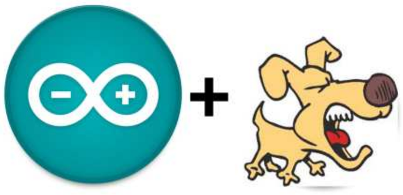
**SwitchDoc Labs**
*Raspberry Pi, Arduino tutorials, scripts and projects*

## Reliable Projects 3: Using the Internal WatchDog Timer for the Arduino

Posted on November 21, 2014 by John Shovic

# Reliable Projects 3: Using the Internal WatchDog Timer for the Arduino

Summary: In Part 3 of this series I look at how to set up the Arduino internal watchdog timer. I also talk about the issues with the Arduino internal WatchDog Timer and explain why an external WatchDog Timer, such as the SwitchDoc Labs Dual WatchDog Timer is a better choice in many, *but not all*, systems.



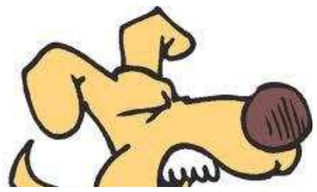
## Setting up the Arduino Internal WatchDog Timer

The Arduino is a much simpler machine than a Raspberry Pi. However, it is actually easier to hang an Arduino than it is a Raspberry Pi because all the code is single threaded on the Arduino. Single threaded means that there is only one program running at a time on the Arudino (with the exception of Interrupts, in a manner of thinking).

The point is if you only have one thread running at a time, any hang up on that thread will stop the computer. Naturally, there are other problems that can cause your code to crash and Arduino to lock up. Timeouts on peripherals, power issues, RFI, etc., etc. Bad code using the millis() function is a classic problem. You need to handle the rollover at 49.5 days if you aren't using a real time clock.

## How to use the Arduino Internal WatchDog (if you can make it work)

First of all a definition. *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.



There are a lot of things that will keep the internal WatchDog from working in the

Arduino, so beware.

Here is a way to work with the internal Arduino WatchDog timer. First of all, the *Wto* of all the Arduino models is a MAXIMUM of 8 seconds. Keep that in mind. Having a longer *Wto* covers a lot more sins in my opinion (Wto is 16 seconds on the internal Raspberry Pi WatchDog – which still isn't long enough for my taste), so the Arduino *Wto* is a bit short. I often have serial processes that run longer than 8 seconds in my design. Yes, you can incorporate patting into the code, but when you are using external libraries, that is a pain.

To experiment with your Arduino WDT , build a new sketch in the Arudino IDE. WARNING: If you are using an ArduinoMega 2560 or similar device, you may "soft brick" your device. See comments in the problems section below. My Arduino UNO from SainSmart worked fine with this sketch.

Start with

```
#include <avr/wdt.h>

 #define RESETWATCHDOG
void setup()
{
          Serial.begin(57600);
          Serial.println("");
          Serial.println ("------->Arduino Rebooted");
          Serial.println("");
          wdt_enable(WDTO_8S);


}


void loop()
{

#ifdef RESETWATCHDOG
        wdt_reset();
#endif

        Serial.println("Arduino Running");
        delay(1000);

}
```

Run the sketch and let it run for 30 seconds or so. You should never see the "Arduino Rebooted" message again after reboot. Then, comment out the RESETWATCHDOG statement like this:

*// #define RESETWATCHDOG*

Now when you run the sketch, if your WatchDog is working, then you should see the Arduino Reboot every 8 seconds or so in the serial monitor.

## Problems with the Internal Arduino WatchDog Timer

Use of the Arduino Internal WatchDogTimer is problematic at best. The Arduino WatchDog Timer has a *Wto* of 8 seconds so if you are downloading a new sketch and the old sketch has the WatchDog enabled, then you can get into an infinite reboot sequence. This is called "soft bricking". The Arduino is then pretty much worthless (without a lot of work), but it is still running. WatchDog expires, bootloader starts, bootload works for a while, WatchDog expires, etc., etc. etc. Some boot loaders now disable the WatchDog appropriately, but beware there are a lot of Arduinos out there (like the Mega 2560 – of which I am a big fan) that still don't work. You can update the bootloader but it is not an easy job. This is a problem I have run into several times.

1. The internal Arduino WatchDog does NOT power cycle the system. It reboots the Arduino via the Reset Line. This means that it does not restart in all conditions. Especially in low power / brownout conditions often experienced with Solar Powered Systems. I have seen this in Project Curacao.
2. There are problems with the bootloader (see first paragraph)
3. The internal WatchDog Timer is not completely independent of the Arduino. If your code jumps to a piece of code disables the WDT, you are finished. Try overwriting your stack to see what interesting things can happen to code in a small embedded system such as the Arduino.
4. The maximum *Wto* is 8 seconds. You can easily be in routines, such as serial communication for more than 8 seconds. Figuring out all of the possibilities and putting *wdt_reset()* calls in the right spot is difficult and with some serial routines, impossible.

## Next up:

Part 4 – Internal Versus External WatchDog Timers

**Share this:**

Google | Facebook 3 | Twitter 5 | Tumblr | Reddit | Pinterest | StumbleUpon

**Google+**

John Shovic | Follow 190

**Like this:**

Like

Be the first to like this.

**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.

View all posts by John Shovic →

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

## 4 Responses to *Reliable Projects 3: Using the Internal WatchDog Timer for the Arduino*

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*

Pingback: *Reliable Projects 4: Internal Versus External WatchDog Timers - SwitchDoc Labs*

Pingback: *在 Arduino 中使用 Watchdog 看门狗功能 | LT*