that ground & high can be provided to any pin in the circuit easily.
• Handle the chips carefully. **And always perform the lamp test prior to using the 7 segment.** It will save you a great deal with respect to the trouble shooting that you will need to do.
• Always keep in mind to give the chip High & Low at the corresponding place i.e. V_{SS} (or ground) & V_{DD}(or V_{CC}) to make the chip work.
• Never leave any input of the chip in floating position i.e. not connected as that can affect the output by introducing noise.
• In digital electronics, keep an LED ready for testing & debugging purposes. An LED should be connected to a resistor (220 Ω or 330 Ω) and a wire, so that it is

flexible. Believe it or not, this is the most useful debugging tool digital electronics has to offer.

• When using resistors, do check that the legs do not touch each other.

Things to Ponder

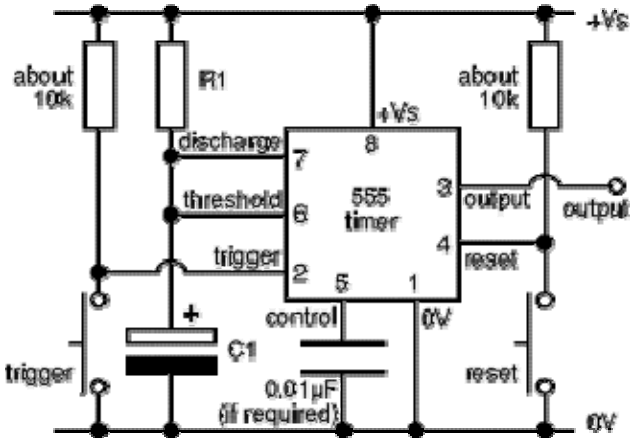
- ✓ Why Is LED connected along with a resistor?
- ✓ Why did we connect each pin of 7 segment display through a resistor?
- ✓ Is there any other way also that would use lesser number of resistors and serve the same purpose? If you can find a way out, what is the problem in this scheme?
- ✓ Is there a way that you can make your circuit safe against any cross connection that takes place between high & low in your circuit?
- ✓ If you used 555 with a different duty cycle but same frequency, how would it affect the counter?
- ✓ Can you think how you could use a 4029 as a memory device? (*Hint*: Parallel Load)
- ✓ Can you upgrade this counter of yours to multiple displays i.e. 2 displays, one for seconds and one for minutes (count up to 10 minutes only) working like a clock i.e. after every 9 sec, the tens digit is incremented and after 59 seconds, the seconds display should go to 00 and minute is incremented? (hint: Parallel load & TC)

The 555 monostable mode operation

Introduction

This is the use of LM555 timer for pulse generation, one of the many facets of this wonderful chip. This pulse generation is obtained from a trigger generated by some outside source (say a switch). Its specifically useful while designing push button switches as it eliminates the bouncing effect. This is one of the major use of 555. Though there are other ways of circumventing this problem.

You can generate a stable pulse of a desirable width from 555 even if the input trigger pulse is erratic and noise prone. So 555 can also act as a buffer of sorts. The circuit necessary to make the chip work in the monostable mode is given below:



The pin arrangement must be sufficiently clear by now, or rather you will have a good idea where to look for the meanings of the pins, so I am not repeating them here. The behavior of this circuit will be discussed here.

The resistance used in series with the switch is of a very high value since the voltage drop across it must be at least $2/3 V_{cc}$ for the trigger (pin 2) to work. Details follow.

$$\mathbf{T_{on} = 1.1 \times R1 \times C1}$$

Practically, not all the values of Resistance and capacitance is allowed, since we have only a few selected values. Hence, choose the Capacitance first (more restricted) and then try and match the resistance to the needed value. Also remember that the maximum length of the pulse can be 10min. So this chip can be exploited heartily.

Note: The interesting thing to note here is the factor 1.1 in the T_{ON} calculation. It comes in because we charge the Capacitor to 67% and not to e^{-1} (63%) as when calculating the time constant.

Note: If you are planning to make a pulse with a very high T_{ON} then keep in mind that the capacitor leakage current may play a significant role. So try choosing lower Capacitance while increasing the resistance. This would give a better precision.

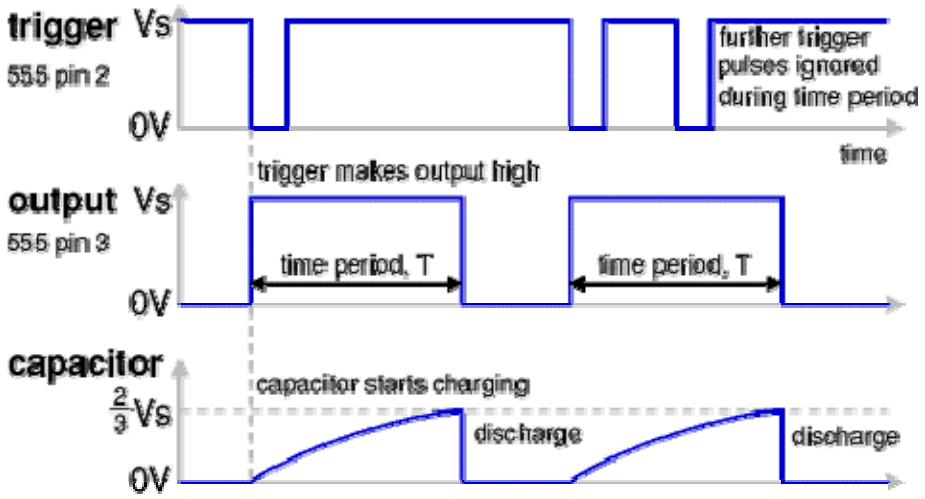
Basic Output Discussion

- The pin 2 (Trigger) is the pin connected to the switch will generates the initial disturbance ('triggers' the chip). It is the negative fall which is accepted as an input.

- The timing period is triggered (started) when the **trigger** input (555 pin 2) is less than $^{1}/_{3}$Vs, this makes the **output** high (+Vs) and the capacitor C starts to charge through resistor R. Once the time period has started, further trigger pulses are ignored.

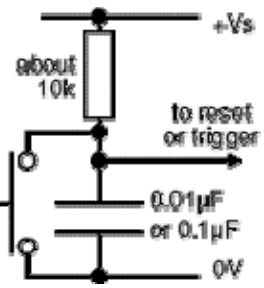
- The **threshold** input (555 pin 6) monitors the voltage across C1 and when this reaches $^{2}/_{3}$Vs the time period is over and the **output** becomes low. At the same time **discharge** (555 pin 7) is connected to 0V, discharging the capacitor ready for the next trigger.

- The **reset** input (555 pin 4) overrides all other inputs and the timing may be canceled at any time by connecting reset to 0V, this instantly makes the output low and discharges the capacitor. If the reset function is not required the reset pin should be connected to +Vs.

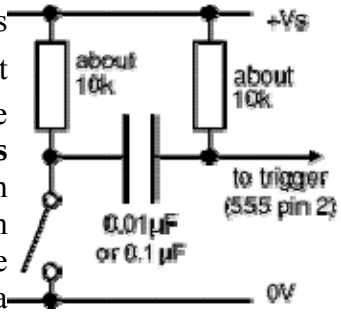


Useful Tips

- *Triggering on Power on:* It may be useful to ensure that a monostable circuit is reset or triggered automatically when the power supply is connected or switched on. This is achieved by using a capacitor instead of (or in addition to) a push switch as shown in the diagram. The capacitor takes a short time to charge, briefly holding the input close to 0V when the circuit is switched on. A switch may be connected in parallel with the capacitor if manual operation is also required.



- *Edge triggering:* If the trigger input is still less than $^1/_3$Vs at the end of the time period the output will remain high until the trigger is greater than $^1/_3$Vs. This situation can occur if the input signal is from an on-off switch or sensor. The monostable can be made **edge triggered**, responding only to **changes** of an input signal, by connecting the trigger signal through a capacitor to the trigger input. The capacitor passes sudden changes (AC) but blocks a constant (DC) signal. The circuit is 'negative edge triggered' because it responds to a sudden fall in the input signal.



- *De-bouncing:* If you are planning to use the 555 as a latch, then a permanent problem will be the length of the delay to introduce so as to avoid bouncing and allow quick switch presses. The ideal value of the delay is nearly 100ms, for which the settings are 100k Ohm and 1uF.

More Help

The datasheet of 555 is excellently laid out and probably you will find whatever you are looking for in there. Also, check out the following sites, the first site particularly has excellent applets which will make the calculations a jiffy:

Link: http://home.cogeco.ca/~rpaisley4/LM555.html

Link: http://www.kpsec.freeuk.com/555timer.htm

Or better still, come to us. :)

Note: The last way is (ahem) NOT (..um... err....) recommended, [because we are lazy, you see. ;)]

Flip Flops

Introduction

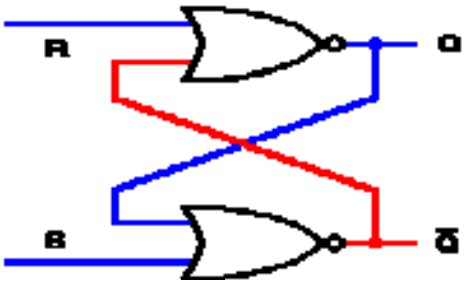
In digital circuits, the flip-flop, latch, or bistable multivibrator is an electronic circuit which has two stable states and thereby is capable of serving as one bit of memory. A flip-flop is controlled by one or two control signals and/or a gate or clock signal. The output often includes the complement as well as the normal output. As flip-flops are implemented electronically, they naturally also require power and ground connections.

Flip-flops can be either simple or clocked. Simple flip-flops consist of two cross-coupled inverting elements – transistors, or NAND, or NOR-gates – perhaps augmented by some enable/disable (gating) mechanism. Clocked devices are specially designed for synchronous (time-discrete) systems and therefore ignore its inputs except at the transition of a dedicated clock signal (known as clocking, pulsing, or strobing). This causes the flip-flop to either change or retain its output signal based upon the values of the input signals at the transition. Some flip-flops change output on the rising edge of the clock, other on the falling edge.

Clocked flip-flops are typically implemented as master-slave devices* where two basic flip-flops (plus some additional logic) collaborates to make it insensitive to spikes and noise between the short clock transitions; they nevertheless also often include asynchronous clear or set inputs which may be used to change the current output independent of the clock.

Flip-flops can be further divided into types that have found common applicability in both asynchronous and clocked sequential systems: the SR ("set-reset"), D ("data"), T ("toggle"), and JK types are the common ones; all of which may be synthesized from (most) other types by a few logic gates. The behavior of a particular type can be described by what is termed the characteristic equation, which derives the "next" (i.e., after the next clock pulse) output, Q_{next}, in terms of the input signal(s) and/or the current output, Q.

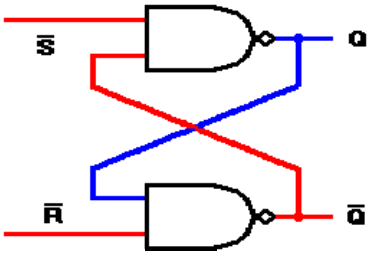
Basic R-S NAND Latch



In order for a logical circuit to "remember" and retain its logical state even after the controlling input signal(s) have been removed, it is necessary for the circuit to include some form of feedback. We might start with a pair of inverters, each having its input connected to the other's output. The two outputs will always have opposite logic levels.

The problem with this is that we don't have any additional inputs that we can use to change the logic states if we want. We can solve this problem by replacing the inverters with NAND or NOR gates, and using the extra input lines to control the circuit.

The circuit shown below is a basic NAND latch. The inputs are generally designated "S" and "R" for "Set" and "Reset" respectively. Because the NAND inputs must normally be logic 1 to avoid affecting the latching action, the inputs are considered to be inverted in this circuit.



S	R	Q_{n+1}	Q'_{n+1}
1	0	0	1
0	1	1	0
1	1	Q_n	Q'_n
0	*0*	*1*	*1*

The outputs of any single-bit latch or memory are traditionally designated Q and Q'. In a commercial latch circuit, either or both of these may be available for use by other circuits. In any case, the circuit itself is:

For the NAND latch circuit, both inputs should normally be at a logic 1 level. Changing an input to a logic 0 level will force that output to a logic 1. The same logic 1 will also be applied to the second input of the other NAND gate, allowing that output to fall to a logic 0 level. This in turn feeds back to the second input of the original gate, forcing its output to remain at logic 1.

Applying another logic 0 input to the same gate will have no further effect on this circuit. However, applying a logic 0 to the *other* gate will cause the same reaction in the other direction, thus changing the state of the latch circuit the other way.

Note that it is forbidden to have both inputs at a logic 0 level at the same time. That state will force both outputs to a logic 1, overriding the feedback latching action. In this condition, whichever input goes to logic 1 first will lose control, while the other input (still at logic 0) controls the resulting state of the latch. If both inputs go to logic 1 simultaneously, the result is a "race" condition, and the final state of the latch cannot be determined ahead of time.

To understand how this thing actually works, log on to
http://isweb.redwoods.cc.ca.us/INSTRUCT/CalderwoodD/diglogic/srflip.htm

Basic NOR latch

While most of our demonstration circuits use NAND gates, the same functions can also be performed using NOR gates. A few adjustments must be made to allow for the difference in the logic function, but the logic involved is quite similar.

The circuit shown below is a basic NOR latch. The inputs are generally designated "S" and "R" for "Set" and "Reset" respectively. Because the NOR inputs must normally be logic 0 to avoid overriding the latching action, the inputs are not inverted in this circuit. The NOR-based latch circuit is:

S	R	Q_{n+1}	Q'_{n+1}
1	0	1	0
0	1	0	1
1	*1*	*0*	*0*
0	*0*	Q_n	Q'_n

For the NOR latch circuit, both inputs should normally be at a logic 0 level. Changing an input to logic 1 level will force the output to logic 0. The same logic 0 will also be applied to the second input of the other NOR gate, allowing the output to rise to the level of logic 1. This in turn feeds back to the second input of the original gate, forcing its output to remain at logic 0 even after the external input is removed.

Applying another logic 1 input to the same gate will have no further effect on this circuit. However, applying a logic 1 to the *other* gate will cause the same reaction in the other direction, thus changing the state of the latch circuit the other way.

Note that it is forbidden to have both inputs at a logic 1 level at the same time. That state will force both outputs to a logic 0, overriding the feedback latching action. In this condition, whichever input goes to logic 0 first will lose control, while the other input (still at logic 1) controls the resulting state of the latch. If both inputs go to logic 0 simultaneously, the result is a "race" condition, and the final state of the latch cannot be determined ahead of time.

Q_n	**S**	**R**	**Q_{n+1}**
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	--
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	--

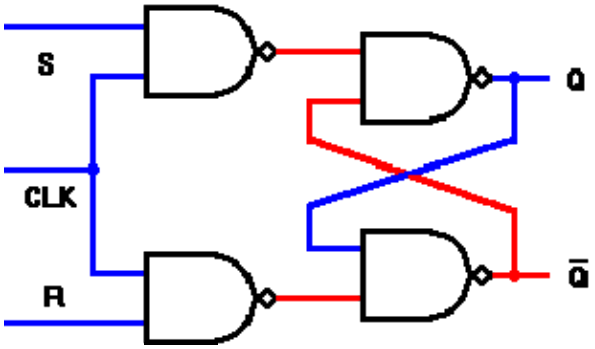
Clocked RS Flip/Flop

By adding a pair of NAND gates to the input circuits of the RS latch, we accomplish two goals: normal rather than inverted inputs and a third input common to both gates which we can use to synchronize this circuit with others of its kind.

The clocked RS NAND latch is shown below.

The clocked RS latch circuit is very similar in operation to the basic latch you examined on the previous page. The S and R inputs are normally at logic 0, and must be changed to logic 1 to change the state of the latch. However, with the third input, a new factor has

been added. This input is typically designated *C* or *CLK*, because it is typically controlled by a clock circuit of some sort, which is used to synchronize several of these latch circuits with each other. The output can only change state while the CLK input is a logic 1. When CLK is a logic 0, the S and R inputs will have no effect.



The same rule about not activating both the S and R inputs simultaneously holds true: if both are logic 1 when the clock is also logic 1, the latching action is bypassed and both outputs will go to logic 1. The difference in this case is that if the CLK input drops to logic 0 first, there is no question or doubt -- a true race condition will exist, and you cannot tell which way the outputs will come to rest. The example circuit on this page reflects this uncertainty.

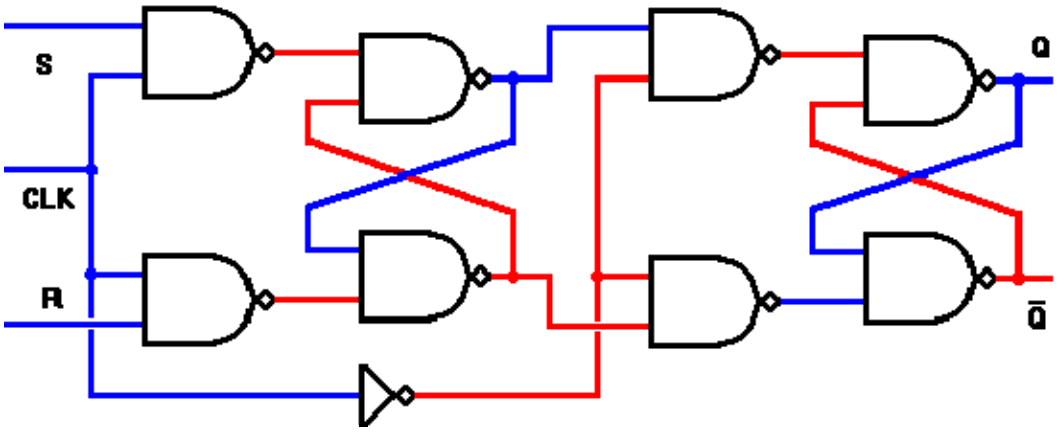
For correct operation, the selected R or S input should be brought to logic 1, the CLK input should be made logic 1 and then logic 0 again. Finally, the selected input should be returned to logic 0.

The clocked RS latch solves some of the problems of basic RS latch circuit, and allows closer control of the latching action. However, it is by no means a complete solution. A major problem remaining is that this latch circuit could easily experience a change in S and R input levels while the CLK input is still at a logic 1 level. This allows the circuit to change state many times before the CLK input returns to logic 0.

One way to minimize this problem is to keep the CLK at logic 0 most of the time, and to allow only brief changes to logic 1. However, this approach still cannot guarantee that the latch will only change state once while the clock signal is at logic 1. This signal must have certain duration to ensure that all latches have time to respond to it, and in that time, most latches can respond to multiple changes.

Edge-Triggered R/S Flip Flop:

The circuit of edge-triggered flip-flop is given as below and the truth table follows, the working must be clear by now and one should try to solve the circuit and try to get the truth table



The edge-triggered RS flip-flop actually consists of two identical RS latch circuits, as shown above. However, the inverter connected between the two CLK inputs ensures that the two sections will be enabled during opposite half-cycles of the clock signal. This is the key to the operation of this circuit.

If we start with the CLK input at logic 0 as initially depicted above, the S and R inputs are disconnected from the input (master) latch. Therefore, any changes in the input signals cannot affect the state of the final outputs.

When the CLK signal goes to logic 1, the S and R inputs are able to control the state of the input latch, just as with the single RS latch circuit you already examined. However, at the same time the inverted CLK signal applied to the output (slave) latch prevents the state of the input latch from having any effect here. Therefore, any changes in the R and S input signals are tracked by the input latch while CLK is at logic 1, but are not reflected at the Q and Q' outputs.

When CLK falls again to logic 0, the S and R inputs are again isolated from the input latch. At the same time, the inverted CLK signal now allows the current state of the input latch to reach the output latch. Therefore, the Q and Q' outputs can only change state when the CLK signal falls from a logic 1 to logic 0. This is known as the *falling edge* of the CLK signal; hence the designation *edge-triggered* flip-flop.

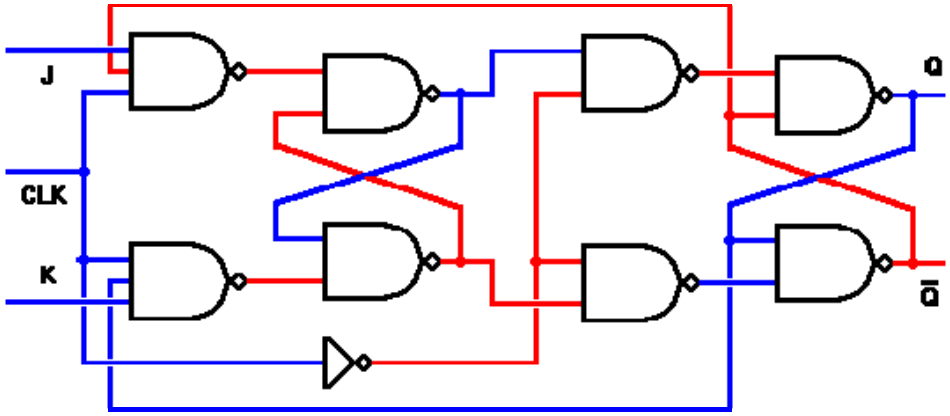
By going to a master-slave structure and making the flip-flop edge-triggered, we have made sure that we can precisely control the moment when all flip-flops will change state. We have also allowed plenty of time for the master latch to respond to the input

signals, and for those input signals to change and settle following the previous change of state.

So, now as we are all geared up with the basic knowledge of Flip- Flops, lets look into the multipurpose J-K Flip Flop.

J-K Flip Flop

To the RS flip-flop we have added two new connections from the Q and Q' outputs back to the original input gates. Remember that a NAND gate may have any number of inputs, so this causes no trouble. To show that we have done this, we change the designations of the logic inputs and of the flip-flop itself. The inputs are now designated J (instead of S) and K (instead of R)

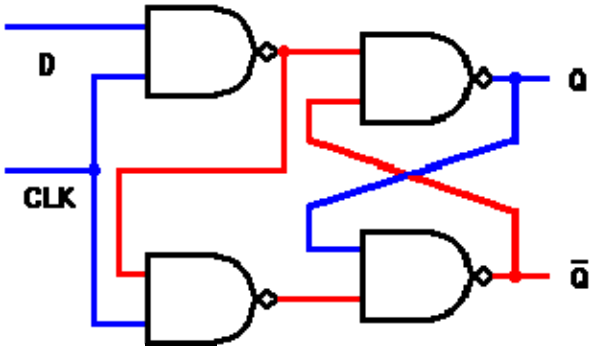


:The J-K Flip Flop is similar to R-S Flip Flop but for the following worthy differences

- Since one of the two logic inputs is always disabled according to the output state of the overall flip-flop, the master latch cannot change state back and forth while the CLK input is at logic 1. Instead, the enabled input can change the state of the master latch *once*, after which this latch will not change again. This was not true of the RS flip-flop.
- If both the J and K inputs are held at logic 1 and the CLK signal continues to change, the Q and Q' outputs will simply change state with each falling edge of the CLK signal. (The master latch circuit will change state with each *rising* edge of CLK.) We can use this characteristic to advantage in a number of ways. A flip-flop built specifically to operate this way is typically designated as a *T* (for *Toggle*) flip-flop. The lone T input is in fact the CLK input for other types of flip-flops.
- The JK flip-flop *must* be edge triggered in this manner. Any level-triggered JK latch circuit will oscillate rapidly if all three inputs are held at logic 1. This is not very useful. For the same reason, the T flip-flop must also be edge triggered. For both types, this is the only way to ensure that the flip-flop will change state only once on any given clock pulse.

So, here we close the theoretical discussion on the flip flops & lets look at more practical applications of the flip flops. But before that, let's see D & T Flip Flops which find a variety of application.

D Type Flip Flop

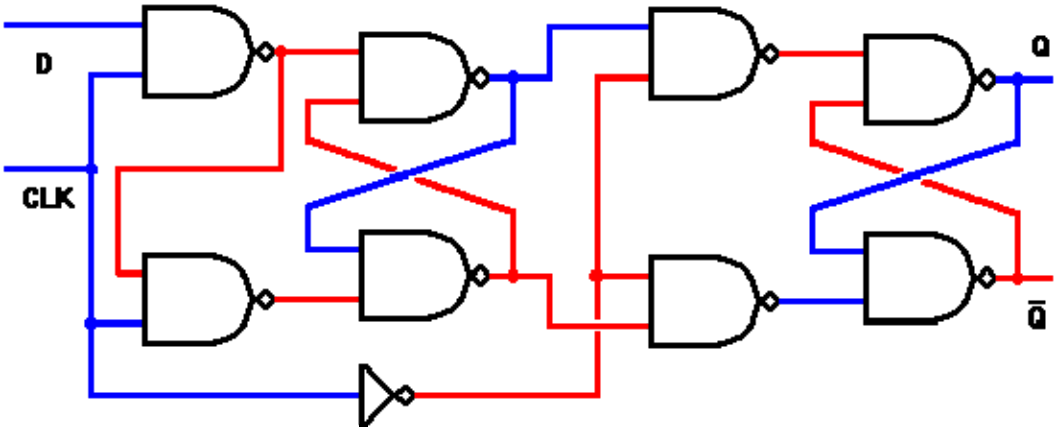


In the D latch, when the CLK input is logic 1, the Q output will always reflect the logic level present at the D input, no matter how that changes. When the CLK input falls to logic 0, the last state of the D input is trapped and held in the latch, for use by whatever other circuits may need this signal.

Because the single D input is also inverted to provide the signal to reset the latch, this latch circuit cannot experience a "race" condition caused by all inputs being at logic 1 simultaneously. Therefore the D latch circuit can be safely used in any circuit.

The Truth Table is pretty easy, on each clock pulse, we have:

D	Q
0	0
1	1



For an edge triggered D Flip Flop, we can modify the latch as:

T Flip Flops

If you have understood the D flip flops pretty well. Then try to think what changes in the above flip flops would give an output as this on clock pulses:

T	Q
0	Q
1	Q'

Applications of Flip Flops

The first application of flip flops is for memory & that is easily figured out from the above discussion. Let's look into one of the most interesting applications of flip flops: Counter.

As discussed before, the counters are basically of two types: Synchronous & Asynchronous. The only difference between the two is that synchronous have the same clock going in all the flip flops. While in asynchronous, clocks that go in the flip flops are different.

Here we are providing you with some counter circuits & their truth tables, try to figure out how the design had been achieved and then do try type of counters given later:

Binary Counter:

To determine the gates required at each flip-flop input, let's start by drawing up a truth table for all states of the counter. Such a table is shown to the right.

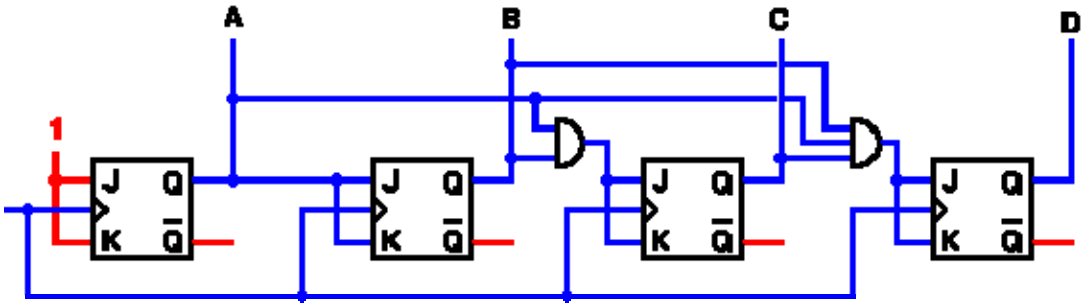
Looking first at output A, we note that it must change state with every input clock pulse. Therefore, we *could* use a T flip-flop here if we wanted to. We won't do so, just to make all of our flip-flops the same. But even with JK flip-flops, all we need to do here is to connect both the J and K inputs of this flip-flop to logic 1 in order to get the correct activity.

Flip-flop B is a bit more complicated. This output must change state only on every *other* input clock pulse. Looking at the truth table again, output B must be ready to change states whenever output A is a logic 1, but not when A is a logic 0. If we recall the behavior of the JK flip-flop, we can see that if we connect output A to the J and K inputs of flip-flop B, we will see output B behaving correctly.

Continuing this line of reasoning, output C may change state only when both A and B are logic 1. We can't use only output B as the control for flip-flop C; that will allow C to change state when the counter is in state 2, causing it to switch directly from a count of 2 to a count of 7, and again from a count of 10 to a count of 15 — not a good way to count. Therefore we will need a two-input AND gate at the inputs to flip-

State				Count
D	C	B	A	
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	10
1	0	1	1	11
1	1	0	0	12
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15

flop C. Flip-flip D requires a three-input AND gate for its control, as outputs A, B, and C must all be at logic 1 before D can be allowed to change state.



The resulting circuit is:

Now, try to design a counter that counts from 0 to 9 only. Also, try to approach these problems by T and D flip flops. When you are done with these, try designing down counters. (H*int:* use Q') Also, refer to asynchronous counters as in the link mentioned before.

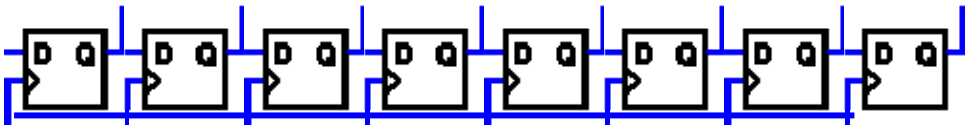
Registers:

The term *register* can be used in a variety of specific applications, but in all cases it refers to a group of flip-flops operating as a coherent unit to hold data. This is different from a counter, which is a group of flip-flops operating to generate new data by tabulating it.

In this context, a counter can be viewed as a specialized kind of register, which counts events and thereby generates data, rather than just holding the data or changing the way it is handled. More commonly, however, counters are treated separately from registers. The two are then handled as separate concepts which work together in many applications, and which have some features in common.

The demonstration circuit below is known as a *shift register* because data is shifted through it, from flip-flop to flip-flop. If you apply one *byte* (8 bits) of data to the initial data input one bit at a time, and apply one clock pulse to the circuit after setting each bit of data, you will find the entire byte present at the flip-flop outputs in parallel format. Therefore, this circuit is known as a serial-in, parallel-out shift register. It is also known sometimes as a shift-in register, or as a serial-to-parallel shift register.

By standardized convention, the least significant bit (LSB) of the byte is shifted in first.



These registers find enormous application in memory devices. Here, we have demonstrated just the left shift of bits, similarly we can have the right shift of bits and combining both of them, we have the Shift Register. For more information, refer to the datasheets of chip – **74194.**

Flip/Flop Chips

4013 7474 Dual Delay Flip Flop

40175 Quad Delay Flip Flop

4027 Dual Master/Slave JK Flip Flop

74112 Dual JK Flip Flop negative edge

74173 Quad Delay Flip Flop (3 state)

74273 Octal Delay Flip Flop

74574 Octal Delay Flip Flop (3 state)

74194 4 bit bidirectional Shift Register

74164 8 bit serial in parallel out Shift Register

How to read a Datasheet

In order to use any electronics at all, there is one necessary document which you will have to deal with; the Datasheet. On the first glance, it looks like a big monstrous work of literature that you would probably credit to Homer or perhaps, looking at the graphic side of things, Frank Miller. Or, on the other side, you may be tempted to keep it aside for a more "appropriate time" when you will be able to understand everything, when you are a good engineer and an expert in these things.

However, the trouble is that to begin **anything at all** in electronics, you need to read a Datasheet. Moreover, all the information present in the datasheet, ***very little is of actual use to us.*** The information in the sheets is succinct and **complete**, the purpose of the sheet being that absolutely no other reference material will be needed for the chip. These include:

• The typical device performance
• Minimum and maximum requirements and characteristics
• What you can do to the device without harming it
• Suggested uses and hints
• The layout and the size of the chips.

We will closely follow the data sheet of 74HC/HCT194 as an example. On the first look itself, it looks like a formidable data sheet, warning us NOT to probe any further unless we have in our possession **three** other datasheets!! However, rest assured that this will be the only datasheet that will be of critical value to us.

Let's begin page by page.

The First page

This tells you about the general features of the chip. It does make for an interesting read, which may include a few things that the rest of the datasheet may have only in fine print!!!

Apart from that, the rest of the material is for general knowledge of the chip. It is strongly recommended that you read this to become familiar with the chip, though (sheepishly) this is not necessary. (Lets admit it, almost nobody ever reads it.)

The table will be indispensable for detailed study of the datasheet later. It is nice to at least make a note of the place where the symbols are defined, no mater which datasheet you are reading. Sooner or later, you will come back to it. Also, though we will only loosely refer to the values in the table, note that the values listed are for two different chips, HCT and HC. Here, you ought to take care of which chip are you actually using. This is a minor place where mistakes may be made.

4-bit bidirectional universal shift register 74HC/HCT194

FEATURES

- Shift-left and shift-right capability
- Synchronous parallel and serial data transfer
- Easily expanded for both serial and parallel operation
- Asynchronous master reset
- Hold ("do nothing") mode
- Output capability: standard
- I_{CC} category: MSI

GENERAL DESCRIPTION

The 74HC/HCT194 are high-speed Si-gate CMOS devices and are pin compatible with low power Schottky TTL (LSTTL). They are specified in compliance with JEDEC standard no. 7A.

The functional characteristics of the 74HC/HCT194 4-bit bidirectional universal shift registers are indicated in the logic diagram and function table. The registers are fully synchronous.

The "194" design has special features which increase the range of application. The synchronous operation of the device is determined by the mode select inputs (S_0, S_1). As shown in the mode select table, data can be entered and shifted from left to right ($Q_0 \rightarrow Q_1 \rightarrow Q_2$, etc.) or, right to left ($Q_3 \rightarrow Q_2 \rightarrow Q_1$, etc.) or parallel data can be entered, loading all 4 bits of the register simultaneously. When both S_0 and S_1 are LOW, existing data is retained in a hold ("do nothing") mode. The first and last stages provide D-type serial data inputs (D_{SR}, D_{SL}) to allow multistage shift right or shift left data transfers without interfering with parallel load operation.

Mode select and data inputs are edge-triggered, responding only to the LOW-to-HIGH transition of the clock (CP). Therefore, the only timing restriction is that the mode control and selected data inputs must be stable one set-up time prior to the positive transition of the clock pulse.

The four parallel data inputs (D_0 to D_3) are D-type inputs. Data appearing on the D_0 to D_3 inputs, when S_0 and S_1 are HIGH, is transferred to the Q_0 to Q_3 outputs respectively, following the next LOW-to-HIGH transition of the clock. When LOW, the asynchronous master reset ($\overline{MR}$) overrides all other input conditions and forces the Q outputs LOW.

The "194" is similar in operation to the "195" universal shift register, with added features of shift-left without external connections and hold ("do nothing") modes of operation.

QUICK REFERENCE DATA

GND = 0 V; T_{amb} = 25 °C; $t_r = t_f$ = 6 ns

SYMBOL	PARAMETER	CONDITIONS	TYPICAL		UNIT
			HC	HCT	
t_{PHL}/ t_{PLH}	propagation delay	C_L = 15 pF; V_{CC} = 5 V			
	CP to Q_n		14	15	ns
t_{PHL}	$\overline{MR}$ to Q_n		11	15	ns
f_{max}	maximum clock frequency		102	77	MHz
C_I	input capacitance		3.5	3.5	pF
C_{PD}	power dissipation capacitance per package	notes 1 and 2	40	40	pF

Notes

1. C_{PD} is used to determine the dynamic power dissipation (P_D in μW):

   $P_D = C_{PD} \times V_{CC}^2 \times f_i + \sum (C_L \times V_{CC}^2 \times f_o)$ where:

   f_i = input frequency in MHz

   f_o = output frequency in MHz

   $\sum = (C_L \times V_{CC}^2 \times f_o)$ = sum of outputs

   C_L = output load capacitance in pF

   V_{CC} = supply voltage in V
2. For HC the condition is V_I = GND to V_{CC}; for HCT the condition is V_I = GND to V_{CC} – 1.5 V

The Second page

This page contains the pin layout of the chip. The important things to take note of are that **names** of the pins. They give a fairly good idea about what is what. This will be the **first** most important page that we will be using in this (or any, for that matter) datasheet while you are debugging. The connections are the primary nuisance in terms of trouble making.

The diagram are given below in standard formats; every format having its own pros and cons, but any is good enough for understanding the basic connections that you will have to make to make the thing work. As an extra information, the IEC is the International Electrotechnical Commission. Also, the typical setup of the chip is generally appended towards the end of the datasheet. So be sure to be through with the datasheet (at least look at the titles) once before beginning to use the circuit. Again, keeping track of the important definitions of symbols/terms is definitely NOT a bad idea.

Just in case you are interested in making sense of the things given in the table, these might help:

t_r := Rising time of the output
t_f := Falling time of the output
T_{amb} := Ambient temperature.
GND := Ground.

These values also play a part in the latter AC characteristics, so they will be repeated there.

4-bit bidirectional universal shift register 74HC/HCT194

ORDERING INFORMATION

See *"74HC/HCT/HCU/HCMOS Logic Package Information"*.

PIN DESCRIPTION

PIN NO.	SYMBOL	NAME AND FUNCTION
1	$\overline{MR}$	asynchronous master reset input (active LOW)
2	D_{SR}	serial data input (shift right)
3, 4, 5, 6	D_0 to D_3	parallel data inputs
7	D_{SL}	serial data input (shift left)
8	GND	ground (0 V)
9, 10	S_0, S_1	mode control inputs
11	CP	clock input (LOW-to-HIGH edge-triggered)
15, 14, 13, 12	Q_0 to Q_3	parallel outputs
16	V_{CC}	positive supply voltage

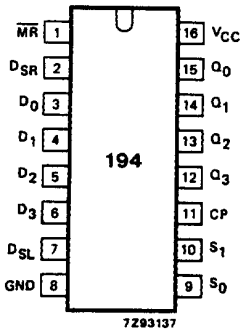


Fig.1 Pin configuration.

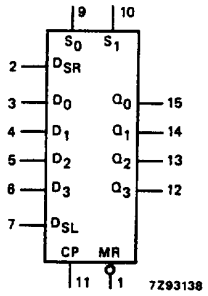


Fig.2 Logic symbol.

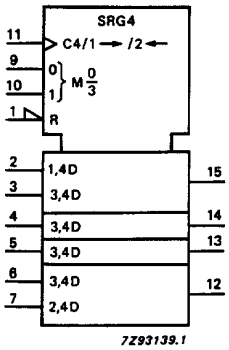


Fig.3 IEC logic symbol.

The Third page

The third page presents a Truth Table; one that might look alien on the first glance, but is very easy to understand once you are through with the definitions of the various symbols given below. This page is best understood if you already have a idea what the chip is about, so don't rack your gray cells too much if you have difficulty in interpreting what is going on.

Remember, the truth table is the most succinct way of delivering information and once you know what the various pins mean to you, (Q_1,Q_2,CP, etc.) in terms of the circuit, only then the real meaning of the Truth table will dawn on you. So make sure what it is you want the chip for before understanding from scratch what the chip does.

Also, maybe the manufacturer makes more than one chip which does related work. Their Truth tables will be common, though their physical make and characteristics may differ.

Also, the first thing that I expect you, if you are not an eagle eyed reader, to overlook on the datasheet is that the Capital (H,L) and small (h,l) have different meanings. Anyways, the transition diagram on the next sheet will take care of this overlook, if you look at it hard enough. :)

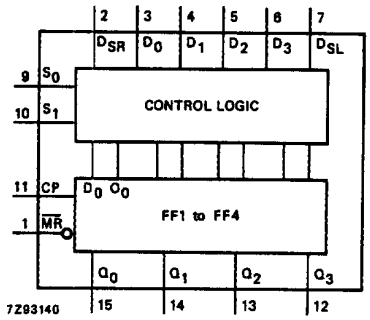


Fig.4 Functional diagram.

FUNCTION TABLE

OPERATING MODES	INPUTS							OUTPUTS			
	CP	$\overline{MR}$	S_1	S_0	D_{SR}	D_{SL}	D_n	Q_0	Q_1	Q_2	Q_3
reset (clear)	X	L	X	X	X	X	X	L	L	L	L
hold (“do nothing”)	X	H	I	I	X	X	X	q_0	q_1	q_2	q_3
shift left	↑	H	h	I	X	I	X	q_1	q_2	q_3	L
	↑	H	h	I	X	h	X	q_1	q_2	q_3	H
shift right	↑	H	I	h	I	X	X	L	q_0	q_1	q_2
	↑	H	I	h	h	X	X	H	q_0	q_1	q_2
parallel load	↑	H	h	h	X	X	d_n	d_0	d_1	d_2	d_3

Notes

1. H = HIGH voltage level
   h = HIGH voltage level one set-up time prior to the LOW-to-HIGH CP transition
   L = LOW voltage level
   I = LOW voltage level one set-up time prior to the LOW-to-HIGH CP transition
   q,d = lower case letters indicate the state of the referenced input (or output) one set-up time prior to the LOW-to-HIGH CP transition
   X = don’t care
   ↑ = LOW-to-HIGH CP transition

The Fourth page

This contains the schematic diagram of the chip's interiors, which is meant only for the obsessively enthusiast. Normal people, who are currently only interested in **using** the chip, need to give a passing glance only to the transition diagram present here. It is presented with almost all Digital chips and you should have no problems in explaining the various rises and falls.

Also, they are the second best way of understanding what happens when the inputs change, especially if you are interested in knowing the time dependence of the behavior. An excellent place to resort to when **everything** else fails. ;)

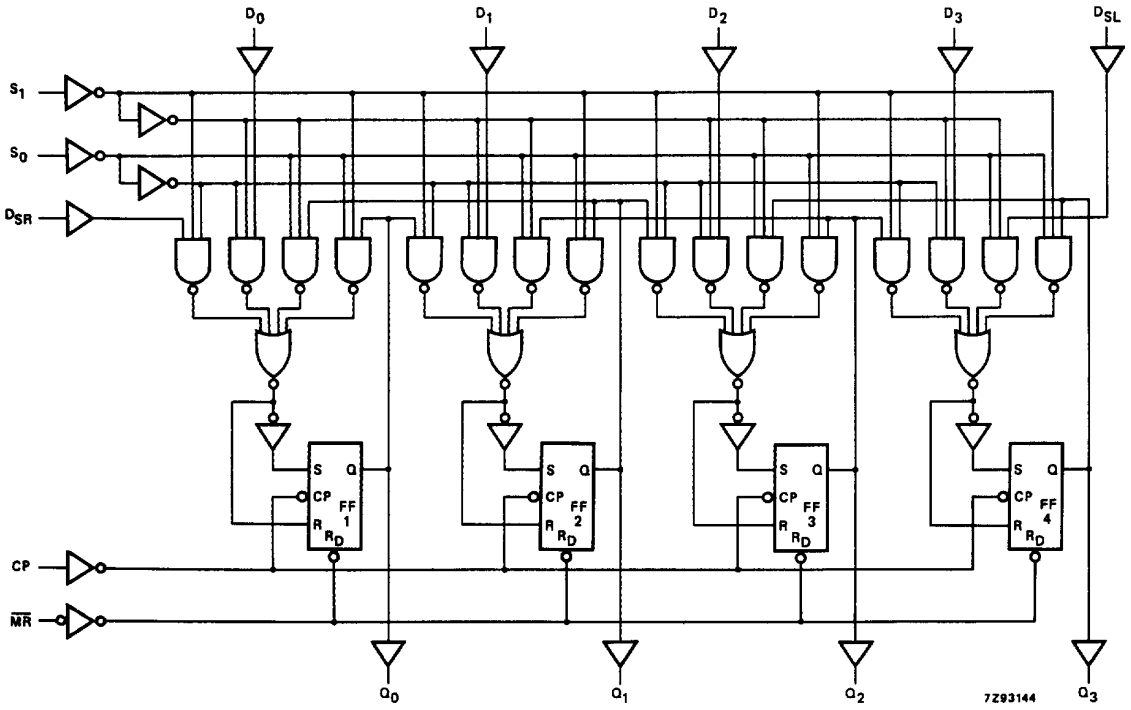


Fig.5 Logic diagram.

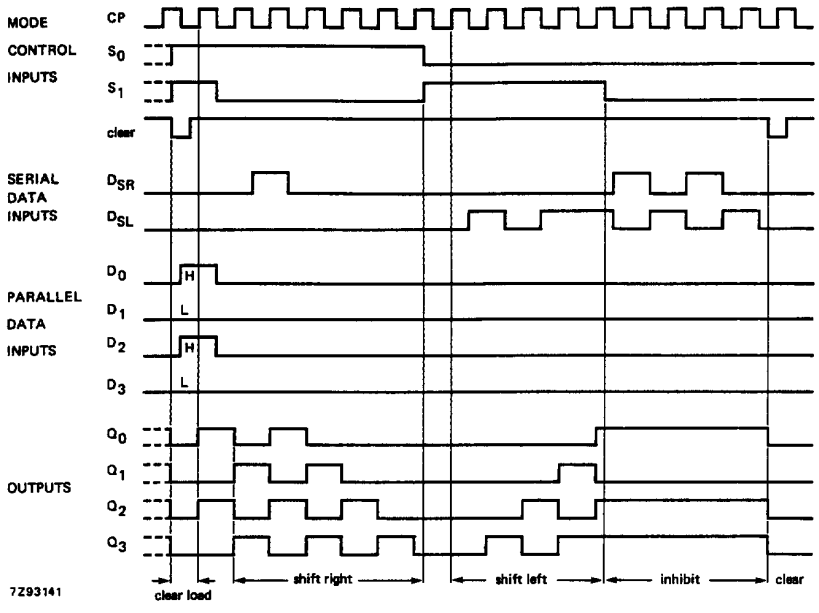


Fig.6 Typical clear, clear-load, shift-right, shift-left, inhibit and clear timing sequences.

The Fifth Page

Hmm... now the inevitable thing. The DC characteristics are present in some other datasheet. Too bad... However, there is an excellent news in the waiting.

When you are working with DC circuits, then the DC characteristics of our circuit will be generally in the typical ranges. Rejoice!!!

One more important thing, in case you do need to consult the DC characteristics, then you should design your circuit so are to keep the values as close to the typical values as far as possible. There will be a mention of maximum and minimum limits for most inputs, but then these will be the values where the chip ***fails***. So, beware.

The next page gives the AC characteristics. Most of the behavior parameters are of little use for the amateur designer, though to help you a little:

t_r := Rising time of the output
t_f := Falling time of the output
C_l := Load Capacitance

The giving any more detail will be boring. Then some datasheets make use of graphs too where more than one parameter may be varied to see results concisely.

DC CHARACTERISTICS FOR 74HC

For the DC characteristics see *"74HC/HCT/HCU/HCMOS Logic Family Specifications"*.

Output capability: standard
I_{CC} category: MSI

The rest of the data sheet contains the outputs for various AC inputs and outputs, which are generally neglected on the pretext of being irrelevant, the actual reason being the innate laziness. However, these details are critical for making robust circuits that take care of *ALL* input conditions. Though such a rigor is not expected from any novice, but if incorporated gives indications that the designer is a **designer**. Stay warned that you should carefully look at the chip model number before working on its characteristics. The manufacturer may be making more than one chip which does the same job, but using different methods or having different makes.

The physical specifications and the packing details will be irrelevant, unless you are planning to buy chips form the manufacturer and using them on specific circuits. Then you may need to choose one particular make of the chip to go with your PCB, etc.

One thing that this datasheet lacks, but which is present in most of datasheets, is the Applications section. LM555's datasheet has an excellent Applications section. One of the designs form this very part should be the starting blueprint of whatever you wish to do with the chip. The section is removed from this particular datasheet because they have included it in a more general one. (See the introductory page of the sheet.)

So that is almost all you should know about datasheets to start using them. One final last word, which is applicable to almost any document; ***read the fine print***. It helps. :)

Best of luck.

***Link*:** www.egr.msu.edu/classes/ece480/goodman/read_datasheet.pdf

***Link*:** http://cnx.org/content/m11857/latest/

Conclusion

There it is. The handout which should help you in moments when disaster is eminent, give you hope when you are on the verge of giving up, tantalize you when things are all clear before reading this, prevent dinosaur stampedes, avoid alien invasions, circumvent tsunamis, promote international disarmament and suffuse eternal happiness in all of mankind. We may be overstating a little, but this is the view that we had in mind when we began on this crusade against ignorance.

This handout should get you started on your circuits and will possibly mend a few leaks that might have remained after the gruelling lectures. Hope this really does make things easier for you, and even if we get a single positive feedback, we will consider our efforts successful. Kindly do keep telling us (and prodding us, if necessary ;-)) if you have ***anything*** that you think should be added to this handout. Also, pointing out glitches and errors in this text is also (...um... err....) encouraged. We will obviously incorporate the name of the contributors in the subsequent editions. ;)

So here, we guess, the ball is in your court. Make the best use of this, **read it so hard that the letters get *scrubbed* out.**

Best of luck.

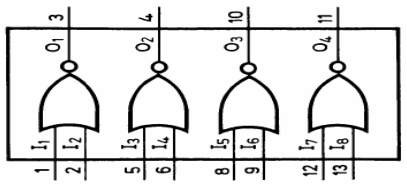
Digitally yours,

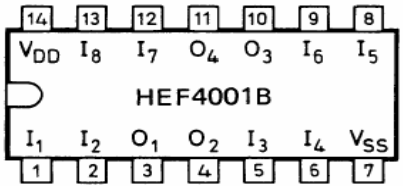
Us.

Appendix: Pin configurations of various ICs.

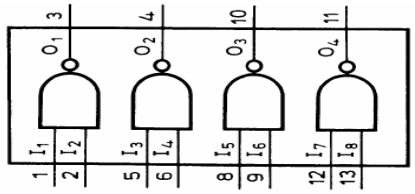
Gates

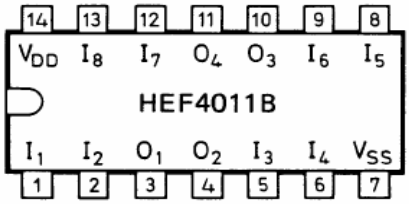
HEF 4001 B---quadruple two-input NOR gate



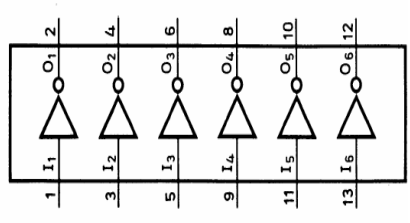


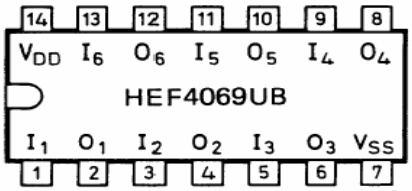
HEF 4011 B---quadruple two-input NAND gate



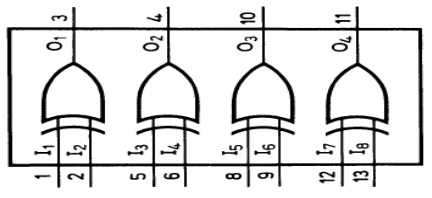


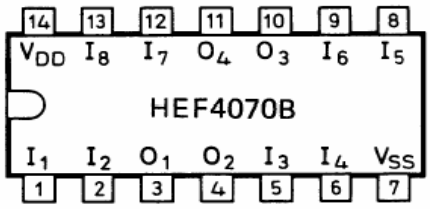
HEF 4069 UB---Hex inverter (NOT gate)



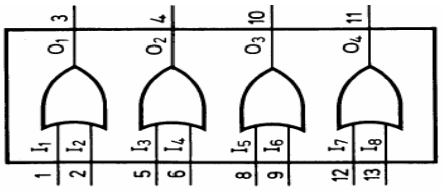


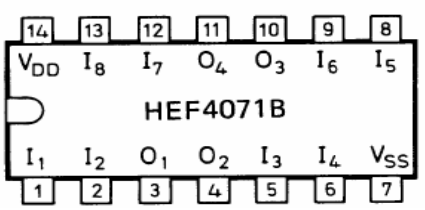
HEF 4070 B---quadruple XOR gate



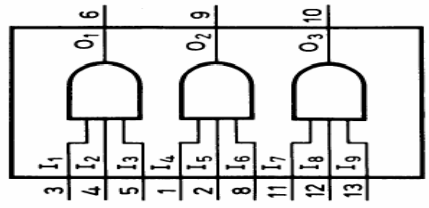


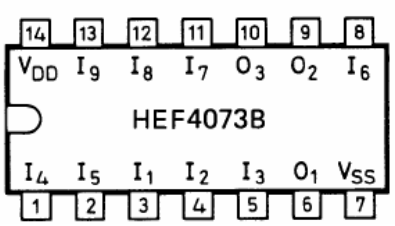
HEF 4071B---quadruple two-input OR gate



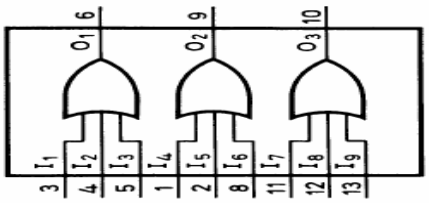


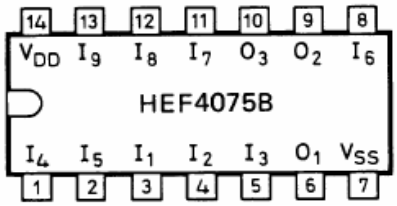
HEF 4073 B---triple three-input AND gate



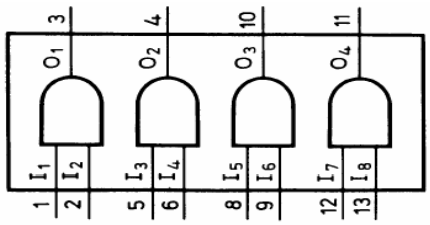


HEF 4075 B---triple three-input OR gate

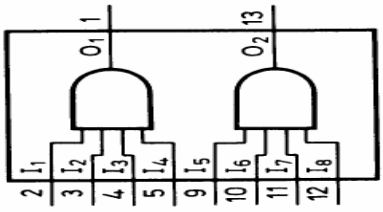




HEF 4081 B---quadruple two-input AND gate



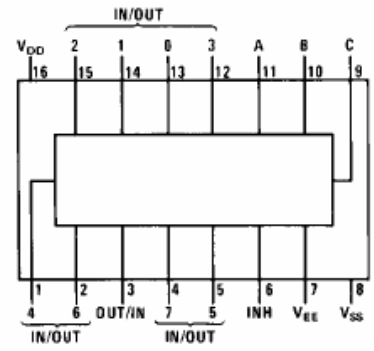
HEF 4082 B---double four-input AND gate

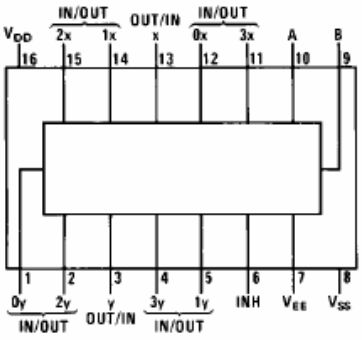


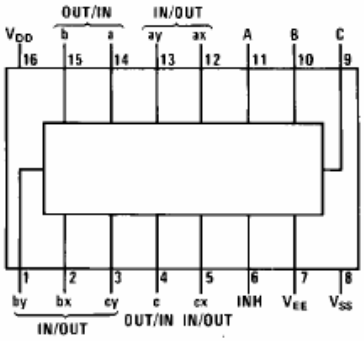
Multiplexers/ Demultiplexers

Same chip can be used as MUX and DEMUX both.

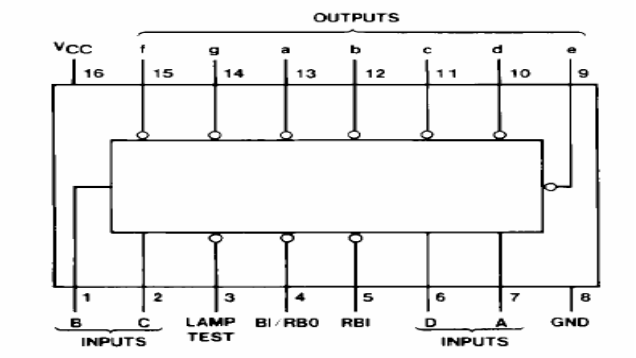
CD 4051 BC --- 8 x 1 Mux/Demux
CD 4052 BC --- 4 x 2 Mux/Demux
CD 4053 BC --- 2 x 1 Mux/Demux







DM 7447 A --- 7 segment BCD decoder



7 Segment BCD display