

General Notes on Construction, Troubleshooting & Schematics

Stephen Petersen

I. Philosophical Notes.

Successfully translating engineering designs from theory to working breadboard circuits involves more than just theoretical understanding. Empirical skills, knowledge gained through experience and judgement following from the application of theoretical knowledge, are equally important. As computer or electrical engineers, our theoretical skills usually dominate, especially at the early stages when we are learning engineering basics. This is when we are all too often perplexed or even fearful of the equipment, circuits, components and materials needed to do our design work effectively. This happens mostly because we don't know what theory to apply where to account for things in the messy experimental world in an understandable and controllable way. We can express the relationship between theory and experience in a philosophically more precise and insightful way.

This philosophical point is really quite obvious when you reflect on it, but is often ignored in practice at least some of the time by nearly all of us: the real circuit is not the one drawn on your piece of paper or pictured in your own mind. At best, these are only symbolic abstract representations which omit much. For example, both will probably assume the existence of a perfect, noiseless power supply with constant voltage and infinite current capability. All interconnecting wires will have zero resistance and exhibit no electromagnetic properties like radiation or propagation delay. Neither will there be any inductive or capacitive effects causing filtering problems. Grounds, or any wire or component connected to this common node, are conveniently symbolized as a single electrical point always with a constant potential of zero volts (whether drawn in many places all over the schematic or not). Switches are perfect and everything lasts forever, etc.

Like territory symbolized by an intelligently drawn map, an engineering schematic can never include everything about the circuit it purports to represent; we rely on experience and an understanding of what is relatively significant to account for that. The map is certainly real, and so is the territory. But we don't say the map and territory are the same thing anymore than we say the circuit and its associated schematic are the same. What insights can this seemingly self-evident observation yield?

Properly, the schematic exhibits a reality which is logically consistent with the theory responsible for creating it, but merely represents a model of something we have somehow gotten nature to also do for us embodied in an actual physical circuit. Which then is the real circuit, the one symbolized or the one constructed? Since the former is always a

model and the latter an extant thing, we gain insight by recognizing that when we think about how a circuit works, that *what is actually going on may differ in significant ways often entirely unknown to us*. In the digital laboratory you design digital circuits that predict digital behavior of eventually constructed digital circuits. Actual digital circuits though are really complex electronic analog circuits that have been engineered to mimic digital logic.

II. General Discussion.

1. Wiring Diagrams, Schematics and Engineering Schematics

A schematic is simply a mechanical drawing showing the interconnection of all devices, components and sub-systems that implement an engineering design. Engineering schematics are mechanical drawings meant to convey not only required interconnections and components, but also as much of the engineering design as possible; they are meant to be intelligible, *i.e.*, read and used by engineers (note that schematics and engineering schematics are often used interchangeably to mean the same thing). This type is particularly useful in prototype and experimental work because they are easy to troubleshoot from. Wiring diagrams show how components are connected on a circuit board or other particular construction format, but convey little if any engineering content.

Students invariably confuse schematics (both types) and wiring diagrams. While there is a one-to-one relation between them, one allows you to make technical deductions about a circuit while the other's sole value is to verify point-to-point physical interconnections. For some strange reason, students almost always want to draw wiring diagrams and use them exclusively. As you will see, this makes troubleshooting difficult or impossible.

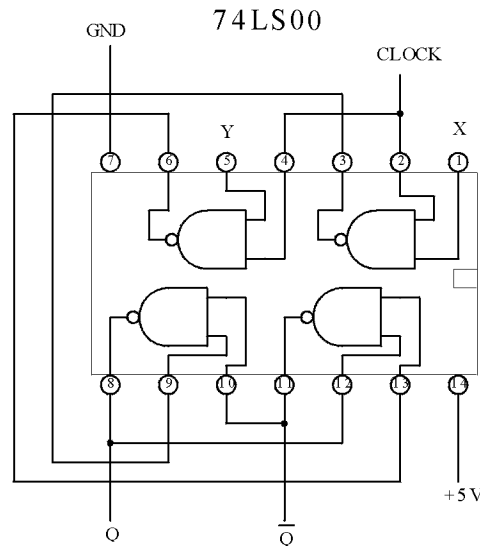


Fig.1. Wiring diagram.

Fig. 1 shows an example of a simple wiring diagram as a student actually wired it on a digital breadboard. Except for the interconnections, no other information is conveyed except what you might conclude from the signal names. Fig. 2 shows this same circuit as an engineering schematic which reflects the way the chip was actually wired

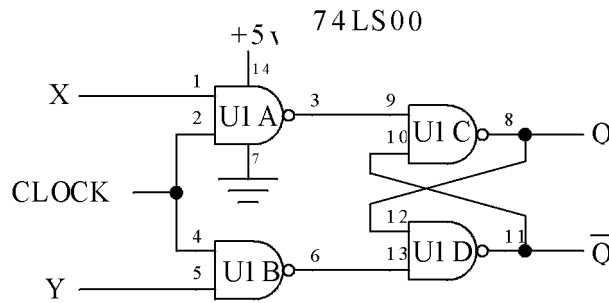


Fig.2. Engineering schematic.

Considerably more information is evident, such as its function as a memory block. Note the pin numbers convey the same information as the wiring diagram does and should be favored for troubleshooting over the wiring diagram.

Guidelines have developed over the years for drawing engineering schematics. Unless you have a good reason for deviating from them, the following should be adhered to:

- Signal flow should move from left to right across the page with inputs on the left and outputs on the right.
- Use the “unit number” convention for assigning a unique package identifier to each chip. For example, the 74LS00 in fig. 2 is a single chip containing four Nand gates, so the chip is assigned identifier U1 with its internal gates identified by letter suffixes: U1A, U1B ... etc. Variations of this convention exist in industry. Only one of the common gates need show the power connections (pins 7 and 14). This is often omitted but you should include it as a reminder as well as to make your schematic complete.
- Electric potentials (voltages) should increase as you move from the bottom to the top of a page. For example, fig.2 shows the +5V supply upwards while the ground pin is downwards (where “ground” really is).
- When using block diagrams to show a component, the same left to right signal flow and potential orientation should be adhered to. Additionally, every signal should be labeled *inside* the block with pin numbers (if appropriate) on the outside. Students often draw block diagrams without labeling the associated interconnecting wires. Sometimes this is evident from the context, but this should not be relied on. Examples are MSI, LSI and VLSI parts, such as counters and CPUs.

Always begin and work with a complete and readable engineering schematic that accurately reflects what *has actually been built*. This allows you to reason *from the schematic* while troubleshooting, taking data or making modifications as you go along. Wiring diagrams (fig. 1) can also be quite useful, but they should never substitute for a good engineering schematic.

Annotate your schematic as you build the circuit. If you don't do this, it will be nearly impossible to troubleshoot after it inevitably fails to work. For example, this would naturally include pin numbers to accurately document how devices are connected.

2. Construction.

Build circuits in a modular fashion, wiring power and grounds first. Look carefully at what you need to build and, if possible, break it into convenient modules that can be built and tested as distinct units. As each module is tested, it can be wired to other already functioning parts; troubleshoot as you progress. This will save you from wondering where to start when the circuit fails to work after completion, and will provide confidence as you go along since portions of it are already working.

3. Troubleshooting.

Learning to troubleshoot can only be done by doing, since it is really both an art and a science. All troubleshooting and reasoning about a circuit's wrong behavior should be done by reference to its experimental engineering schematic. Even so, you should always keep in mind that the schematic is *not* the actual circuit. That is always more complex than the schematic can convey. Learning to reason from an experimental engineering schematic is an invaluable skill. When incomplete or poorly annotated, they make it much more difficult to troubleshoot and modify what has been built.

You should also be aware that besides the schematics in your engineering notes, your own conception of things - that picture of what is going on in your own mind (itself an abstract model) is *always provisional*. More often than not, when seemingly insolvable problems are encountered, you will turn out to be your own worst enemy!

Troubleshooting is essentially an ordered and deductive procedure with a paradoxically strong inductive component. You must always start from a known point and proceed in a cause and effect manner to find the source of some wrong functioning. Ask yourself what it is doing right and what it is doing wrong. Always solve *one* problem at a time. Employ inductive reasoning when you consider what could be the cause of such-and-such an observation. For example, a circuit producing an output isn't necessarily bad because it's wrong; it might be what's connected to it and loading it down. As each malfunction is repaired, others will often mysteriously disappear. Avoid the trap of randomly playing with a circuit hoping to discover the cause accidentally. This is warranted to get an initial feel for what is going on (and then you are only cataloging symptoms – not investigating causes), but will always lead to frustration and a strange conviction that spirits are stubbornly holding your circuit hostage. With experience, you will gain insights into rapidly zeroing in on many common problems. Some helpful tips tailored for digital circuits follow:

- Look for obvious things first: power (Vcc) , grounds, power switch, AC plug (very embarrassing), chips not properly inserted into the circuit board. Shorting Vcc and ground is a common problem that is easily fixed.
- Judiciously use whatever led's are available to help you find problems. Sometimes a partially working sub-circuit can be used as a troubleshooting aid.
- When checking for a signal at a chip's pin, look *at the pin itself* and not the breadboard. Otherwise you assume the breadboard is making good connection to the chip. Check both the pin and the breadboard.
- If an output is wrong, be sure and confirm that the problem is not really being caused by *something else connected to the output*. You should suspect this sort of thing whenever an output seems to remain either high or low and can never be made to change. To find the problem, disconnect all loads from it, confirm the output is working, then one by one re-connect the loads until the problem is duplicated.
- Chips seldom go bad unless you deliberately do something wrong, like reverse power and ground connections.
- Learn to distinguish between electrical and logical problems. The former involve things like wrong supply voltage (check with voltmeter), or power or ground pins not connected, or logic outputs connected together. Logic problems are due to errors in design, like a latch that toggles on the wrong clock phase.
- Some problems occur with digital devices so often they are worth commenting on here. CMOS and most FET-based devices (NMOS, PMOS etc) are usually very sensitive to improperly terminated input pins that have not been tied to “solid” logic levels (typically “pulled down” to ground or “pulled up” to Vcc). This is the classic *floating input problem*. CMOS inputs that float to a middle value between logic 0 and 1 will cause significant current to flow that will increase power dissipation and often strange (*really strange*) intermittent behavior. A good way to find these is to touch pins with your finger (don't worry, the voltage is low and your body has a relatively high resistance). This will mimic a poor termination and cause at least different behavior. Inputs properly terminated (or connected to some output) won't be affected by touching.

Finally, avoid the beginner's predilection to tear the whole thing apart and start over, hoping it will miraculously work next time. One of the side benefits of troubleshooting is the truly deeper understanding you will gain by finding and confirming *by theory* what caused the problem. Congratulations! You will then have closed the time-honored engineering design loop: design and experimental confirmation of theory.