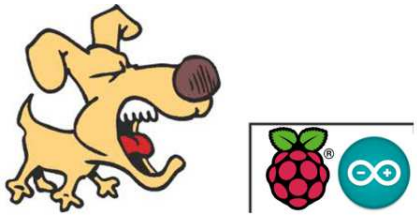
SwitchDoc Labs
Raspberry Pi, Arduino tutorials, scripts and projects

Reliable Projects 4: Internal Versus External WatchDog Timers

Posted on November 22, 2014 by John Shovic

Reliable Projects 4: Internal Versus External WatchDog Timers

Summary: In Part 4 of this series I look at the differences between an Internal and External WatchDog Timer. I also review the issues with the Arduino and the Raspberry Pi Internal WatchDog explain why an External WatchDog Timer, such as the SwitchDoc Labs Dual WatchDog Timer is a better choice in many, *but not all*, systems.



Part 1 – Introduction

Part 2 – Using the Internal WatchDog Timer for the Raspberry Pi

Part 3 – Using the Internal WatchDog Timer for the Arduino

Part 5 – Setting up an External WatchDog Timer for Raspberry Pi/Arduino Systems

The Bigger the Dog the Bigger the Bite

What is an External WatchDog Timer? It is an independent timer that is separate from the package or CPU entirely. Sometimes (such as with the SwitchDog Labs WatchDog) an entirely separate board. What is an Internal WatchDog Timer? It is a timer that is internal to the CPU and intimately related to the CPU (such as the Arduino Internal WatchDog Timer and the Raspberry Pi Internal WatchDog Timer).

In the case of the Raspberry Pi and an Arduino, an External WatchDog Timer has a **much bigger bark** than an Internal one. Why do I say this? Because there is NO WAY that the internal software, however buggy, can stop an External WatchDog Timer from doing its work, where an Internal WatchDog Timer can be shut off by software. In some designs, shutting off the Internal WatchDog Timer makes sense. In others, it doesn't.

Of course, if you want to shut off an External WatchDog Timer via software, you could by using a GPIO pin to control a relay or a transistor, but generally, you don't want to do that if you don't have to.

Problems With Internal WatchDog Timers

Summarizing the problems with Internal WatchDog Timers from Part 2 and part 3:

1. The internal WatchDog does NOT power cycle the system. It reboots the computer. This means that it does not restart in all conditions. Especially in low power / brownout conditions often experienced with Solar Powered Systems. Without some clever circuitry, sometimes the Raspberry Pi or Arduino will not come back with just a reset. **Solution:** An External WatchDog Timer can keep hitting the device until it does come back, or better yet, can power cycle the computer which will bring it back when the power levels are less brown.
2. You can stretch out your *Wto* to a lot more than 16 seconds to cover all the possible bootup sequences. *Wto* is defined as the maximum amount of time the WatchDog timer can count before it needs to be reset (in other words, when it will reboot the computer if the computer goes away. **Solution:** A circuit like the Dual WatchDog Timer goes all the way to a *Wto* of 240 seconds. That is even long enough for a Windows machine to boot. Well, most of the time.
3. If you halt the computer, you are finished. The internal WatchDog will not reboot. Not so much a problem with the Arduino, but a bigger problem with the Raspberry Pi. **Solution:** An External WatchDog is independent of what you do with the software inside. You can't screw it up.
4. On the Raspberry Pi , there are situations where the CPU is loaded up too much for your program but might the process patting the watchdog might still keep patting the dog. On the Arduino, since the the patting is done all on one thread, this is less of an issue. **Solution:** An External WatchDog is independent of what you do with the software inside. You can't screw it up.

Now, please understand, I don't hate Internal WatchDog Timers. In any system design, I always look to use the internal one first. Over the past 20 years, I find myself drifting away from using the Internal WatchDogs because of the issues above, and just maybe, I don't have to think so hard during the design. Fewer variables to control. More defined behavior. I can see what is going on by looking at the LEDs. Even without my glasses on.

Next Up:

Setting up an External WatchDog Timer for Raspberry Pi/Arduino Systems

About John Shovic

Dr. John C. Shovic is currently Managing Partner of SwitchDoc Labs, LLC and Chief Technical Officer of InstiComm, LLC, a company specializing in mobile medical software solutions for health practitioners. He has worked in industry for over thirty years and has founded multiple companies: Advance Hardware Architectures, TriGeo Network Security, Blue Water Technologies, InstiComm, LLC, and bankCDA. He has also served as a Professor of Computer Science at Eastern Washington University, Washington State University and the University of Idaho. Dr. Shovic has given over 55 invited talks and has published over 35 papers on a variety of topics on HIPAA, GLB, computer security, computer forensics, embedded systems and others.

This entry was posted in Arduino, Project Curator, Raspberry Pi, SwitchDoc Dual WatchDog Timer, SwitchDoc Products. Bookmark the permalink.

3 Responses to *Reliable Projects 4: Internal Versus External WatchDog Timers*

Pingback: *Reliable Projects 3: Using the Internal WatchDog Timer for the Arduino - SwitchDoc Labs*

Pingback: *Reliable Projects 2: Using the Internal WatchDog Timer for the Raspberry Pi - SwitchDoc Labs*

Pingback: *Reliable Projects 1: WatchDog Timers for Raspberry Pi and Arduinos - SwitchDoc Labs*

SwitchDoc LabsGoogle
Proudly powered by WordPress.