DEBUGGING

Michael D’Argenio – mjdargen@ncsu.edu
Electrical Engineering – SS 2019 – Duke TIP

Overview

- Debugging is a learned behavior. It is all about your mindset and having the correct toolbox.
- Trust Nothing / Assume Nothing
- What you might think:
  - *"It should be working, but it isn't. It just doesn't make any sense!"*
- What you should think:
  - *"One of my assumptions is wrong. I need to find out which one and why!"*

Preventative Actions

- KISS – Keep It Simple Stupid
- Keep it clean
  - Avoid messy breadboards and jumper wires.
  - Format code properly and comment well.
- Follow a methodical process. Design then build.

- No matter what, you will always have bugs.
- Even the best HW/SW designers have bugs.

NINE RULES OFDEBUGGING

Adapted from the following book:

Agans, D. J. (2002). *Debugging: The nine indispensable rules for findind even the most elusive software and hardware problems*. New York, NY: American Management Association - AMACON.

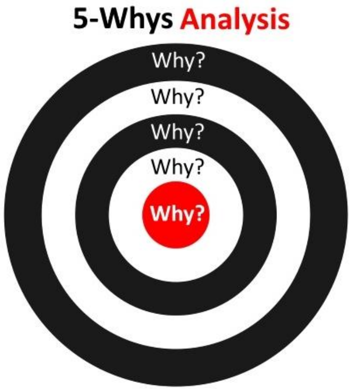
9 Rules of Debugging

1. Understand the System
2. Make it Fail
3. Quit Thinking and Look
4. Divide and Conquer
5. Change One Thing at a Time
6. Keep an Audit Trail
7. Check the Plug
8. Get a Fresh View
9. If You Didn’t Fix It, It Ain’t Fixed

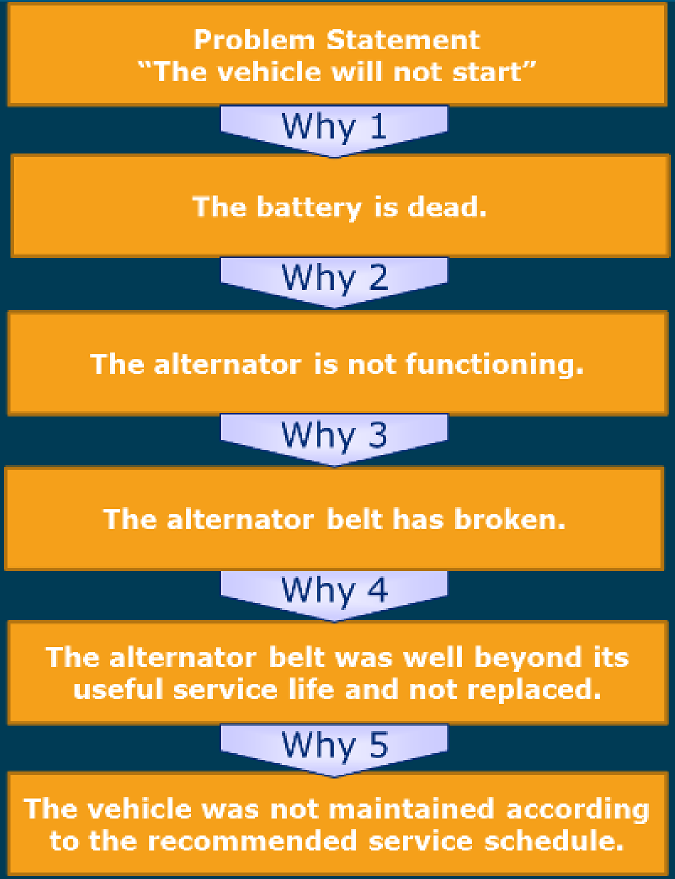
5 WHYS?

5 Whys

- Iterative technique exploring cause-and-effect.
- Attempts to sort through symptoms/effects to get to the true root cause.
- First, understand and define the problem.
- Iteratively ask why each effect is happening until you can't go further and arrive at the root cause.
- "Five" whys is a good rule-of-thumb. Can have 6 or more.



5 Whys – Example



Problem Statement
“The vehicle will not start”

Why 1

The battery is dead.

Why 2

The alternator is not functioning.

Why 3

The alternator belt has broken.

Why 4

The alternator belt was well beyond its useful service life and not replaced.

Why 5

The vehicle was not maintained according to the recommended service schedule.

HARDWARE DEBUGGING

Some Useful Tips

- If the issue is causing parts to blow up, check resistance!
  - Always check resistance before powering the first time.
  - If the issue is blowing components: don’t power, check Ω!
  - Resistance values can clue you in on if the right things are connected/disconnected.
- Check connectivity
  - Is everything connected that’s supposed to be?
  - Is everything disconnected that’s supposed to be?
- Check voltage values – use meters and oscilloscopes
- Did you design it correctly? Simulate schematic.
- Did you build it correctly? Rebuild circuit with schematic.

Circuit Debugging Activity

- https://www.allaboutcircuits.com/worksheets/basic-circuit-troubleshooting/

SOFTWARE DEBUGGING

Bugs vs. Errors

- Definitions
  - **Bug**: an incorrect statement in a program
  - **Error**: incorrect system state or behavior resulting from executing that bug
- Examples of Common Bugs
  - Misuse of programming lang.
  - Type mismatches in expressions
  - Incorrect control flow nesting
  - Incorrect comparisons (off-by-one)
  - Peripherals: misconfiguration
  - ISR not misconfigured

- Examples of Common Errors
  - Variable has wrong value
    - miscalculated
    - out-of-bounds array access
    - stack under/overflow
    - invalid pointer
  - Processor resets or hangs
  - ISR
    - never runs
    - never returns
  - Subroutine
    - never runs
    - never returns
    - returns wrong value

What's happening?

- It can be hard to really see what's going on in software.
- Drop breadcrumbs to signal what is happening in software.
  - Signal when entering/leaving subroutines
  - Signal when interrupt is triggered
  - Signal when a particular event occurs.
- Signaling methods
  - Use meters or oscilloscopes
  - Print statements – display on LCD or serial port out
  - Turn LEDs on when in particular states
  - Drive GPIO outputs – use logic analyzer

Software Debugging Activity

- http://tpcg.io/W0v7id