

N9LYA TECHNICAL GUIDES

Troubleshooting Live Systems

Diagnostic Approaches for Production

Jerome M. Kutche

N9LYA

Second Edition · September 2026

About This Book

Every trade school and every textbook teaches you what things do. A relay closes a contact. A VFD ramps a motor. A PLC scans its program every few milliseconds and updates its outputs. That knowledge is necessary and it is not the job. The job is standing in front of a machine that isn't doing what it's supposed to do, with a shift waiting on it, and figuring out why — fast, safely, and correctly, in that order of urgency but not that order of importance.

Nobody teaches that part. I've spent thirty years doing it, and I got good at it the way everyone gets good at it: by being wrong a lot, early, in front of people, and paying attention to what the wrongness taught me.

This book is the method I eventually built. Not a list of common faults — those change with every new piece of equipment — but the underlying discipline that lets you walk up to any system you've never seen before and start narrowing the problem down instead of guessing at it. Guessing is what separates a two-minute fix from a four-hour one, and it is also what gets people hurt, because a guess made under time pressure has a way of becoming "I'll just check this one thing real quick" with a panel door open and a meter set to the wrong function.

Troubleshooting is a search problem before it's an electrical problem. You are hunting for one piece of bad information — a symptom — that has to be consistent with exactly one explanation, out of everything the system could possibly be doing. Get the search strategy right and the electrical knowledge tells you where to point the meter. Get it wrong and you can know everything about relay logic and still spend all day swapping parts that were never broken.

Safety gets its own third of this book, not a chapter tacked on the front. Diagnosis is disproportionately where people get hurt, because it is the phase of the work where the equipment has to be running, or at least energized, for you to see what it's doing. You cannot troubleshoot a dead circuit — you can only troubleshoot a live one, or one you've deliberately, verifiably made dead. That distinction, and knowing which state you're actually in rather than which state you assume you're in, is the difference between a career and an incident report.

Instrumentation gets real depth — six chapters on the tools, because a technician's actual skill is not knowing what a fault looks like on paper, it's knowing how to make an instrument tell you the truth about a circuit that's actively lying to you through intermittent contacts, ground loops, and symptoms that only show up when nobody's watching.

Who This Is For

Working technicians and electricians who already know Ohm's law and want the methodology nobody handed them. If you've ever swapped three parts before the one that was actually bad, this book is aimed directly at you.

Maintenance and controls people working with PLC-driven equipment, motor control centers, and mixed electrical/mechanical/pneumatic systems, where the fault could legitimately be in any of the three and you need a way to figure out which before you start disassembling things.

Apprentices and anyone early in a trade who wants to shortcut the "get yelled at, learn something" cycle by getting the method explained instead of absorbed by osmosis over a few painful years.

Anyone who read my book *Electronics: A Technician's Approach* and wants Chapter 23 of that book — Troubleshooting as a Craft — expanded into the full treatment it deserves. This book assumes the same working knowledge of components and circuits that one does, plus more real electrical and safety depth.

How to Use It

Each chapter ends with a **Bench Check** — practical questions and exercises. Do them, even the ones that seem obvious. The obvious ones are usually testing whether you'd actually catch yourself doing the wrong thing under pressure, not whether you know the right answer in the abstract.

Part V assumes you've read a PLC or automation reference, or at least know what a ladder diagram looks like — it treats the automation layer as one more system to troubleshoot, not a tutorial in programming one. If you want that depth, see PLC and Industrial Automation, elsewhere in this series.

Where I give a procedure — verifying zero energy, using the one-hand rule, sequencing a half-split search — follow it in order. These aren't suggestions dressed up as steps. They're steps, dressed down as suggestions so they read less like a checklist and more like something a person would actually say to you at a bench.

A Word Before You Start

You will occasionally be wrong, publicly, on equipment other people are waiting on. Everyone in this trade has been. The people who get good at this are not the ones who are never wrong — they're the ones who narrow down why they were wrong and fold it into the method, instead of chalking it up to bad luck and repeating the same mistake with a different symptom six months later.

You will also, if you get complacent about the safety material in Part II, hurt yourself. Diagnostic work has a specific, well-documented way of doing that: it puts your hands and your attention near energized equipment for longer, and with less structural caution, than installation or repair work does, because you're not there to change anything yet — you're just looking. "Just looking" is exactly the frame of mind that leads to reduced caution around real hazards. Part II exists to argue you out of that frame of mind before it costs you something.

About the Author

Jerome M. Kutche holds the amateur radio callsign N9LYA, first licensed in 1991 and upgraded to General class in 2003. His interest in electronics was sparked in the early 1970s by a mentor named Frank Kime, and his understanding of interference and grounding owes a great deal to the late Mike Baugh, W8AKF, an RFI mentor of rare patience.

From 1990 to 1996 he worked at Applied Instruments in Indianapolis, building RF test and measurement equipment for the telecommunications and cable television industries, alongside engineers Doyle Haywood, David Polstra, and Jerry Brown.

He then moved into industrial electrical and controls work, accumulating more than three decades across manufacturing, materials, and heavy industry — Allen-Bradley and Siemens PLC systems, motor control centers, plant electrical maintenance and supervision. He holds NFPA 70E, lockout/tagout, arc flash, and thirty-nine other professional certifications.

He has worked amateur radio contacts through the Space Shuttle SAREX program, the Mir space station, and the International Space Station, using a two-watt handheld radio.

Part I — The Discipline of Troubleshooting

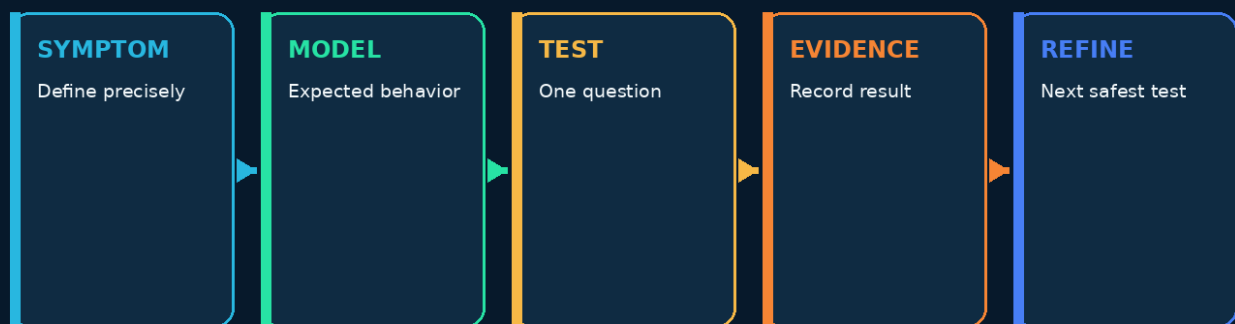
Chapter 1: Why Troubleshooting Is a Skill, Not Luck

Watch two technicians with identical training walk up to the same dead machine. One pokes around for forty minutes, swaps a contactor that turns out to be fine, and eventually finds a tripped overload by accident. The other has the machine running again in six minutes. Same certifications. Same toolbag. Different skill, and it isn't the skill either of them would list on a resume.

Troubleshooting is not "knowing electronics harder." It's a distinct discipline that sits on top of electrical knowledge the way navigation sits on top of knowing how an engine works. You can know everything about internal combustion and still get lost. You can know everything about relay logic and still spend a shift chasing a fault that a five-minute search strategy would have isolated in the first ten minutes.

Troubleshooting Is an Evidence Loop

Replace guesses with observations that change the model



N9LYA TECHNICAL GUIDES

Figure 1: Troubleshooting as an evidence loop

The Two Failure Modes

Watching technicians fail at this for three decades, I've only ever seen two actual failure modes, dressed up in a hundred different costumes.

Failure mode one: no strategy at all. The technician tests whatever is easiest to reach, in whatever order occurs to them, updating nothing about their understanding of the system as results come in. This looks like activity — meters get used, panels get opened — but it's not narrowing anything down. It's closer to shaking a vending machine than diagnosing a fault.

Failure mode two: a strategy, applied to the wrong model of the system. This one is more dangerous because it looks competent. The technician has a genuine hypothesis, tests it correctly, gets a result — and then misreads what the result actually rules out, because their mental model of how the system is wired or sequenced is subtly wrong. They eliminate the actual fault from consideration early, and everything after that is architecturally unable to find it, no matter how carefully it's executed.

Chapter 3 is about building the model that prevents failure mode two. This chapter and the next three are about the search strategy that prevents failure mode one.

What Good Troubleshooting Actually Looks Like

It looks unglamorous. A competent technician doesn't leap to the answer — they ask a sequence of questions, each one roughly cutting the remaining possibilities in half, until only one explanation survives contact with the evidence. From outside, this can look slow at first and then suddenly fast, because most of the work is silent — it's happening in the sequence of measurements, not in visible activity.

The tell that separates strategy from luck: ask the technician, at any point mid-diagnosis, "what does that measurement rule out?" A technician working a real strategy can answer instantly. One who's guessing will describe what they're about to check next, not what the last check proved.

The Cost of Poor Troubleshooting

It's worth being explicit about what bad troubleshooting actually costs, because the cost is usually invisible in the moment and only obvious in aggregate. A technician who habitually swaps parts instead of diagnosing doesn't look incompetent on any single call — the machine does get fixed, eventually, and "eventually" is easy to write off as just how that particular fault happened to go. Add it up over a year, though, and the difference between a technician working a real strategy and one working failure mode one or two is measured in real money: unnecessary parts inventory consumed, extended downtime on every call that takes three tries instead of one, and — the cost nobody puts on a spreadsheet — the accumulated risk exposure from more total time spent with panels open and hands near energized equipment, because inefficient diagnosis means more total exposure per fault found, not less.

This is the actual business case for the discipline in this book, if you ever need to make one to someone holding a budget: it isn't really about looking smart in front of a machine. It's about reducing total exposure time and total cost per fault, which strategy does and guessing doesn't, reliably, across enough faults for the pattern to show up in the numbers.

Bench Check

1. Describe, in one sentence each, the two failure modes in this chapter, and give a real or hypothetical example of each that's different from the ones in the text.
2. Next time you troubleshoot anything — at work or at home — stop after your third measurement and write down, in one sentence, what that measurement ruled out. Notice whether you can answer immediately or have to think about it.
3. Why is "no strategy" more visible from the outside than "wrong model"? Why does that make the second failure mode more dangerous to a technician's reputation over time?

Chapter 2: Symptom vs. Root Cause

A conveyor trips on overload three times a shift. The easy fix is bumping the overload setting. The overload was set correctly and the motor is drawing excess current because a bearing is failing. Bump the setting and you haven't fixed anything — you've disabled the protection that was correctly telling you something was wrong, and next week you're replacing a seized motor and possibly a gearbox instead of a \$40 bearing.

A symptom is what the system is doing that you don't want. Root cause is why. Every fault has exactly one root cause and often several symptoms, and the single most common way troubleshooting goes wrong is treating a symptom as if fixing it were the same as fixing the cause.

The Five Whys, and Where They Stop Working

The classic tool here is asking “why” repeatedly until you hit something you can actually act on:

- Conveyor tripped. Why? Overload relay opened.
- Why did the overload open? Motor current exceeded the setting.
- Why did current exceed the setting? The motor is working harder than it should.
- Why is the motor working harder? A bearing is dragging.
- Why is the bearing dragging? It's past its lubrication interval and starting to fail.

Five whys, and now you have an actionable root cause: overdue lubrication leading to bearing wear. The fix is a bearing, not a relay setting, and probably a look at the lubrication schedule too.

Where this breaks down: five is a convention, not a law. Sometimes you hit an actionable cause at why number two. Sometimes you're still one or two away at why number five, and stopping there because you've “done the five whys” gives you a false root cause. The number isn't the point. The point is: **keep asking until the answer is something you can actually change, and changing it would prevent recurrence — not just resolve the immediate symptom.**

Root Cause vs. Contributing Factor

Real failures are rarely one clean cause. The bearing failed because lubrication was overdue and because the conveyor had been running with a slight misalignment for months and because the vibration monitoring that should have caught the early-stage failure wasn't being reviewed. All three are true. Fixing only the bearing leaves the other two in place, and you'll be back.

A practical rule: when you find a cause, ask “if I fix only this, does the fault definitely not recur?” If the honest answer is “probably not, but maybe,” you haven't found the whole picture yet.

The Trap of the Convenient Culprit

Under time pressure, there's real pull toward whichever explanation lets you close the ticket fastest — usually the one requiring the least additional investigation, not the one that's actually correct. I've watched a genuinely skilled technician replace a \$600 drive because it was the newest part on the machine and “new parts fail,” when the actual fault was a loose termination in a junction box three panels away that had been loose since installation. The drive replacement didn't fix it. The next call was the same fault, an hour later, on a machine now missing a perfectly good drive.

Convenient culprits share a tell: the explanation doesn't require you to check anything else. If a hypothesis conveniently explains the symptom and conveniently requires no further verification, be suspicious of it precisely because it's convenient.

Bench Check

1. Take a real fault you've dealt with (work or home) and run the five-whys chain on it in writing. Did you stop at a genuine root cause, or at a symptom that was just further upstream than the first one?
2. Describe a contributing factor that wasn't the root cause, from any fault you've seen — something that made the failure worse or more likely without being the direct cause.

3. What's the difference between "this fix is convenient" and "this fix is correct"? Why do the two get confused under time pressure specifically?

Chapter 3: Building a Mental Model of the System

You cannot troubleshoot a system you don't have an accurate model of, and the print in the cabinet is not automatically that model — it's a claim about the system that may or may not still be true. Equipment gets modified in the field without the prints being updated more often than anyone wants to admit. Field wiring changes that "fixed" something three years ago and were never documented are the single most common reason an experienced technician's troubleshooting goes sideways on equipment they know well.

Print, As-Built, and Reality

The print is the design intent. It tells you how the system was supposed to be wired and sequenced.

The as-built, if one exists and is current, is closer to the truth — it reflects field changes made after commissioning. Many facilities don't maintain these, or maintain them inconsistently.

Reality is whatever is actually there, and it's the only one of the three you can fully trust, because it's the only one you can directly verify.

When these three disagree — and eventually they will — reality wins, every time, for the purpose of diagnosis. Use the print to build your initial hypothesis about the system, but be ready to discover, mid-diagnosis, that a wire doesn't go where it's drawn, and update your model rather than assuming your meter is lying to you.

Layered Models

A complex system is easier to hold in your head as a stack of layers than as one flat diagram:

- **Power layer** — where energy comes from and how it's distributed.
- **Control layer** — what decides when things happen (relay logic, PLC program, or both).
- **Sequence layer** — the order operations are supposed to happen in, and what conditions gate each step.
- **Feedback layer** — how the system knows whether what it commanded actually happened (limit switches, encoders, current feedback, proximity sensors).

Most faults live cleanly in one layer. A machine that won't start at all is usually a power or control-layer problem. A machine that starts but does things in the wrong order is a sequence-layer problem. A machine that commands a motion and doesn't verify it happened is a feedback-layer gap, and those are some of the ugliest faults to find because everything looks right in the control layer — the PLC genuinely did tell the motor to run — while the physical world quietly failed to comply.

Identifying which layer a symptom lives in is often 80% of the diagnosis. It immediately eliminates entire branches of the system from consideration.

Building the Model on Unfamiliar Equipment

On equipment you've never seen, you don't have the luxury of a model built from experience. Build one fast, deliberately:

1. **What is this machine's job, in one sentence?** Not how it works — what it's for.
2. **What are the major energy flows?** Where does power come in, and what physically moves or heats or transforms as a result?
3. **What's the control device, and roughly what does it decide?** You don't need to read the whole program. You need to know: does this thing use a PLC, relay logic, a dedicated controller, or some mix?
4. **What tells the system its actions succeeded?** Find the feedback layer before you need it.

Ten minutes spent on this before touching a meter routinely saves an hour of measurement on equipment you don't know.

Common Model-Building Mistakes

Assuming symmetry that isn't actually there. Three identical-looking machines on a line are rarely wired, programmed, or maintained in perfectly identical fashion by the time you encounter them years after commissioning — field modifications tend to accumulate unevenly, one machine getting a fix or tweak that its siblings never received. Don't assume what's true of one is automatically true of its neighbor, even when they left the factory identical.

Confusing “how it's supposed to work” with “how it currently works.” This is Chapter 3's central warning restated: your model should represent current reality, built from verification, not design intent inherited uncritically from a print, a manual, or “how it's always worked” as someone remembers it.

Building the model once and never updating it mid-diagnosis. Your model of the system should be a living hypothesis, revised the instant a measurement contradicts it — not a fixed reference you keep consulting while quietly ignoring evidence that it's wrong. This is the practical link back to Chapter 1's failure mode two: a wrong model, held rigidly, actively prevents you from finding a fault that contradicts it.

Overbuilding the model for the question at hand. Not every fault requires understanding the entire system in depth. Build only as much model as the current hypothesis requires, and extend it only when the evidence demands a wider view — a habit that keeps unfamiliar-equipment diagnosis fast rather than turning every call into a full system study.

Bench Check

1. Pick any machine or system you interact with regularly and sketch its four layers from memory. Where are you least confident, and why?
2. Describe a time a print or documentation turned out to be wrong. How did you discover the discrepancy — through measurement, or after guessing wrong first?
3. For an unfamiliar piece of equipment, walk through the four-question model-building sequence out loud, using a machine you've genuinely never worked on before.

Chapter 4: The Half-Split and Other Search Strategies

Once you have a working model of the system, you need a search strategy — a disciplined way to convert “the fault is somewhere in this system” into “the fault is in this specific component,” using the fewest measurements that get you there reliably.

Half-Split

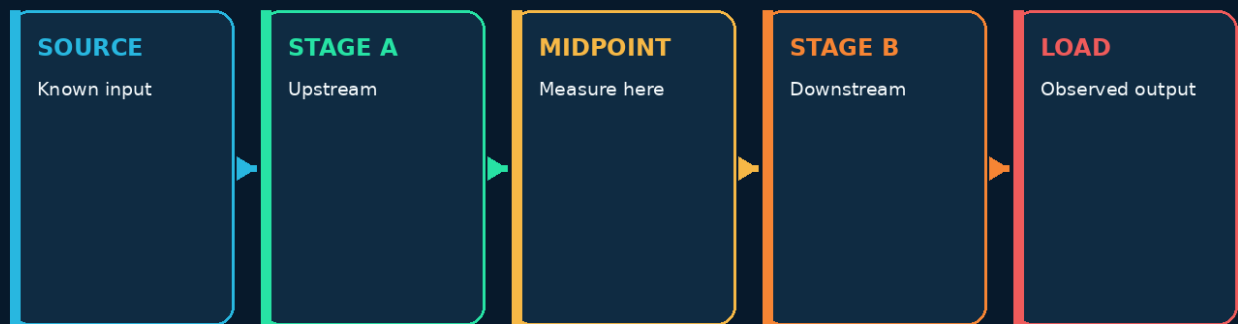
The core idea: at each step, measure at a point that, regardless of the result, eliminates roughly half the remaining possibilities. This is the same principle as searching a sorted list by checking the middle element rather than scanning from one end.

Worked example. A conveyor system has: incoming power → disconnect → motor starter → overload relay → motor. It's dead. Instead of starting at the disconnect and working toward the motor one component at a time, measure in the middle — say, at the starter's output terminals, with the starter commanded on.

- **Voltage present at starter output, motor still dead:** the fault is downstream — overload relay or motor or the wiring between them. You've eliminated the disconnect and the starter itself in one measurement.
- **No voltage at starter output:** the fault is upstream — disconnect, incoming power, or the starter isn't actually being commanded on. You've eliminated the overload relay, the motor, and the wiring between them in one measurement.

Half-Split the Signal Path

Choose a test point that eliminates the most uncertainty



N9LYA TECHNICAL GUIDES

Figure 2: Half-split signal-path testing

Either result cuts the search space roughly in half. Repeat inside whichever half survived. On a chain of five suspects, half-split typically finds the fault in two to three measurements instead of up to five.

Where half-split doesn't apply cleanly: systems that aren't a simple series chain — parallel paths, feedback loops, anything with more than one possible current path. Half-split assumes a mostly linear signal or power path. For branching systems, you need to identify which branch the symptom lives in first, then half-split within that branch.

Signal Tracing

For control and signal-level faults — a sensor input that isn't registering, a communication fault, an analog signal that's noisy — trace the signal from its source toward its destination (or the reverse), checking at each meaningful point along the path: is the signal correct here?

The difference from half-split: signal tracing is sequential rather than binary-search, because the intermediate points usually aren't equally spaced in "how much they eliminate." A signal conditioning stage, for instance, is a much higher-value place to check than a plain wire run, because a wire run rarely fails silently in the middle while a conditioning stage has active components that do.

Practical rule: trace to the highest-suspicion point first, not the geographically middle point, when you have reason to suspect one stage more than others — recent work done there, a component with a known failure mode, an environment (heat, vibration, moisture) that stresses one stage more than the rest.

Substitution

Swap a suspect component for a known-good one and see if the symptom follows the part or stays with the location.

- **Symptom follows the part** (moves with it, or a spare fixes it): that part was the fault.
- **Symptom stays with the location:** the part was fine, and something about that position — wiring, a connector, an environmental factor — is the actual cause. Putting a new part back in the same bad spot just delays the callback.

Substitution is fast and satisfying, and it's also the technique most abused into becoming a parts-swapping exercise instead of a diagnosis. Use it to confirm a specific hypothesis, not as a way to avoid forming one. "I'll just swap this and see" is a reasonable last step after other evidence points there. It's a poor first step, because you learn much less from a successful swap than you do from a measurement, and a failed swap has taught you nothing while costing you a part and time.

Injection and Isolation Testing

A fourth technique, less commonly taught but valuable on systems where you can't easily trace a signal end to end: **deliberately inject a known-good signal partway into the system and see how far it propagates correctly.**

Worked example. A control system's analog input reads incorrectly. Rather than tracing the entire sensor-to-PLC path, disconnect the sensor at the input module and inject a known reference signal (a calibrated signal source, or even a simple potentiometer wired to produce a known voltage) directly at that point. If the PLC now reads the injected value correctly, the input module and program logic are fine, and the fault is entirely upstream — the sensor itself or the wiring to it. If the PLC still reads incorrectly with a known-good signal injected directly at the input, the fault is in the input module, its configuration, or the program logic downstream of it, and the sensor and field wiring are exonerated entirely.

This technique is particularly valuable specifically because it produces a clean, unambiguous split — unlike signal tracing at intermediate points, which can be muddled by loading effects or by not being entirely sure the intermediate measurement point is representative, injection at a well-defined boundary (a connector, a terminal block, a module input) gives you a genuinely binary result.

The tradeoff: injection testing requires disconnecting something and introducing a substitute signal, which takes more setup than a simple measurement, and on a live process it may not be safe or practical to inject a fake signal into logic that's actively controlling something. Reserve it for situations where tracing and half-split have narrowed things down but not resolved the ambiguity, or for systems that are safely taken offline for the test.

Choosing Between Them

- **Series chain of suspects, roughly equal likelihood:** half-split.
- **A signal or process with a clear direction and unequal suspicion:** signal tracing toward the highest-suspicion point.
- **One strong suspect, spare readily available, low cost of being wrong:** substitution, but only after you can articulate why you suspect that part specifically.

Combining them is normal and often fastest: half-split to narrow to a subsystem, signal trace within it, substitute at the end to confirm before you close the work order.

Bench Check

1. Design a half-split sequence for a five-stage series system of your choosing (real or hypothetical). Show which measurement you'd take first and what each possible result would eliminate.
2. Describe a case where substitution alone would have been misleading — where the symptom would have stayed with the location rather than following the part.
3. When is "just swap it" actually the right first move, and why does that depend on cost and availability rather than on diagnostic purity?
4. Design an injection test for a system you're familiar with — identify a clean boundary point where you could substitute a known-good signal, and describe what each possible result would tell you.

Part II — Working Safely Under Power

Chapter 5: Deciding Whether to Work Energized

This is the single most important decision in troubleshooting, and it gets made too fast, too often, by habit rather than by evaluation. NFPA 70E is explicit on this and I'll paraphrase rather than quote regulation text at you: **energized work is permitted only when de-energizing introduces a greater hazard, or when it's infeasible due to equipment design or operational requirements** — and troubleshooting to find a fault is one of the few genuinely legitimate reasons diagnostic measurements sometimes have to happen live, because a dead circuit tells you nothing about what it's doing wrong.

"It's faster this way" is not on that list. Neither is "I've done this a hundred times." Those are the two justifications I hear most often for skipping the evaluation, and neither one is a hazard-based reason — they're convenience-based reasons wearing a hazard-based costume.

The Actual Question

Before you open a panel on anything above 50 volts (and honestly, get in the habit below that too), ask explicitly: **can this specific diagnostic step be done with the circuit de-energized, and would that still give me the information I need?**

Sometimes the honest answer is no — you need to see what the circuit does while doing it, and a dead circuit is just a dead circuit. Voltage measurements to confirm a live fault, current readings during actual operation, waveform capture on a signal that only exists when the system is running: these genuinely require energized access.

Sometimes the honest answer is "actually, yes" and you were about to work live out of habit. Continuity checks, resistance measurements, visual inspection for damage, checking torque on terminations — all of these are better and safer done dead, and doing them live tells you nothing extra while adding real risk for zero diagnostic benefit.

Decide Before Opening the System

Production pressure does not erase hazard or scope

CAN STOP?

De-energize
Preferred state

JUSTIFIED LIVE?

Documented need
Qualified person

SYSTEM RISK

Electrical / motion
Pressure / process

STOP CONDITIONS

Unknown state
Missing controls

N9LYA TECHNICAL GUIDES

Figure 3: The decision before opening a live system

End of the free sample

Troubleshooting Live Systems — Second Edition · by Jerome M. Kutche

That was the first 12 of 49 pages. The full book continues with the practical detail, worked examples and recovery steps that don't fit in a sample.

Get the full book (\$5.99, PDF + EPUB):

<https://booksrus.biz/product/troubleshooting-live-systems/>

Also on Gumroad:

<https://n9lya4.gumroad.com/l/mhvdk>



All 17 Second Edition guides in one bundle for \$89.99:

booksrus.biz/product/n9lya-technical-guides-complete-bundle

This sample may be shared freely in unmodified form. © Jerome M. Kutche. Second Edition.