THE PROPER CHOICE OF WORDS LEADS TO MORE EFFECTIVE TROUBLESHOOTING.

The linguistics of electronics troubleshooting

Engineers frequently underestimate the time that electronics troubleshooting requires and often completely leave this step out of hardware-development schedules. Yet projects often hinge on getting recalcitrant electronics working correctly under an impending deadline. Making matters worse, engineers rarely learn troubleshooting skills in school.

The useful troubleshooting techniques that follow are the result of 20 years of electrical-engineering projects in the automotive, biomedical, instrumentation, and telecommunications areas. Linguistics is an important aspect of all of these techniques. Careful study of these techniques can significantly tighten the learning curve for electronics troubleshooting.

The *Merriam-Webster Collegiate Dictionary* defines linguistics as "the study of human speech including the units, nature, structure, and modification of language." Linguistics guides our thoughts and what we assume to be true at the moment. If you say, "I know it's a software problem," then you're probably precluding the search for a hardware problem.

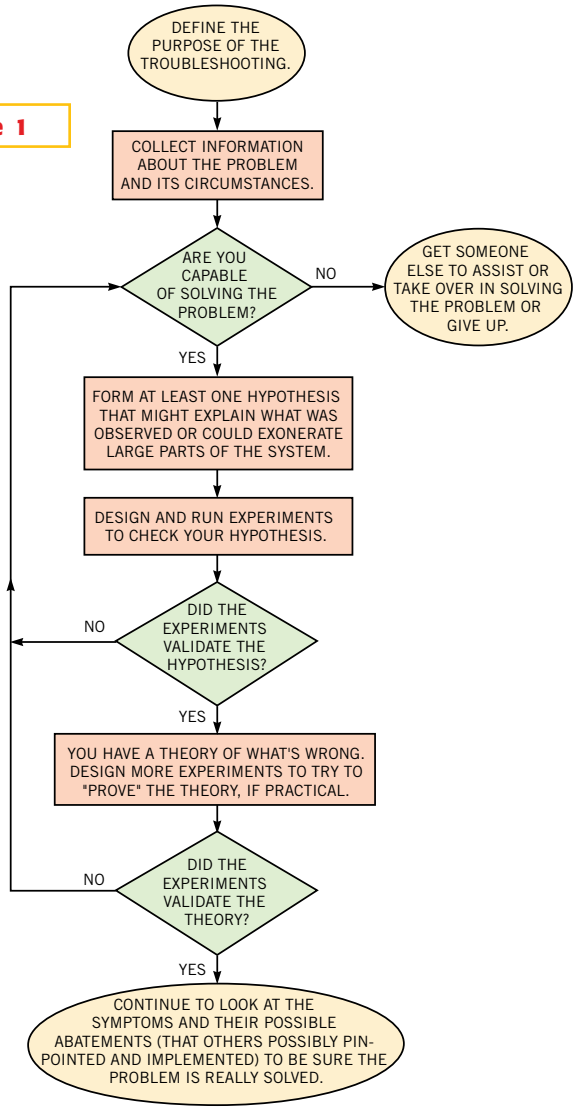
The first three techniques are essential troubleshooting methods that use linguistics to form and engage useful questions, hypotheses, and theories. The next three techniques focus on using linguistic interactions to build rapport, call for outside help, and manage perceptions.

1. USE THE SCIENTIFIC METHOD

The terms "hypothesis" and "theory" distinguish guesses from what is closer to truth. A flow chart for the scientific method, adapted for troubleshooting (**Figure 1**), includes the following steps:

1. Define the purpose of the troubleshooting. Obviously, you want to fix the circuit or the problem. If the system is small and you designed it to begin with, then fixing the problem is probably your goal. Otherwise, your goal might be to eliminate every problem from all areas except one, such as firmware, and then assign that area to someone else to troubleshoot. If many people are involved, documenting the process of confining the problem can be useful. This documentation may allow you to relate the results of the data rather than appear to offer judgment about someone else's contribution. Generally, you troubleshoot problems one at a time,

Figure 1



The scientific method for troubleshooting uses linguistics to form useful questions, hypotheses, and theories.

even if more than one problem exists.

2. Collect information about the problem and its circumstances. This step is useful whether you're trying to fix a problem on your own board or someone else's. Instead of swapping out parts and boards, try to observe the symptoms of the problem—or the smoky remains—while the problem is happening. Be patient and collect the information systematically. At this stage, try to avoid verbal judgments about what's really going on.

3. Ask yourself whether you're capable of solving the problem. Electronics systems can be complex. After collecting only a little information, you may realize that you probably can't solve the problem. Or, you may find that you can solve the problem only if it's in one small domain, in which case it might make sense to find out whether the problem is in your small domain. Otherwise, it may be time to pass the problem along to someone else, or at least include others in the troubleshooting process.

4. Form at least one hypothesis that might explain what you observed or that could exonerate large parts of the system. It is often useful to come up with several hypotheses that you can investigate. Ask what your observations could be true of. Remember that hypotheses are not statements of fact. Rather, they are ideas that you may choose to investigate. Suppose you are troubleshooting a problem in which a green LED marked Ready is not turning on when you turn on an instrument. You could say, "The LED indicator most likely broke when we plugged the power in backward." However, a more useful statement is "One hypothesis is that the LED indicator itself might have broken when we plugged the power in backward." Another hypothesis is that software is just not issuing the command to turn on the LED, which could be true if the software is not in the Ready state.

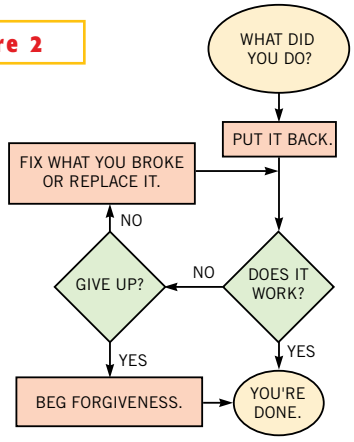
5. Design and run experiments to check your hypotheses. Say or write what you expect to see to validate or invalidate each hypothesis. Otherwise, the results might be ambiguous. Prioritize the experiments and start running them. Obviously, you may have to delay destructive tests. You may choose to first run likely hypotheses with simple go/no-go tests, because such tests yield clear results. For example, if you're measuring valve clearances on an automobile engine, the 30-thousandths gage may go in easily, and the 33-thousandths gage may not go in at all.

Continue to refer to your hypotheses as just that. It is sometimes useful to state assertions to check hypotheses in predicate logic terms. For example, you might state, "If the Ready indicator turns on, and the board is out of reset, then we might claim that the interrupts are in their enabled states. Yet it doesn't appear that software has actually enabled the interrupts. How can we verify the situation?"

It can be useful to develop experiments using a binary search technique—successively dividing the possible domain of the problem in half—that allows you to quickly isolate the problem.

6. Once a hypothesis appears to have some validation, design more experiments to prove or disprove the hypothesis. If the experiments hold up, you'll have a theory about what happened. Many times, the problem and resolution are clear once you identify them. For instance, you might find that the reason the Ready indicator isn't coming on is because a green LED is missing on the board. Once you stuff the LED, the Ready indicator comes on as expected upon power up, and the problem is solved.

Figure 2



The put-it-back troubleshooting technique is simple: Reverse what you did prior to the appearance of the problem.

7. Continue to look at the symptoms and their abatement in certain conditions. After sufficient experience, you may start referring to the problem's symptoms and resolution more as facts than as tentative hypotheses or theories. Continuing the above example, suppose you find that the Ready indicator still doesn't come on appropriately on your circuit boards, and these boards' green LEDs are in place. In this case, you may be stuck at a local minima, but you haven't resolved the problem. Go back to Step 3.

THE SCIENTIFIC METHOD IN PRACTICE

I once spent three months troubleshooting a problem that started off with a fairly straightforward symptom. A prototype pulse oximeter had occasional voltage glitches on the output of the preamp stage, which turns the light received by a photodiode into a proportional voltage. The targets of my hypotheses ran the gamut from power-supply noise, poor preamp layout or design, a bad op amp, poor wiring, and a bad cable configuration, to noisy light sources, which create the light for detection by a photodiode. Several of these hypotheses successively graduated to theories before additional evidence trashed them. The problem seemed to center around a sequence of gain changes from high to low.

Finally, when I had investigated everything else, I formed the hypothesis that the photodiode itself was the source of the problem. I couldn't understand why, however, because the photodiode's response-time rating was in dozens of nanoseconds. The breakthrough came when I found that certain photodiodes didn't seem to evoke the problem, and others did. Further investigation revealed a secondary 10 to 0.01% turn-off tail of silicon photodiodes on the order of hundreds of microseconds. This tail is not a normal characteristic; understanding it, with the help of some device-physics books, solved the problem. The point of this digression is that careful use of the scientific method can keep you on track to solve a previously unrecognized problem in many designs. I started out thinking I had a problem on one prototype pulse oximeter and ended up discovering a whole new design issue and even patenting the solution (US 6,018,674).

ASK RHETORICAL QUESTIONS

Here are a few "standard" rhetorical questions that may be useful if you're stuck while forming your hypotheses:

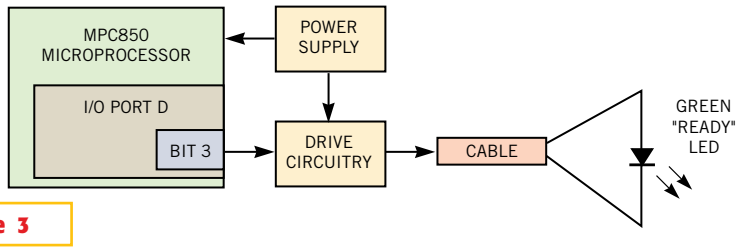


Figure 3

Drawing and studying the block diagram, in this case for a problem associated with a green LED, facilitates forming hypotheses and designing experiments.

- Are the power-supply voltages at their nominal values? Where is ground? These are usually the first questions to ask about a board. Hundreds of symptoms may disappear once the power-supply voltages are at the right levels with low noise and ripple. Check the voltages on the actual pins of the ICs that are using the various voltages. A power-supply voltage that is within specification but not at the nominal value, such as at 4.8V when the spec is 5V±5%, can indicate a salient problem.
- Is the board out of reset or cycling in and out of reset? Why? Of course, several levels of reset may exist.
- Are the PROMs (EEPROMs, EPROMs, and flash) programmed? Are the relevant board parts stuffed in their sockets with Pin 1 correctly oriented?
- Is the board from a system that has been in production for years? If so, what changed, or has the problem always existed?
- Is the problem intermittent or continuous? Under what conditions does it occur?
- Have I examined the circuit board in question? Are there obvious solder shorts or opens? Do I need to look at it under a microscope?
- Has the system been correctly initialized? Have the power supplies been correctly sequenced to avoid problems such as CMOS latchup? Is there an artificial way to bring up the system to make the problem disappear or to eliminate possibly separate system start-up issues?
- Am I operating the board in a system that supplies it with all of its typical parameters? For instance, a board with an analog sensor may rail out if the sensor isn't plugged in or if the sensor isn't in its typical operating situation.
- Have I been following standard ESD precautions?
- Are the power-supply voltages still OK?
- What do the clocks look like? Are all of the clocks running at the correct frequencies and duty cycles? Are the clocks appropriately synchronized, phase-locked, and within specification?
- Are all of the frame syncs lined up relative to each other and the clocks?
- Does unacceptable jitter exist across clock domains?
- If the problem is in programmable logic, does it exist in simulation? What assumptions did I build into the simulation that may not hold? Did I simulate the system around the logic design?
- For programmable logic, am I sure that I understand what the place-and-route software actually did?
- For sensitive analog circuitry, is noise coming in at the front end around the preamp or most sensitive analog sensor? How do I know when the output is quiet enough? What happens if I turn off all of the possible noise sources and run the analog section off of a bench supply?
- Is the feedback in the circuit correctly functioning? Is there unwanted feedback?
- Do all of the data streams, such as E1, T1, ATM, and Ethernet, meet the applicable physical and software standards?

2. PUT IT BACK

Another troubleshooting technique is simple: Reverse something you or someone else may have done prior to troubleshooting. For example, if you took a part out and there's a problem, put it back and see if the problem goes away (**Figure 2**). A poster in an engine dynamometer shop describes a useful method of troubleshooting things that once worked well but no longer do. It's amazing how many times the questions in **Figure 2** can be useful even in prototype systems. In fact, these questions can be very useful when you're doing FPGA or software-code revisions. This technique also works well during troubleshooting when you've agitated the problem.

3. USE A SYSTEMS PERSPECTIVE

Find or draw a block diagram of the part of the system you're trying to troubleshoot (**Figure 3**). If there is feedback within the system, show the elements that generate that feedback. Drawing and studying the block diagram facilitates forming hypotheses and designing experiments based on isolating the fault to a block. Show enough detail in the block diagram that you can intelligently form hypotheses. Often, timing diagrams are useful in addition to block diagrams. Timing diagrams can help to organize observations and test data. State diagrams can also be useful.

Your brain processes pictures in different areas than it processes language. The act of drawing involves the hand and fingers, which let other parts of your brain control. Thus, drawing and looking at pictures enhances the use of language to form and check hypotheses, because doing so engages more parts of your brain.

If you begin by drawing diagrams on a whiteboard with colored markers, you may be able to begin processing the information in a fluid way and easily bring others into the process if you need to. It can also be useful to simplify the system to just one section or circuit board to investigate the problem. This approach may allow you to make simple custom test equipment that emulates the rest of the system. You can design the custom test equipment to provide the visibility you need to look at an interface. Sometimes, the test equipment may just involve a different way of plugging together circuit boards that involves better probing.

4. BUILD A RAPPORT

People generally build a rapport when they're talking face to face. A lot of what we do to build rapport is subconscious. Linguists tell us that most of our communication is nonverbal—including tone of voice and body language, for example. Of course, our minds tend to establish rapport or a feeling of being comfortably in sync with almost anything we interact with. For instance, people frequently anthropomorphize characteris-

tics of their car or their computer workstation.

You can develop a rapport with the circuit or software you're troubleshooting by learning its characteristic behavior and waveforms. When the item "misbehaves," you may become aware of it in ways you don't always consciously realize. For instance, the system may issue a series of beeps during start-up, which you may cease to be aware of, except when they disappear.

One way you can build additional rapport with a system is by adding more visual and audio aids. Attach LEDs and scope probes, for example. (In the movie *Contact*, which explored using radio telescopes to search for extraterrestrial intelligence, scientists "listened" to the data as well as looked at it.) Rapport with the system tends to be most useful while gathering data and symptoms about the problem and while running experiments.

The process also involves developing an intuition about what is likely to work and not work with the system. Verbalizing this intuition can be important in both developing hypotheses about what you think is likely to be wrong and in challenging what you initially thought was working, such as the photodiode in the previous example.

For example, you may remember that when you initially designed the power-supply circuitry for a circuit board, you weren't really sure how much filtering to add. Now the system has some strange oscillations in the analog section when you turn up the gain. You may form a hypothesis that the problem involves the power supply based on your original design uncertainty. Questioning the original designer can also impart some of this information.

A special intuitive case—the "manqué" case—involves questioning something that looks or sounds "wrong" even if you have not worked directly on the system. This case comes up especially when parts of the system are conglomerated for testing. Intuition may help when you lack a full understanding of the mechanical and electronic interactions of these system parts.

5. USE OUTSIDE HELP

Most electronics systems are the result of design work by many people. After you've checked out some of the basics, you may want to contact some of the designers to see whether they've seen the problem before. If the design of some of the equipment or software comes from other companies, you may find helpful information on their Web sites.

If you do talk to a real person, it is often most helpful to confine the discussion to the behavior of the subsystem or part that the person really understands. In his book *Nonviolent Communication* (**Reference 1**), Marshall Rosenberg describes a useful system of communications that involves four steps: observing, feeling, needing, and requesting. (You can find out about the Center for Nonviolent Communication at www.cnvc.org.) If you were talking to a busy applications engineer from Motorola, you might say, "The description in the hardware manual for the MPC850 HDLC ports on the TDMa is brief, and our HDLC channels are not functioning. We have a valid clock. I am feeling anxious because I need to provide functional hardware for the software engineers. I would like you to show me an example of how to correctly connect multiple HDLC ports using the TDMa."

SOMETIMES, YOU ONLY NEED A FOIL TO HELP RESOLVE A DIFFICULT TROUBLESHOOTING PROBLEM.

Sometimes, you only need a foil to help resolve a difficult troubleshooting problem. For instance, you might go into your colleague's office and describe the problem and the current state of your hypotheses. Your colleague might suggest other hypotheses and experiments, or you might end up making all of the suggestions yourself.

Recently, I was troubleshooting a series of problems in a large programmable-logic simulation. The system had once worked. Then I had added some necessary modifications, and some parts of the system no longer worked properly. I had been working on the changes for several weeks, and I was weary of it. Toward the end, I had a few more knotty problems to resolve. I thought that if I spoke with a colleague, and we mapped out what was going on, I could more quickly find a resolution. Then I realized that after I spent two hours explaining the problem to my colleague, he was going to ask whether I checked these aspects of the system under these conditions. Of course, these aspects were the ones I hadn't wanted to deal with.

So I looked at these background conditions. I continued to check all of the surrounding aspects of the problem that I would need to know before my colleague could begin to offer useful support. Each time I thought I was ready to talk to him, I had a few more questions to answer. By the time I had answered all of the questions I knew he would ask, I had solved the problem. I then solved a second problem in the same way. Later, I saw the colleague and thanked him for all of his help. I explained what had happened—that I got pretty annoyed at him for asking all of those questions. In the end, anticipating his questions allowed me to resolve the problem. He laughed because, of course, I had never really talked to him.

6. MANAGE EXTERNAL PERCEPTIONS

If a troubleshooting effort becomes extended, it is usually necessary to manage the perceptions of what others involved in the project think you're doing. Regular, proactive reporting, especially in meetings, can include what you've already checked out and discovered based on your hypotheses and experiments. Then, you can draft a plan of future experiments based on the hypotheses you have developed thus far.

Relating these two elements helps you in three ways. First, it reassures everyone that you are indeed proceeding in a thoughtful, methodical way. Next, it provides a useful framework for others to constructively contribute to what you're doing. Third, you set yourself up to receive congratulations once you resolve the problem.□

Author's biography

Kurt Aronow, PE, is an electrical engineer at Aztek Engineering in Boulder, CO. He received a BSEE from the University of Texas (Austin) and an MA in engineering from the University of Colorado (Boulder) in. You can reach him at 1-303-417-7073 or kurt@aztek-eng.com.

Reference

1. Rosenberg, Marshall, *Nonviolent Communication*, PuddleDancer Press, 1999.