Binerry

Ask me anything

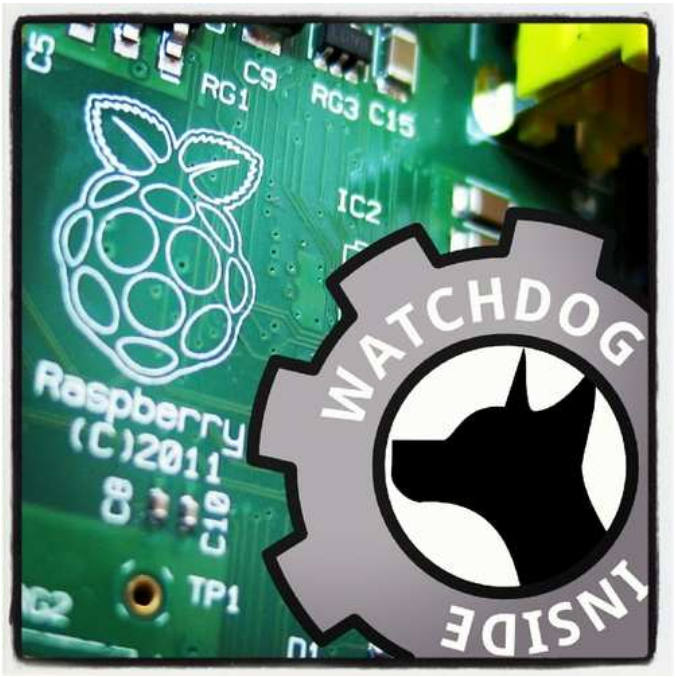
Search

My blog All of Tumblr

RSS Feed Random

Archive Mobile

Raspberry Pi Watchdog Timer



Since i'm familiar with using a watchdog timer in some microcontroller-projects, i've been thinking about watchdog-support for Raspberry Pi. Generally a watchdog timer is nothing else than a timer-register which can be realized on hardware- and/or software-side. It's main purpose is to trigger a system reset or other corrective action if something doesn't works anymore as it should be. In detail the timer-register needs to get resetted in a functional workflow before its filled up and triggers the reset or special action.

This means that the functional system must trigger a heartbeat to the watchdog (*"feeding the dog"*) within a hard (*hardware, not changeable*) or soft (*software, self-definable*) time period. An absence of some heartbeats results in reaching the time limit and triggering the reset/action.

Luckily the *BCM2835 SoC* has a hardware-based watchdog timer on board - but sadly there are no further informations about it in the actual datasheet. If i'm right informed this watchdog timeout register has 20 bits and counts down every 16µs (source: Raspberry PI BB), so the the hardware timer limit is something around 16 seconds (*16*2^20µs*) which seems to be right according to bcm2708_wdog module source.

The actual firmware includes the driver-module *bcm2708_wdog* (*or if you compiled your kernel by yourself ensure that you have enabled BCM2835 watchdog support [CONFIG_WATCHDOG=y, CONFIG_BCM2708_WDT=m|y]*). Just load it and add it if necessary to your modules-configuration for loading it at startup:

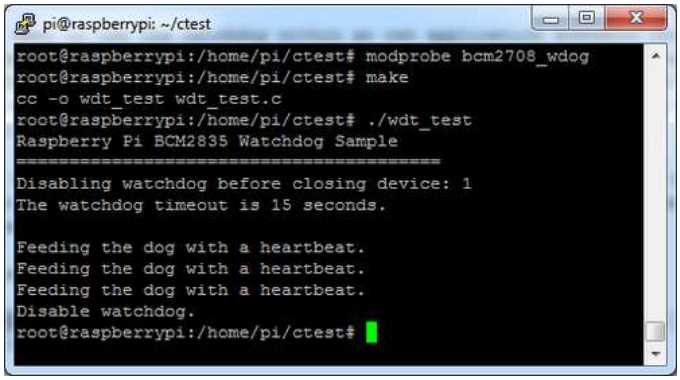
> sudo modprobe bcm2708_wdog
> sudo nano /etc/modules *(add line "bcm2708_wdog")*

This will create the device /dev/watchdog. The watchdog is automatically started once the device is opened. Once the watchdog is started, it needs to get feeded with heartbeats - this is done by writing anything but the character "V" to the device (*see watchdog api*). "V" is a magic character which will disable the watchdog - so it's important to know that just closing the device without writing the magic char to it won't stops the watchdog, quite the contrary will happen and the timer limit will be reached and the reset gets triggered.

If you want to use the watchdog within an own application scenario its recommended to use ioctl-api. For feeding the watchdog you can insert *WDIOC_KEEPALIVE* via ioctl. You can get or set the soft timer limit via *WDIOC_GETTIMEOUT* and *WDIOC_SETTIMEOUT*, but remember that its hard limited to 16s, so if you set a soft-timeout larger than this limit it will be just setted to 15s. Also remember to disable the watchdog by writing a "V" for a straight program exit.

I have written a small test-snippet for demonstrating the use of the watchdog timer within an own application. You can find it in my github-repository - Just build it and play with it - It supports a clean and a failure exit for testing purposes:

> cc -o wdt_test wdt_test.c *(or just:)* make
>
> sudo ./wdt_test *(for a clean exit)*
> sudo ./wdt_test -t *(for a failure exit)*



If you don't want to control the watchdog by your own, of course you can use a ready-to-use watchdog-daemon, which provides some more features than just feeding the dog with heartbeats every 10 or 15s. It can also run several tests to check the system healthfulness like monitoring memory and load, checking ping-results, measuring temperature or performing user-defined commands to do arbitrary tests and trigger reboot when they fail.

> sudo bash
> apt-get install watchdog chkconfig
>
> chkconfig watchdog on
> /etc/init.d/watchdog start
>
> nano /etc/watchdog.conf
> *(enable line "watchdog-device = /dev/watchdog")*

So now its time to test if everything works as it should be. Ensure that there aren't any critical processes running and try to cause a kernel panic for example. This can be tried with deactivating any swap space and a nasty forkbomb (*:(){ :|:& };:*), but more reliable will be a fake kernel-module.

If everything is set up correctly the system will reboot a few seconds later. So from now your system is protected by the watchdog.

Posted 2 years ago 6 Comments

Tagged: Raspberry Pi, BCM2835, BCM2835 SoC, Watchdog.

Comments Community Login

Sort by Best Share Favorite

Join the discussion…

d3331892828 · 6 months ago
But how do you test it? How can I lock up the machine?
1 · Reply · Share ›

fik · 2 years ago
I found the problem preventing watchdog to work properly, solution: http://www.raspberrypi.org/php...
1 · Reply · Share ›

meufeu · 2 years ago
don't works for me. my raspberry pi do not restart !
1 · Reply · Share ›

21m0n · 2 years ago
I used the kernel module and watchdog-daemon following your post. This is great, thank you very much!
1 · Reply · Share ›

Bigun · a year ago
Watchdog isn't installing:

pi@raspberrypi ~ $ sudo apt-get install watchdog chkconfig
Reading package lists... Done
Building dependency tree
Reading state information... Done
chkconfig is already the newest version.
watchdog is already the newest version.
0 upgraded, 0 newly installed, 0 to remove and 11 not upgraded.
1 not fully installed or removed.
After this operation, 0 B of additional disk space will be used.
Do you want to continue [Y/n]?
Setting up watchdog (5.12-1) ...
/run/udev or .udevdb or .udev presence implies active udev.
Aborting MAKEDEV invocation.
insserv: warning: script 'mathkernel' missing LSB tags and overrides

see more

· Reply · Share ›

Kayo · 2 years ago
Don't works for me. System reboots immediately when watchdog daemon is started. May be my watchdog.conf is wrong?

watchdog-device = /dev/watchdog
watchdog-timeout = 15 # I tried with and without this line
interval = 1

· Reply · Share ›