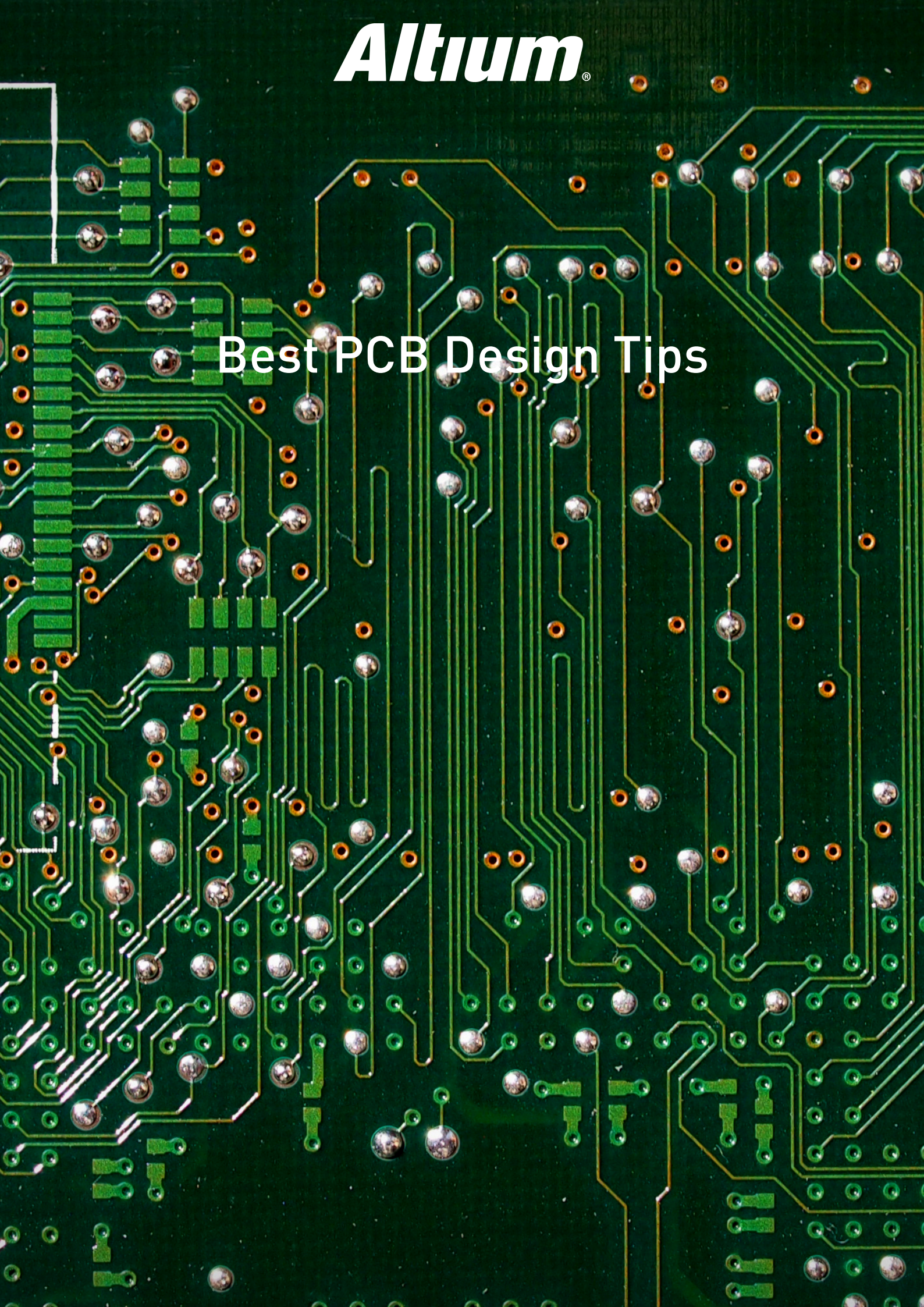
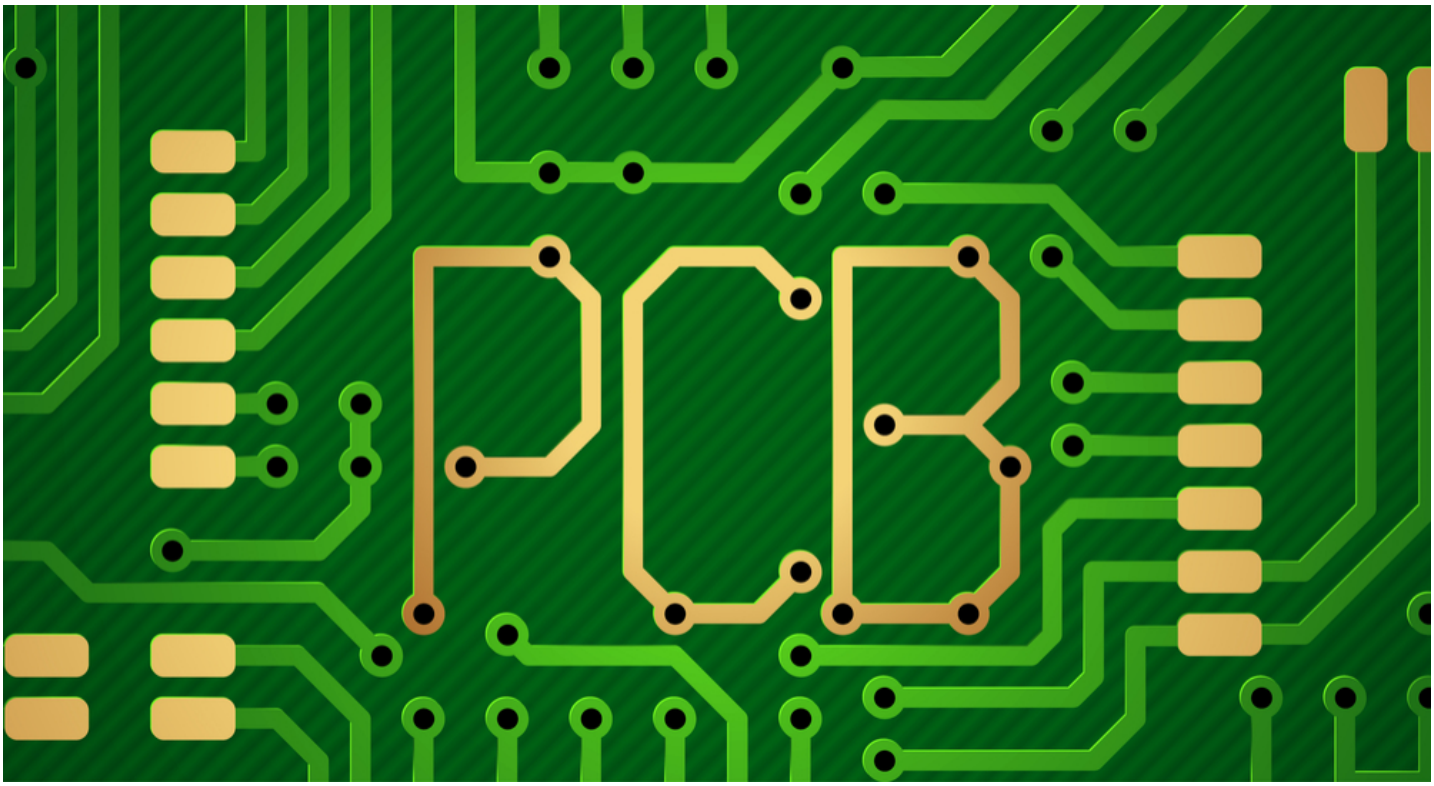
Altium®

Best PCB Design Tips





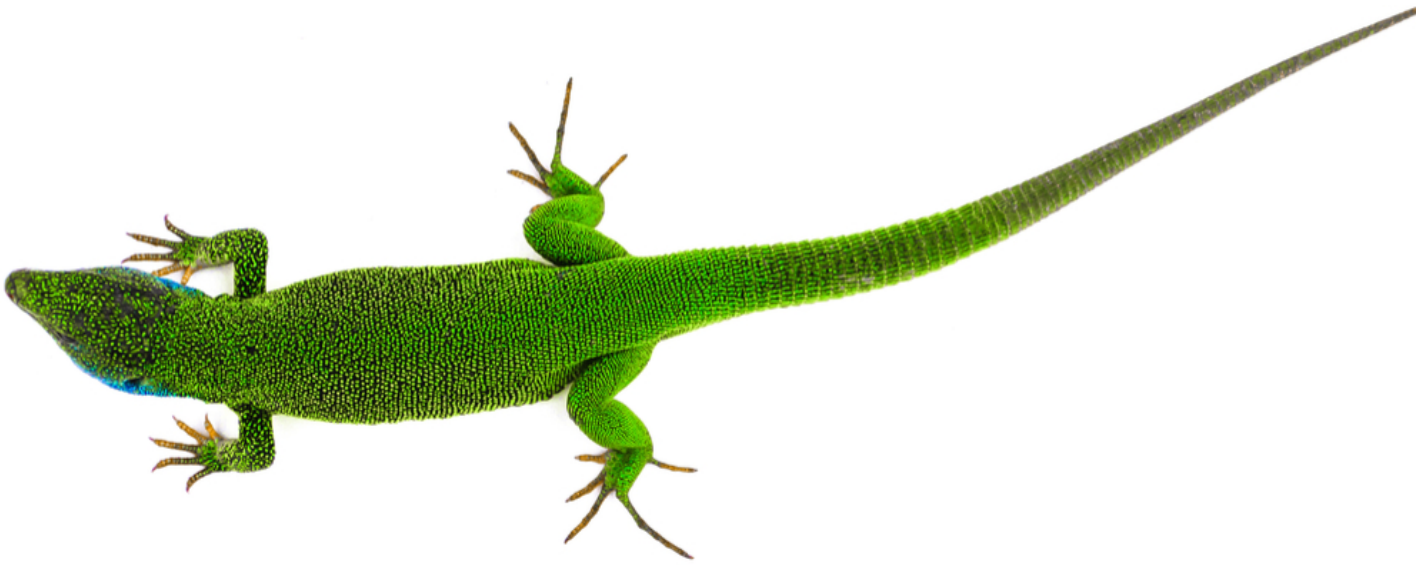
BEST PCB DESIGN TIPS

PCB Design is a complex field and it pays to learn from other's mistakes so you can avoid repeating history. Here we have brought together a small collection of insights to help inform your design process. From implementing WDTs in your board design to determining components we want you to stand on the shoulders of giants when it comes to improving your design and yielding better results. Read on for more information about environmental concerns, and component selection.

Join us as we discuss a variety of topics to help you choose the Best PCB design tips for you, including:

- Circuit Design Tips: PCB Moisture Protection for Humid Environments
- PCB Circuit Design Tips: Standard vs. Specialized Component Selection
- PCB Design Tips: Should You Include an External Watchdog Timer (WDT) in Your Board Design
- Essential PCB Design Tips: How to Implement a Watchdog Timer in Your PCB Design

CIRCUIT DESIGN TIPS: PCB MOISTURE PROTECTION FOR HUMID ENVIRONMENTS



I pride myself on being meticulous when I check my designs. I can spend hours checking schematics, footprints and the proper values of components before finalizing the design. Unfortunately, the same zeal and perfectionism cannot be said with my housekeeping skills. A cleaner would have been horrified by the accumulated dust in the corners of my house.

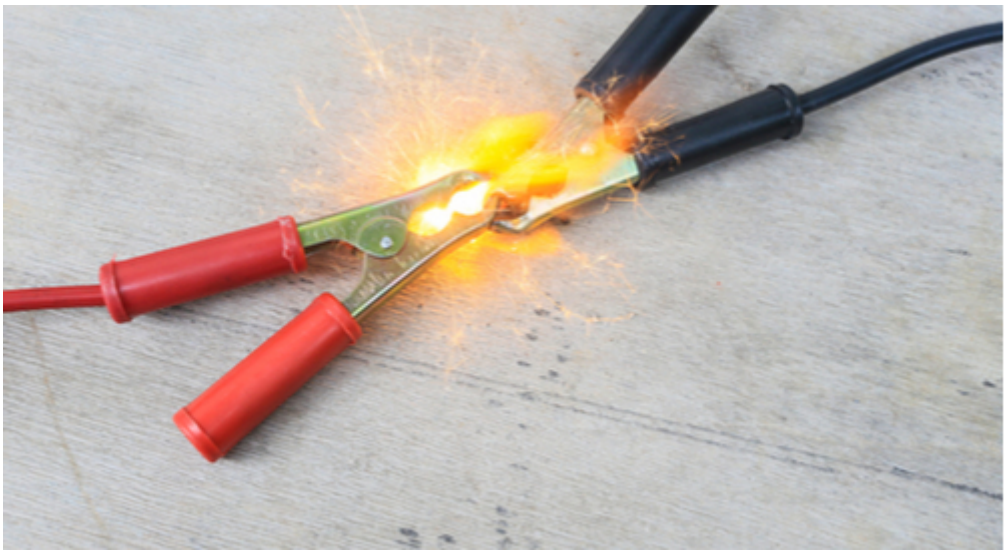
It was particularly embarrassing when I forgot to wash my dish tray and it became the home of to a family of five lizards. If your electronics PCB is placed in a damp, dark and humid environment, these friendly reptiles are going to be only one of many problems. This is why it is important to design with your board's environment in mind. Humid environments are particularly challenging, so let's explore what can go wrong and how you can prevent damaging your electronics.

HOW HUMIDITY CAN AFFECT YOUR PCB

Humidity represents the amount of water vapor in the air and relative humidity is a value that quantifies it.

A common problem caused by humidity is the formation of water droplets on electronics, particularly PCBs since it corrodes the copper traces. Condensation on a powered PCB can cause short circuits and damage to other components. Besides directly damaging the PCB, humid environment attracts reptiles and insects that can potentially cause short circuits.

I've encountered a couple of these situations in my line of work. One was a faulty emergency phone where a colony of ants decided to build a nest on the PCB. Scraping off the nests revealed irreparable damage of corroded PCB tracks. Another incident, which I'd rather forget, was when I went to investigate a damaged power management component and found a burnt lizard stuck to.



HOW TO PREVENT HUMIDITY FROM DAMAGING EMBEDDED SYSTEMS.

Obviously, these are situations we would all like to avoid. Here are some best practices for doing this.

1. Conformal Coating And Enclosure

Of course, the easiest solution for keeping your electronics safe from moisture is to apply the conformal coating and place it into an enclosure. It gives a decent protection on the PCB as the exposed copper and components are coated with materials like acrylic, urethanes, and silicone. The downside of this passive approach is that rework on the PCB can be difficult, as you'll need to strip of the coating before the components can be removed and reapplying them before the PCB is re-installed.

2. Suction Fan

Some embedded systems are commonly placed inside industrial enclosures and trapped moisture can be an annoying problem. Installing a fan that sucks the air out from the casing can help in reducing the humidity. You can see a similar application in your bathroom's exhaust fan.

3. Silica Gel

While not the most elegant solution, placing a pack of silica gel with your PCB can help in reducing the moisture content in the air. There is a reason why vitamin C came with a pack of silica gel. However, silica gel is only effective as a moisture absorbent below 60°C.

HOW TO PREVENT HUMIDITY FROM DAMAGING EMBEDDED SYSTEMS.

Obviously, these are situations we would all like to avoid. Here are some best practices for doing this.

1. Conformal Coating And Enclosure

Of course, the easiest solution for keeping your electronics safe from moisture is to apply the conformal coating and place it into an enclosure. It gives a decent protection on the PCB as the exposed copper and components are coated with materials like acrylic, urethanes, and silicone. The downside of this passive approach is that rework on the PCB can be difficult, as you'll need to strip of the coating before the components can be removed and reapplying them before the PCB is re-installed.

2. Suction Fan

Some embedded systems are commonly placed inside industrial enclosures and trapped moisture can be an annoying problem. Installing a fan that sucks the air out from the casing can help in reducing the humidity. You can see a similar application in your bathroom's exhaust fan.

3. Silica Gel

While not the most elegant solution, placing a pack of silica gel with your PCB can help in reducing the moisture content in the air. There is a reason why vitamin C came with a pack of silica gel. However, silica gel is only effective as a moisture absorbent below 60°C.

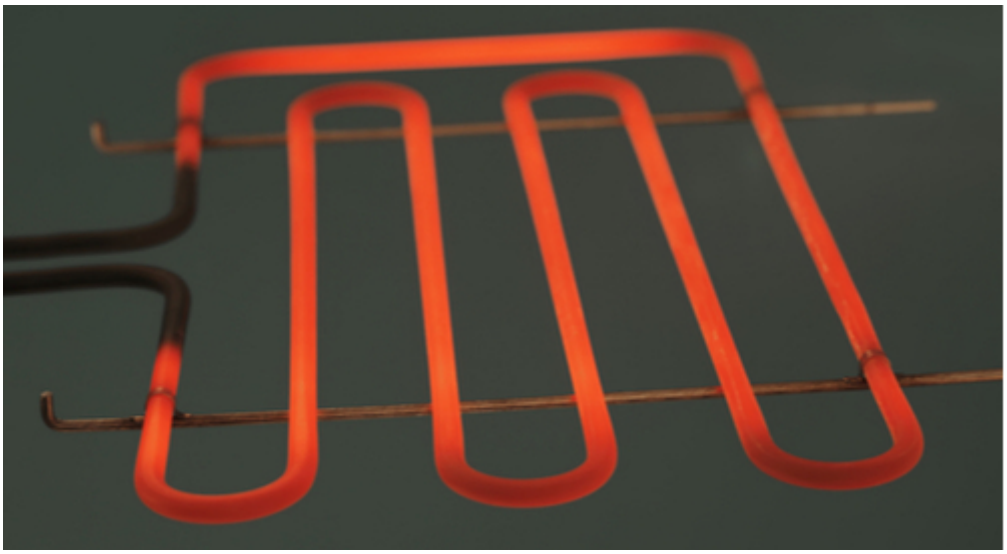
4. Heating Elements

Turning your embedded system into an intelligent mini heater can be an effective way to solve moisture problems. This works well for embedded systems that are placed in an industrial enclosure for outdoor applications. I've used a heating element to lower the relative humidity and prevent moisturization in vehicle parking machines, where condensation can get really bad in the morning.

Instead of blindly heating up the air, a humidity and temperature sensor can be installed in the enclosure along with a heating element. The intelligent proportional–integral–derivative (PID) algorithm can be applied to ensure that the humidity and the temperature of the air can be continuously regulated in an efficient manner. This will greatly reduce any chance of water droplets from forming.

Turning your embedded system into an intelligent mini heater can be an effective way to solve moisture problems. This works well for embedded systems that are placed in an industrial enclosure for outdoor applications. I've used a heating element to lower the relative humidity and prevent moisturization in vehicle parking machines, where condensation can get really bad in the morning.

Instead of blindly heating up the air, a humidity and temperature sensor can be installed in the enclosure along with a heating element. The intelligent proportional–integral–derivative (PID) algorithm can be applied to ensure that the humidity and the temperature of the air can be continuously regulated in an efficient manner. This will greatly reduce any chance of water droplets from forming.



Turning Up The Heat Lowers Relative Humidity

Needless to say, I've had no problems of burnt lizards or ant colonies after turning my embedded system into a mini heater. But it's prudent to ensure that the heated temperature does not exceed the maximum operating temperature of your components on the PCB. You can easily add the maximum allowed temperature on each components using circuit board design software like, Altium's Circuit Studio, and cross-checking on the BOM list before tuning your heating parameters.

ESSENTIAL PCB DESIGN TIPS: HOW TO IMPLEMENT A WATCHDOG TIMER IN YOUR PCB DESIGN

When you work from home there are some perks of the job. You can make your own meals, slot in some laundry over lunch, and drink all the tea you want. I use a stove-top kettle to boil water for my tea, so when I get in the writing zone, I rely on its high pitched whistle to tell me when it's done.

Except sometimes when I'm careless, I don't fit the lid on properly. As a result, the kettle remains silent despite the fact that the liquid water inside of it is rapidly becoming a gas. My careless behavior in this scenario only means that I'll be drinking less tea, in embedded systems, the consequences are much higher if you don't know how to operate a Watchdog Timer (WDT). When your WDT fails to operate, a stalled microcontroller will remain stalled and cause your embedded system remains down. Let's look at how you can get them to function correctly on the first try so that you can avoid this scenario.

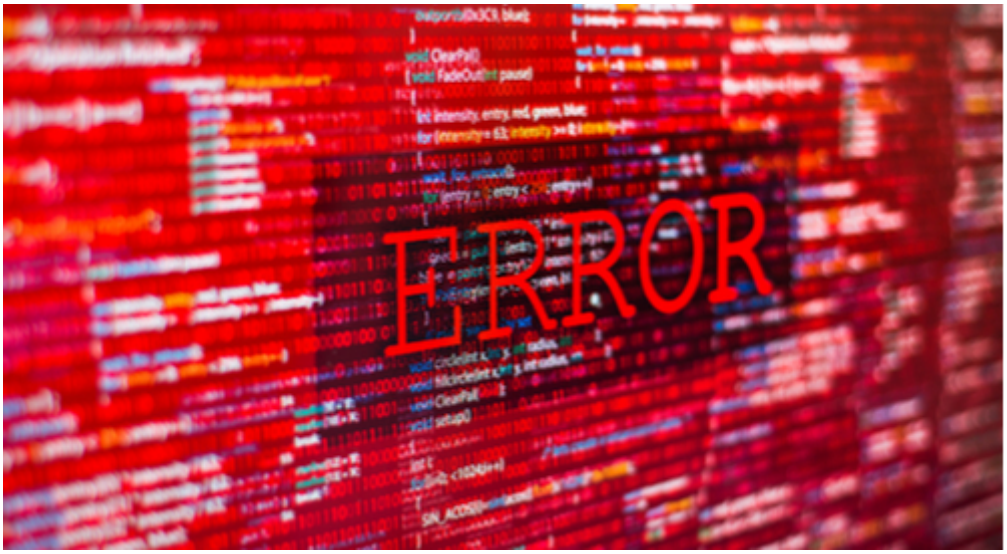
WHY EMBEDDED SYSTEMS FAILED TO RECOVER DESPITE HAVING A WDT

The WDT is a simple fail-safe feature in electronics that helps to reboot a microcontroller in the event of a hardware or software crash. The WDT is available as a separate integrated circuit (IC) or as a built-in feature within the microcontroller itself. Not using a WDT In embedded systems design is often an unpardonable sin.

The way a WDT operates is simple. It is programmed to countdown over a set time interval. Under normal operation, the microcontroller periodically refreshes the countdown timer of the WDT to prevent it from expiring. If the microcontroller is unresponsive then it will not refresh the WDT. As a result, the when WDT expires, it will trigger a pulse or signal to reset the microcontroller. This simple feature compensates for design errors or environmental factors that may cause a microcontroller to crash.

Yet, if your WDT fails, then it is unlikely that your embedded system will recover from its erroneous state. This is why it is important to pinpoint the cause of why a WDT might fail to reset the microcontroller. The most obvious answer is that the WDT is faulty. However, if you repeatedly have embedded systems in multiple units failing to recover, then there could be something up with your designs.

Actually, in my many years of designing and deploying hundreds of microcontroller-based devices, I've never encountered a single case of a failed WDT. The root cause is often simply human error.

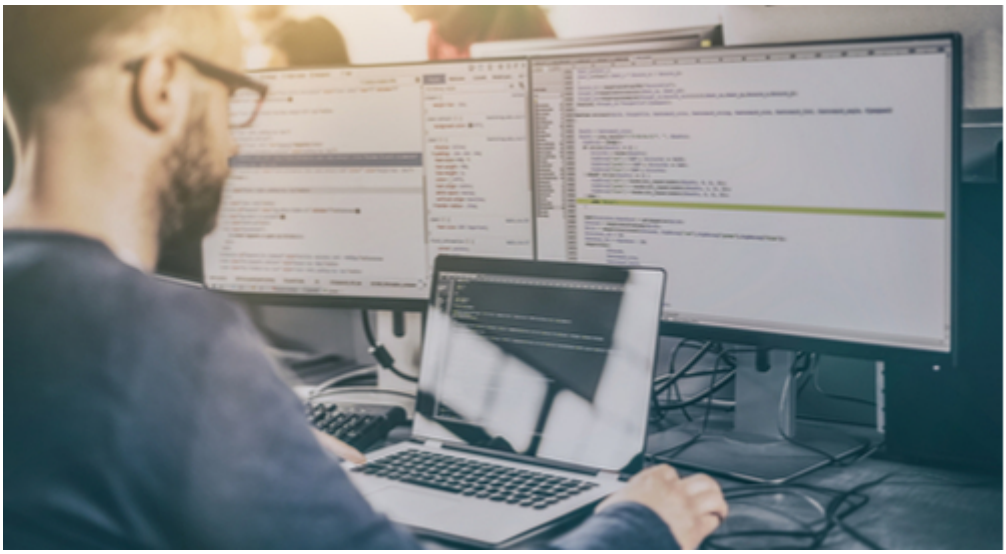


What you see when a WDT Fails to revive a stalled microcontroller

WHY A WDT MIGHT NOT OPERATE PROPERLY

For embedded systems using an internal WDT, runaways code can deactivate the WDT if the configuration bits are unintentionally overwritten. External WDTs suffer from entirely different problems. In this case, it is common to have a jumper pin that can disconnect the reset signal from the external WDT when firmware engineers are developing and debugging a program. Often these jumper pins need to be manually connected before the units are deployed on site. If they're not, then the WDT reset signals will remain disconnected fail to reset the microcontroller.

A more common reason why WDTs fail to function is because of coding errors. If the functions that refresh the WDT timers are placed in the wrong part of the program, they won't operate when they're supposed to. Firmware for microcontrollers gets complicated when there are multiple tasks with different priorities in a Real Time Operating System (RTOS). Higher priority tasks may continue to execute even when lower priority tasks are in an abnormal infinite loop. If refreshing the WDT timer the highest priority task, then the microcontroller will not be refreshed when it is not functioning correctly.



It is good practice to ensure that WDTs are refreshed correctly

HOW TO ENSURE THE WDT IS FUNCTIONING RELIABLY

Ensuring that the WDT does its job involves the firmware developer, system installer, and the hardware designer. Firmware developers should apply the best practices in programming to avoid code overruns that switch off internal WDT. Firmware developers must have a good understanding of the memory architecture of the microcontroller and how to correctly use memory pointers and allocation in the code.

Besides that, the structure of the program should be drafted out so that the WDT is refreshed at appropriate locations in the program. This means that the program will trigger a reset if an infinite loop develops at any point in the program. You can also develop a test utility to check the functionality of the WDT on-site. This also eliminates the risk of missing any disconnected jumper pins between an external WDT and the microcontroller.

Last but not least, proper PCB design is required to ensure that the WDT behaves reliably. When designing with external WDT, you'll want to ensure the refresh signal is kept away from other high-speed signals. This is to prevent cross-coupled electrical interference from refreshing the WDT. You can set up precise clearance rule using PCB design software like Altium's CircuitStudio to ensure the signal integrity of the WDT.

Having doubts about your WDT design? Talk to an expert at Altium.

PCB CIRCUIT DESIGN TIPS: STANDARD VS. SPECIALIZED COMPONENT SELECTION

As a kid, I was fascinated by hovercraft. These nearly frictionless vehicles go from land to water without pause and spin 360s on demand like a carnival ride. I knew that the cost of a full-scale build would require mowing every yard in the country, so instead, I started by constructing a homemade radio controlled model. It worked beautifully on ponds and flat parking lots, but in our hilly corner of the woods, the sleek craft required all of the small motor's power just to inch up a slight incline. For all of its amazing qualities, it was simply the wrong design for where I lived. It taught me that designs should be specifically tailored to their application. In electronics, the components you use in your circuits will influence the success of your design in its intended application. This is why it is important to carefully consider whether your PCB circuit should be designed with specialized or standard components. Here are some of the approaches that I take to determine which components will be best suited to my design's intended application.

WHAT ARE STANDARD AND SPECIALIZED COMPONENTS?

Once the scope of a project is defined and the functional block diagram has been drawn, it is time to determine how to implement the design. Many of these decisions come down to whether to use standard components or more specialized components. Standard components are readily available, they are often more affordable, and their reliability is a known quantity based on many years of test and analysis. On the other hand, one specialized component can often replace multiple standard components. This reduces the PCB's footprint and the time that it takes to design a circuit. Whether it's a high volume Class 1 design like an affordable CF (Compact Fluorescent) light bulb circuit, or a small run Class 3 Mil-spec power supply requiring a FMECA report (Failure Mode Effects and

Criticality Analysis report), your components can dictate the success of the board. While it's tempting to choose the latest, fastest microprocessor or the voltage regulator with the most features, you'll need to consider whether these features will actually help make your board successful. For example, these IoT devices come with excellent hardware, but their price tags exceed their practicality. Be prudent, you don't want to end up like the Juicero Juicer.

Choosing standard components can help lower your expenses.

SPECIALIZED COMPONENTS CAN BECOME STANDARD OVER TIME

Discretion is key when using specialized components, which also means that picking the right ones can pay off in the long term. Over time, some specialized components will evolve into standard components. This means that it's important to keep an eye on the market in order to stay competitive. In the case of the CF light, the main reason to choose standard parts is affordability. Still, specialized components are becoming more and more common in these circuits. For example, the CFs tiny ballast coils regulate power using discrete components that generate a high speed switching signal are beginning to be replaced by integrated circuits. These components include features that reduce radio frequency (RF) noise, improve power factor correction (PFC), and can even include dimming functionality. This makes for competitively priced products that have superior performance and a faster design time.

RULES, REGULATIONS, AND CHOICES

Designs for military use, like our hypothetical Mil-Spec power supply, require certain components to have reliability specifications. For example, Mean Time To Failure (MTTF). Components in this design have an effect on the reliability rating of the entire assembly, which must meet certain overall criteria. Component selection in these situations is critical for meeting the basic design requirements. There is a broad array of standard components that meet these well-tested specifications, but using them can mean using more components. This can result in more potential points of failure. Sometimes system redundancy requirements can make using specialized components a necessity. In these cases, using specialized components means finding designs from manufacturers

that have gone through the painful process of being validated, and may or may not be readily available for the next production run. Since meeting these stricter criteria takes a lot of work, manufacturers will try to keep these components on the market for as long as possible. However, there are no guarantees. There is an entire industry based around providing obsolete Mil-Spec components, and you better believe it's for a price.

Good Help Makes Great Things Happen

The right components can help you meet design regulations.

While component selection varies on a case-by-case basis, these guidelines provide a general idea of when you should opt for more specialized components, or stick with the cheaper standard ones. At the end of the day you'll need to start somewhere, and wherever that is, having BOM tools like the ones Altium provides can be a massive help in sorting through your alternatives. Historical data for both part pricing and part availability conveniently put real data in front of you to help make those big decisions at the start of a design. Customizable access and permissions for your BOM in the cloud makes team collaboration on these decisions easy, and real-time price information keeps estimates up to date as the process evolves. When it's time to build the first prototype, Altium's BOM tools can even automatically populate shopping carts, saving time and reducing errors in the purchasing process.

PCB DESIGN TIPS: SHOULD YOU INCLUDE AN EXTERNAL WATCHDOG TIMER (WDT) IN YOUR BOARD DESIGN

Would you invest in an external mouse, keyboard, and a high definition monitor when your laptop comes equipped with these features? It's a personal choice, especially when you want to turn your laptop into a portable workstation. My girlfriend, who's starting her career as a professional accountant, dismissed my choice as a waste of money.

In electronics, deciding whether to include an external watchdog timer (WDT) into your design can be an equally difficult choice. This is especially the case if your microcontroller (MCU) already has a built-in WDT. Like my girlfriend and I, design experts are divided in their opinions on whether it is a good decision. However, as a hardware designer, the final decision is ultimately yours. That is why it is important to have a good understanding of internal and external WDTs, to draw your own conclusions.

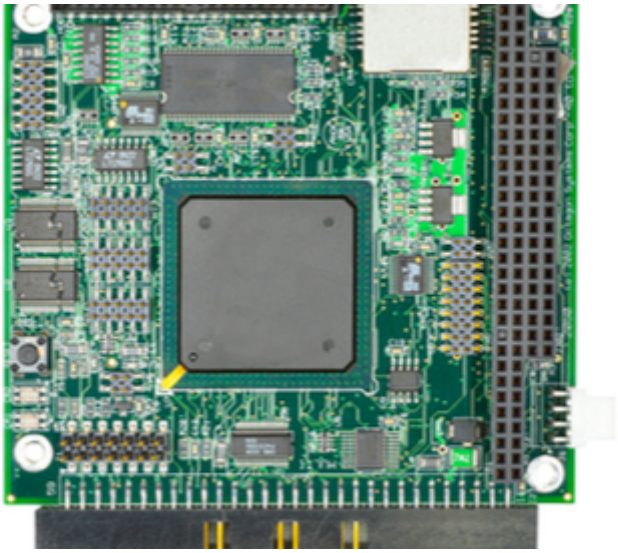
WHAT IS A WDT AND WHY DO YOU NEED IT IN YOUR DESIGN

A watchdog timer (WDT) is an electronics feature that is used to detect anomalies in embedded systems and reset the microcontroller. It usually consists of a pre-loaded timer that counts down to zero. When the pre-loaded timer expires, the microcontroller will be reset. Under normal operation, the microcontroller consistently refreshes the value of the timer to prevent it from entering the reset state. This is often candidly called "kicking the watchdog".

As much as you try to perfect your hardware and firmware, mistakes can happen. Unstable power supply, memory stack overflow or having your program trapped in a perpetual loop are common reasons why microcontrollers stall. These errors can result in a

system crash, which can be problematic in applications that have little to no tolerance for downtime. When this happens, systems equipped with a WDT will reset automatically. This is because a WDT triggers a reset of the system so that it can resume functioning normally without human intervention.

In the past I've made the mistake of not using a WDT with my microcontroller since I was overconfident about my coding skills. After experiencing multiple system crashes, which were caused by a bug that wasn't discovered in development, I learned to make WDTs a priority in all my designs. The question that remains is, should you incorporate an external WDT in MCUs that come with an internal one or rely solely on the internal WDT.



Most modern MCUs come with an internal WDT.

INTERNAL WDT VS EXTERNAL WDT, WHAT IS YOUR BEST CHOICE?

Before choosing one WDT over the other, it is important to understand how they might be the same or similar to one another. Internal WDTs are watchdog timers built within the microcontroller itself. Configuring and refreshing the WDT is done by writing values to respective registers of the WDT.

On the other hand, external WDTs are physical integrated circuits (IC) and require passive components to function. The duration of the reset countdown is often determined by capacitor's value. External WDTs are usually refreshed by sending a voltage pulse and they reset the microcontroller in the same manner.

The advantage of choosing an internal WDT over its external counterpart is that you save money by minimizing the cost of additional components and can have a smaller PCB. Since most modern MCUs are equipped with an internal WDT that are said to be reliable, this seems like a sensible choice.

Cost saving and minimalism were my motivation when I decided not to use an external WDT in one of my designs. However, my supervisor at the time changed my opinion on this. He pointed out that the internal WDT is part of the same IC as its MCU. So, if a microcontroller could fail, wouldn't it also be possible for the WDT to fail?

With today's microcontroller manufacturing capability, the likelihood of encountering an unreliable internal WDT is pretty slim. However, they do stand a chance of failing from runaway code that mistakenly deactivates the timer. Also, an internal WDT that shares the same system clock with the microcontroller has a higher chance of malfunctioning if the system clock fails.

You just can't afford a stalled system in certain applications.

In mission-critical applications, it is always safer to place an external WDT on top of the internal ones. In the end, you'll find that the cost of adding a handful of components is still lesser than the damage inflicted by the stalled system. This is particularly the case for applications in the medical, oil and gas, and automotive industry.

At the end of the day, you have the final say in choosing one over the other. Including an external WDT shouldn't be a huge problem, especially when you're designing with professional pcb design software and PCB components libraries, like Altium Designer or Altium's CircuitStudio.

Have a question about WDTs? Talk to our team at Altium now.

ADDITIONAL RESOURCES

Thank you for reading our guide on Auto-Interactive Routing. To read more Altium resources, visit the Altium resource center here or join the discussion at the bottom of each original blog post:

- Circuit Design Tips: PCB Moisture Protection for Humid Environments
- PCB Circuit Design Tips: Standard vs. Specialized Component Selection
- PCB Design Tips: Should You Include an External Watchdog Timer (WDT) in Your Board Design
- Essential PCB Design Tips: How to Implement a Watchdog Timer in Your PCB Design