Introduction

So, the title of this chapter may seem somewhat intimidating and you're probably now thinking you need either a maths degree or a "brain the size of a planet" just to get started. The good news is you don't need either and if you can work your way through this section, you can probably grasp all the concepts in this series.

Electronics falls very broadly into two categories; Digital and Analogue.

Digital refers to circuits that deal with logic, computing, counting etc, where Analogue deals with circuits relating to audio, radio, transmitters etc.

Of course, the divide isn't as simple as that. A DAB (Digital Audio Broadcast) radio contains a microprocessor, but also must contain components that allow it to receive radio waves and audio components so you can hear the broadcast station.

This series is geared towards digital electronics though there will be a small amount of analogue usage.

So why start here

Binary, logic and logic gates and Boolean algebra are the basic building blocks of digital electronics. You find these concepts in just about everything electronic and they are concepts that you need to understand if you are to progress in digital electronics. Truth tables are something else you need to understand if you are to be in with any chance of comprehending many of the component supplier datasheets, or if you're wanting to define certain types of problems correctly for yourself.

You can't solve a problem if you can't define what it is.

These may sound like complex and complicated terms, but at their simplest they can all be explained with the help of a simple household table lamp.

Rules… is rules

You've probably heard the saying "there are always exceptions to the rule"... well there aren't... or if there are, then what was originally defined wasn't a rule, or it was wrong.

If you say "I always go shopping on a Saturday" then that means you always go shopping on a Saturday. You can't then turn around and say "Well, the exception to the rule is unless it's raining... then I don't".

That exception MUST be included of the original rule if it's still to be classed as a rule.

In other words, you must say "I always go shopping on a Saturday unless it's raining, or Christmas day, or my Gran's birthday, or I'm ill, or I have to work that day". The rule, to be a rule, must include everything.

Rules do not contain “usually” or “sometimes” or “most of the time”.

Another rule, "You can always drive through a junction if the traffic light shows green", isn’t a valid rule at all as it “may” be true, unless there is some selfish driver blocking your path whilst attempting to make an illegal U-turn.

So, the original definition of the rule "You can always drive through a junction if the traffic light shows green", was actually invalid and likely to get somebody seriously hurt or killed.

You could modify the rule to say "You can always drive thought a junction if the traffic light shows green AND your way is clear". As it happens, there are probably quite a few factors that would need to be considered when trying to define this rule so you don't end up in a car crash.

OR

Imagine what would happen if the driving through traffic lights rule was amended to say:
“You can always drive thought a junction if the traffic light shows green OR your way is clear".

What do you think would happen if your self-driving car obeyed that rule?

Well, as the car approached the junction, if the light was green it would happily drive through the junction, running over the little old lady trying to cross the road. However, if the car approached the junction and the way was clear (but irrespective as to what colour the traffic lights were showing) it would still drive through. The rule is valid if EITHER of the conditions are true.

Pure logic has a problem

Imagine a client has asked you to design a machine that turns a tap (facet for those of you in foreign lands) off whenever it sees the tap has been left on for more than 10 seconds but in the mean-time decides to employ a human attendant to perform this task until the machine is delivered.

A customer now walks into the bathroom, goes to sink, the attendant sees them turn on tap, wash their hands and walk away. Ten seconds later the attendant leans over and turns off the tap. Perfect.

The "conditions" of the rule have been met in that the attendant saw the tap turn on, left turned on for more than 10 seconds, and then performed the rules "action" and turned the tap off.

Did the attendant care that when the client walked away, or the tap was gushing water all over the place?

No, as the rule doesn't mention anything about how much water is flowing or if it's making a mess. As far as the rule is concerned, that tap has two states... it's either on or it's off, and how much it's turned on is completely irrelevant to the rule.

However… yes… but…

What happens if the customer is still stood there washing their hands after 10 seconds have elapsed? Should the attendant still turn off the tap after 10 seconds have elapsed?

The customer is going to get rather annoyed if the attendant keeps constantly leaning over and turning the tap off 10 seconds after they turn it on and start scrubbing away at their hands.

Rules is rules

But the rule... think of the rule... It appears that the management don't give two hoots about the customers. If the tap is on for more than 10 seconds you turn it off, but If they cared about the client or employed somebody to do some systems analysis they would have made sure that the rule catered for people still using the tap after 10 seconds.

They could have made the rule quite complex and said if it's gushing water all over the place for more than 2 seconds, turn the tap down to its 50% flow setting so it doesn't make a mess all over the counter top and floor, or they could have made the rule ONLY start once the customer has walked away from the tap.

There's also nothing in the rule about the attendant having to mop up excess water so that the next customer doesn't slip on the wet floor and break their neck. He was only following orders... or rules!

Now this is an important concept and one that any good computer programmer, mechanical or electronics engineer needs to understand.

Don’t blame the machine

Logic, is logical. Logic doesn't have intuition and it can't act outside its rules. Computer programs and electronic circuits will always follow the rules you specify (assuming there is nothing mechanically wrong with the machine obeying the rules). If the machine, gizmo or computer isn't following your rules, and there isn't anything mechanically wrong, then either the machine isn't being fed the correct information, or there’s a problem with your rules.

Interesting and sometimes extremely dangerous situations can arise if there isn't a rule to deal with a specific scenario, but that's not the machines fault. It's also only following orders.

For example, what happens if the tap has been on for more than 10 seconds, but no water is coming out.

Well, clearly there is a fault and a helpful attendant should report the fault to the janitor, but the attendant should still turn the tap off.

Seems sensible but your rules machine doesn't have any common sense... only rules to follow.

If you deal with the fault condition but forget to turn the tap off, when the water supply is restored the tap will start gushing water.

Now it's possible your 10 second rule will kick in and turn the tap off... maybe...

The rule does state "whenever it sees"... so what happens if it didn't "see" because the machine was switched off at the time the plumber was messing around, and they turned the tap on before switched the machine back on.

You, the programmer or engineer are responsible for adding some common-sense rules and fault condition detection to your machine, but don't expect the machine to exceed the sum of its programming or rules, and don't expect it to do you any favours.

We count in decimal

As humans we tend to count in decimal (also called base 10). We make use of the digits in the decimal system (0 to 9 - which is 10 digits) and using these digits we can represent any number we like, and to make life easier for humans who are interacting with computers and digital electronics, information is often displayed in decimal. The humble calculator for example.

Thank heavens the Romans didn't invent the digital calculator - can you imagine working in Roman numerals or a number system didn't have zero.

However, under the hood computers and digital circuits don't typically use decimal (there are exceptions so it's not a rule), instead they use a system called Binary (which is base 2).

Now might be a good time to ask Google to give you some basic info on number bases, and if you want to do well in the pop quiz, I'd ask it about base 2, base 10 and base 16 and if you really want to be a suck up, ask it about base 8.

Base 10, also called Decimal contains 10 digits; 0 to 9.

How many digits do you think base 2 (also known as binary) contains, and what would the digits be?

For an extra house point, how many digits do you think base 16 has, and what would the digits be?

Consider the decimal number 123

Each digit means something and in the decimal number 123 there are 3 units, 2 tens, and 1 hundred, or you could say there are 100 units (1 x 100), plus 20 units (2 x 10) plus 3 units (3 x 1) which equals 123 units.

Let’s now consider the binary number 1101, which we now know is base 2 so can only contain the digits 0 or 1. This binary number is NOT "one thousand, one hundred and one" as that's a decimal number, but one-one-zero-one; which doesn't exactly roll off the tongue, and as humans prefer to work with decimal numbers it's not by luck it's possible to convert between different number bases fairly easily.

The binary number 1101 is equivalent to the decimal number 13

Is your brain hurting yet?

With the help of Google, you should be in a position to now figure out how we get decimal 13 from the binary number 1101.

State of the lamp

Think of a simple table lamp. The lamp has two states as it's either on, or off. A more complicated lamp could be dimmable or it could keep changing brightness because a storm is blowing the outside power cabling around. However, irrespective of it's brightness, the lamp is either on, or off.

You're probably wondering where all this is leading. Well, the circuits within digital electronic devices and computers all work on the basis that everything has two states; something is either on, or it's off. Let's call this the "digital rule".

If you wanted to represent the state of something, our humble table lamp for instance using numbers, decimal would be a bad choice.

A single decimal digit can hold a value from 0 to 9, that's 10 possible values (the maths purists will be bouncing around that zero isn't actually a value well tough... we are sticking with it) ... or 10 possible states. But we only need 2 states, so binary is ideal and in fact, most computers work exclusively in binary; just 1's and 0's.

So, our humble table lamp with its two states can be on (there is a voltage applied to the bulb), or off, there is no voltage applied to the bulb, with the state of the table lamp usually being controlled by a switch of some type. The terms "on" and "off" are fine for table lights but it's not always technically accurate when used to describe digital circuits, so over the years better ways have been devised to help describe the two basic states.

Binary is a number system so uses numeric digits, and it makes sense to use 0 and 1. 0 is equivalent to off, and 1 being on. But for reasons that will hopefully become clear later on, a slightly better system is the "H" for High and "L" for Low.

It's all the same

Bottom line, Yes or No, On or Off, 1 or 0, High(H) or Low(L) it's all the same and you will see all of these systems used in documentation and datasheets. Throughout this series we will use the H and L system.

Now, if you were to draw a table that listed every possible state of the table lamp, it would only have two entries.

Light On or Light Off. Those are the only two possible states.

So, what if you have TWO table lamps (let's call them A and B)?

Lamp A	**Lamp B**
L	L
L	H
H	L
H	H

Remember that H and L, 1 and 0 or On and Off are all interchangeable terms.

In total there are four possible combinations of the two lamps, but each lamp still always follows the "digital rule" in that each individual lamp can still only either on or off.

if we had three lamps, A, B and C, how many possible combinations would there be? Draw a truth table to help you. Remember, it must cover every possible combination.

Think about this very real-world problem - an outdoor security light.

When the ambient light level falls below a certain threshold, we want the security light to switch on, and then, when the ambient light level exceeds a certain threshold, we want to the security light to turn off. This rule is found is most types of outside security lights.

To be able to perform this rule, the security light contains a daylight sensor whose output is on when it's daylight, and off when it's dark.

Remember previously that I said that rules have "conditions" and "actions", so in this example, the condition is the output of the daylight sensor, and the action is to turn the security light bulb on or off.

We can describe this rule with a truth table.

Daylight Sensor		**Bulb**
L		H
H		L

Pop quiz

Rule: "you need to wear a coat if it's raining and / or it's windy".

So, there are two conditions "raining", "windy" and one action, "wear coat".

a) how many possible permutations of "raining" and "windy" are there?

b) how many "states" does "wear coat" have?

The answer to "a" is there are four possible combinations:
It's neither raining nor windy, it's raining but not windy, it's windy but not raining, and finally it's rainy and windy.

The answer to "b" is easy. You either wear a coat, or you don't. You could carry the coat, put it in a backpack or throw it over your shoulder… but none of those are actually wearing the coat. You either wear it or you don't. So, there are two states.

A truth table for this logic question would look like this:

Raining	**Windy**		**Wear Coat**
L	L		L
L	H		H
H	L		H
H	H		H

Threshold

So, we're all hopefully in agreement that the table light (irrespective of its brightness) is either on or off.

But when exactly is a lamp, on? What is the definition of on and off when related to the table lamp?

A lamp could be so dim that you can't see it when viewing in sunlight, but that doesn't mean it's not switched on. What happens if you view the lamp in a completely dark room. Is the lamp said to be on only if you can see it? Maybe your eyes don't have great night vision and whilst you think the light is off, other people can actually see it glowing ever so slightly.

Here's something else to think about. If the bulb is blown of even removed from the lamp, is the lamp on when the switch is turned on, even though there is no light?

Do trees make a sound in the wind if nobody is listening (well, yes of course they do).

Clearly, when dealing with lamps there needs to be a scientific definition employed and a way that eliminates all confusion. This is especially true when dealing with electronics and logic.

Fortunately, electronics employ a term call threshold. If a voltage is at or above a certain level or threshold, it’s deemed to be on, but if it’s below that threshold it’s deemed to be off.

The manufacturer of the electronic component sets its logic threshold voltage and as long as we take this into consideration everything should be fine. To make things simpler there are standards for the threshold voltages and most of the manufacturers follow these standards… usually.

So, our “digital rule” is even better defined now. At or above a certain voltage level, a logic signal is said to be HIGH and below this level, is said to be LOW.

Let’s get practical

Right, enough theory let’s do something practical.

We are going to design and build a circuit using a logic gate to solve our weather-related logic question above.

As a reminder, the rule is: "you need to wear a coat if it's raining and / or it's windy".

To simulate the weather there will have two push buttons, one labelled "raining" and one labelled "windy" and an output LED labelled "wear your coat" will illuminate if you need to wear a coat.

Now as it happens you don’t really need a logic gate to implement this rule as it's really simple and could just be done with the switches, bits of wire and the LED… but where’s the fun in that.

What are these logic gates and where do I find them?

The logic gates you will probably use the most are contained within integrated circuits (ICs). Don't worry too much about how it's done, only that they are tiny and so to make them easier to use the manufacturers encase them in plastic or ceramic, put little metal legs on so you can make electrical contact with the insides, and construct them to defy the laws of physics so as to nearly always land legs up in the air ready for you to stand on them in your bear feet! They are usually always black (engineers have no artistic taste), available in standard package sizes and so the only way you can tell one type of IC from another is to look at its number that's been stamped on the IC body in ink (sometimes the markings are laser etched into the plastic) that fades over time so you can hardly read it.

The building blocks of logic

You may be surprised to know that there are only a few different types of logic gate.

The four blocks of logic (sometimes called logic elements) are:

- **AND**
- **OR**
- **NOT**
- **XOR**

Those of you who use programming languages may recognise this list. Some programming languages may use symbols or other words and terms but these four elements are usually available in one form or another in programming languages.

Even though there are only a handful of base logic element types, they can be combined together in an infinite number of ways creating thousands if not millions of different types of IC.

There are some additional types of logic elements that you will come across including:

- **NAND**
- **NOR**
- **BUFFER**
- **LATCH**

but more on these later.

The above picture is of an IC that is quite common.

It's a 14-pin package in the "DIL" style - (Dual In-Line because it has two rows of pins). This IC is made by Texas Instruments (notice the tiny little logo map of Texas at the start of the first line of text). You can ignore the rest of this line as it's just manufacturing and production date information.

The little horseshoe cut out on the left is used for package orientation. With the cut-out on the left, the bottom left-hand pin is pin 1.

DO NOT ASSUME that the printing is a clue to the ICs correct orientation. If this is wrong it can be an indication that the IC is a fake or a reject! Always go by the cut-out or sometimes a little dot just above pin 1.

The second line contains the information we need to know.

- SN - This is another indicator to the fact that it was produces by Texas Instruments.
- 74 - This is the series or range number. There are hundreds of ICs in the 74 range. No matter which manufacturer, a 74 part from one will always be compatible with a 74 party from another.
- HC - This code describes the operating properties of this IC – more on this later.
- 32 - This is the part we're really interested in. You can ignore any alpha characters after this number

Ok, so what is it

The 7432 IC contains four (hence the Quad in the name), identical, 2-input logic OR gates.

If you want to see the datasheet for this IC, just type "74HC32 datasheet pdf" into Google.

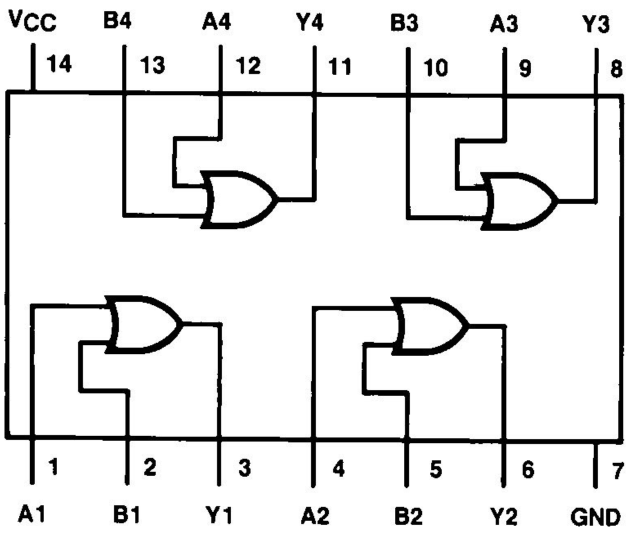
We've looked at a few simple truth tables so you should be starting to get the general idea of what they are and how they work by now.

Below is a partially completed truth table for a 2-input OR gate. Complete the table.

A	B		OUTPUT
L	L		
H	L		H

Hopefully you've not had too much difficulty with the above. For an extra credit, draw a truth table for a 2-input AND gate? (Yes, I know, we've not looked at AND logic gates yet, but use your common sense).

The datasheet for the 7432 IC contains lots of technical information that to be frank will give most people a headache, but the section we really need to understand is the description of how the logic gates are wired up inside the IC package, which is usually presented as a nice diagram.



Notice pins 7 and 14. All logic ICs need to have power to work and it's connected to these pins. Vcc (sometimes called positive, +, +ve) is connected to pin 14, and the GND (sometimes called 0v, - or Vss) connection to pin 7. The IC pictured has 14 pins, and 14, 16, 18, and 20 pin DIL packages are very common. Also, almost all of the 74 series ICs use the top left pin for Vcc, and the bottom right pin for GND... ALMOST!!!

ALWAYS... ALWAYS... CHECK THE DATASHEET.

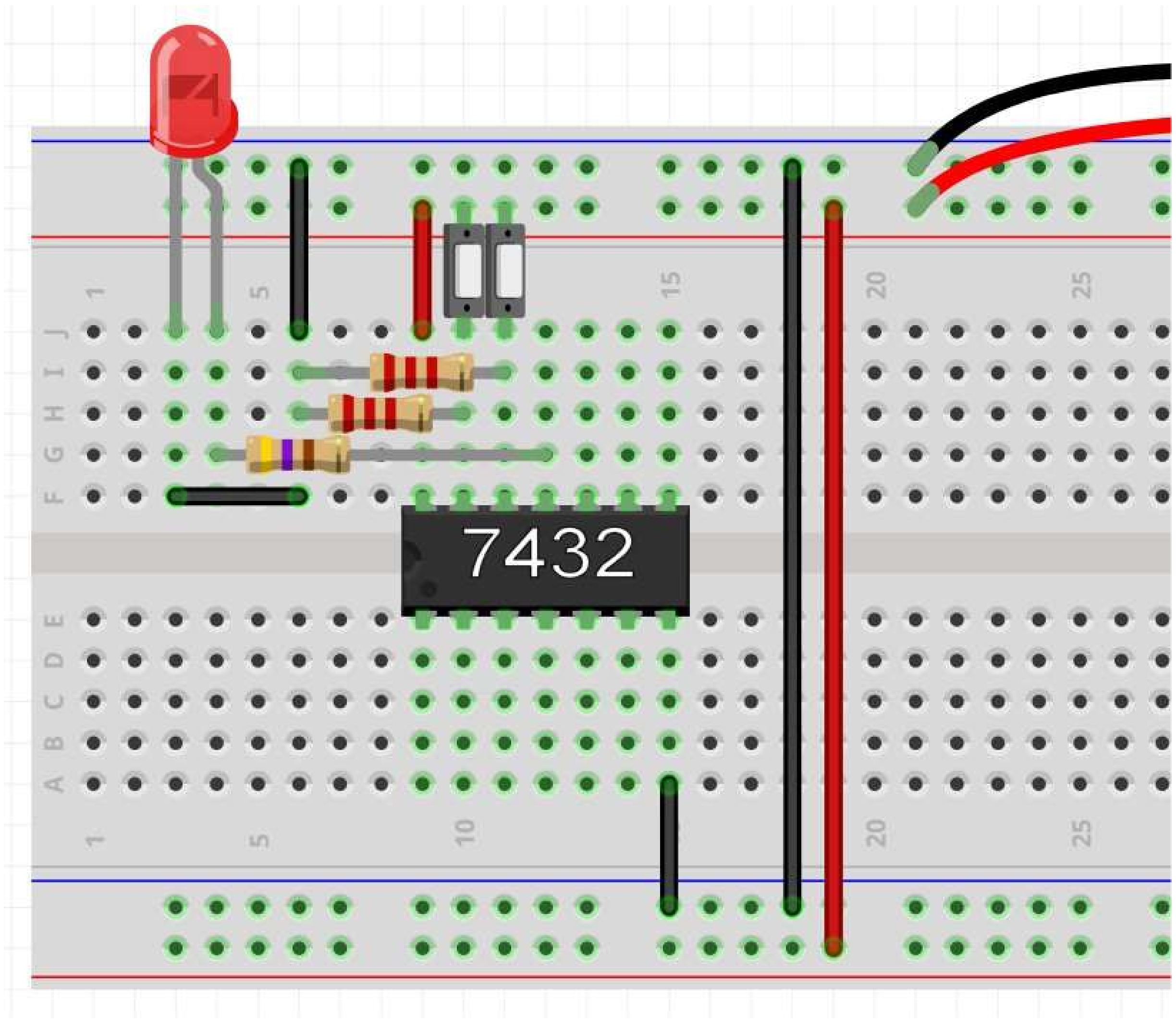
Some IC's don't have their power connections in the usual place and some have the logic gates back-to-front. Most of the time you will just glace at the internal diagram on the data sheet and that's all you need to know.

Looking at the above IC internal diagram, hopefully it doesn't take a rocket scientist to realise that any pin labelled A or B is an input, and any Y is the gate output.

Why use A, B and Y

Well, using numbers could be confused with the IC's pin numbers. And typically, letters at the start of the alphabet are used to indicate inputs, and letters at the end of the alphabet are used to indicate outputs or have other special meanings. Usually...

Time to build something



This circuit will allow you to apply logic levels (high or low) to each of the two OR gate inputs (pins 12 and 13) and monitor the gate output (pin 11) on the red LED. Pressing the white push buttons applies a logic high to the associated gate input, and releasing the button applies a logic low.

Floating is bad

As we know, logic is based on the concept of state... and the state is either on or off, or high and low etc.

However, it's important when dealing with logic gates and circuits that the gate inputs the required threshold voltages specified by the manufacturer. Gate inputs that are not connected to anything are said to be floating and floating inputs nearly always end in trouble. Floating inputs can cause random behaviour in circuits and all sorts of strange behaviour.

Looking at the circuit on the breadboard, when a push button is pressed, +ve flows through the switch and into the corresponding gate input. But when it's released without the resistor the gate input would start to float as it wouldn't be connected to anything. The resistor gently "pulls" the input towards 0v whilst preventing a short circuit from occurring when you press the button. You can use almost any resistor value but don't go lower than 1K else you can start drawing a lot of current when you press the switch. Anything from 2.2K to 10K is usually ideal.

Testing the circuit

After you've assembled the circuit do a final quick check to make sure it looks correct and then connect to a suitable power source set to around 5v and switch on.

DO NOT EXCEED 6v to the IC

You may get away with a higher voltage for a little while but eventually the IC will be destroyed. Some IC ranges have different voltage capabilities so as you should now be accustomed to me saying, CHECK THE DATASHEET.

With no buttons pressed, the two 2.2K resistors are pulling the two gate inputs low, and the truth table states that with two low inputs the output of an OR gate should also be low, so no LED illumination. Pressing either or both of the buttons should illuminate the LED.

It should take you around 30 seconds to get bored with the above circuit - it's not exactly mind-bending stuff.

Let's make a little circuit change.

- Switch off the power and rotate the LED around 180 degrees thus connecting backwards.
- Remove the black wire link between point F3 and F6 and insert a red wire from F3 directly down to the red (positive) bus rail on the lower strip.

Switch on the power, press the buttons and now see what happens.

If you've done the changes correctly, the LED should have come on as soon as the power was switched on and pressing either or both buttons should turn OFF the LED. But why?

Draw a truth table to show what's now happening.

Well here's a clue. Remember there are only two states, High or Low, so the gate output MUST ALWAYS be one of them (ok… it must usually be one of them but there are some special versions of logic gates that can work slightly differently but more on these later).

Now a word of warning about inputs, outputs and loads.

The 74 IC outputs are not capable of powering much more than one or two LED's. Do NOT try connecting bulbs, motors, relays or more than two LEDs to a gate output else you will almost certainly damage the gate. Usually the remaining gates in the IC will be ok but not always.

Many inputs… but only one output

You can connect as many gate inputs together as you like and there are lots of valid reasons for doing this. You can connect ONE gate output to one or more gate inputs, however, NEVER connect multiple gate outputs together. There is a variant of the gate called "Open Collector" for when you need to connect multiple outputs together but DO NOT do this with regular gate outputs.

You've previously worked out the truth table for an AND gate and A 7408 IC is a quad 2-input AND gate.

Replace the 7432 IC with a 7408 IC and check if the AND gate truth table you worked out previously is correct. Do NOT forget to find a 7408 datasheet and check the pinouts.

At this point you've experimented with a 74032 OR and 7408 AND gates. Now try using 7400, 7402 and 7486 ICs. Again, remember to read the data sheet for each IC before you use it to determine what the pin connections are.

You may find it helpful to write out the truth tables for each of the different logic gates.

As a bonus exercise repeat the experiment but use a 7404 IC. I'm warning you… check the data sheet

The end… for now

Ok we've covered a lot in this article. We've touched on binary; though I've not really told you why… yet… and I've left you to go off and research it more on your own. We've looked at reading and drawing truth tables, hopefully grasped the concept of state and experimented with a few basic logic gates. You've also hopefully read quite a few datasheets now.

We will be building on these fundamentals in the next thrilling instalment.