Watchdog on Linux/Ubuntu

Background

Watchdog timers are commonly found in embedded systems and other computer-controlled equipment where humans cannot easily access the equipment or would be unable to react to faults in a timely manner. In such systems, the computer cannot depend on a human to reboot it if it hangs; it must be self-reliant.

Odroid-N2 and ODROID-C4 support watchdog driver **meson_wdt** to control the PMU.

- Available with Linux odroid 4.9.177-28 (May 16, 2019) or higher version
- This sections is based on the **ODROID-N2**. But it also works on the **ODROID-C4**.

Test Watchdog module

Watchdog driver meson_wdt is configurable for Odroid-N2 and ODROID-C4.

You should be able to see /dev/watchdog and /dev/watchdog0 device files being created.

target

```
odroid@odroid:~$ ls -la /dev/watchdog*
crw------- 1 root root  10, 130 Jan 28  2018 /dev/watchdog
crw------- 1 root root 243,   0 Jan 28  2018 /dev/watchdog0
odroid@odroid:~$
```

Install Watchdog daemon

To install watchdog daemon

target

```
sudo apt-get install watchdog
```

Create dir for watchdog logs files

target

```
sudo mkdir -p /var/log/watchdog
```

Append the default watchdog configuration. **/etc/default/watchdog**

```
# Start watchdog at boot time? 0 or 1
run_watchdog=1
# Start wd_keepalive after stopping watchdog? 0 or 1
run_wd_keepalive=1
# Load module before starting watchdog
watchdog_module="none"
# Specify additional watchdog options here (see manpage).
watchdog_options="-s -v -c /etc/watchdog.conf"
```

Watchdog demon configuration files

You need to edit the **/etc/watchdog.conf** file to un-comment and so actually use the **/dev/watchdog** device access to the module. Otherwise the watchdog will not use the hardware and rely only on its internal code to soft-reboot a broken machine.
This configuration example sets the WDT timeout at 15 seconds. If you need a faster reboot, reduce the value of “watchdog-timeout”.

target

```
$ cat /etc/watchdog.conf
#ping         = 172.31.14.1
#ping         = 172.26.1.255
#interface        = eth0
#file         = /var/log/messages
#change           = 1407

# Uncomment to enable test. Setting one of these values to '0' disables
it.
# These values will hopefully never reboot your machine during normal
use
# (if your machine is really hung, the loadavg will go much higher than
25)
#max-load-1       = 24
#max-load-5       = 18
#max-load-15      = 12

# Note that this is the number of pages!
# To get the real size, check how large the pagesize is on your
machine.
#min-memory       = 1
```

```
#allocatable-memory   = 1

#repair-binary        = /usr/sbin/repair
#repair-timeout       = 60
#test-binary      =
#test-timeout     = 60

# The retry-timeout and repair limit are used to handle errors in a
more robust
# manner. Errors must persist for longer than retry-timeout to action a
repair
# or reboot, and if repair-maximum attempts are made without the test
passing a
# reboot is initiated anyway.
#retry-timeout        = 60
#repair-maximum       = 1

watchdog-device = /dev/watchdog

# Defaults compiled into the binary
#temperature-sensor   =
#max-temperature  = 90

# Defaults compiled into the binary
#admin            = root
#interval     = 1
#logtick                = 1
#log-dir      = /var/log/watchdog

# This greatly decreases the chance that watchdog won't be scheduled
before
# your machine is really loaded
realtime        = yes
priority        = 1

# Check if rsyslogd is still running by enabling the following line
#pidfile      = /var/run/rsyslogd.pid

watchdog-timeout        = 15
```

Note : watchdog-timeout will generally determine after which watchdog failed to keep-alive, then it will trigger reboot.

For more configuration please follow link below.
http://www.sat.dundee.ac.uk/psc/watchdog/watchdog-configure.html

Start Watchdog Service and Verify

on Ubuntu 18.04.x enable watchdog service status

In order to start watchdog service we need to create soft links of service as below.

target

```
sudo ln -s  /lib/systemd/system/watchdog.service
/etc/systemd/system/multi-user.target.wants/watchdog.service
```

target

```
sudo systemctl enable watchdog.service
sudo systemctl start watchdog.service
```

Check for watchdog service is running successfully.

target

```
odroid@odroid:~$ service watchdog status
● watchdog.service - watchdog daemon
   Loaded: loaded (/lib/systemd/system/watchdog.service; enabled;
vendor preset: enabled)
   Active: active (running) since Tue 2019-05-28 08:12:51 UTC; 2min 32s
ago
  Process: 2718 ExecStart=/bin/sh -c [ $run_watchdog != 1 ] || exec
/usr/sbin/watchdog $watchdog_options (code=exited, status=0/SUCCESS)
  Process: 2715 ExecStartPre=/bin/sh -c [ -z "${watchdog_module}" ] ||
[ "${watchdog_module}" = "none" ] || /sbin/modprobe $watchdog_module
(code=exit
 Main PID: 2720 (watchdog)
   CGroup: /system.slice/watchdog.service
           └─2720 /usr/sbin/watchdog -s -v -c /etc/watchdog.conf

May 28 08:15:13 odroid watchdog[2720]: still alive after 121
interval(s)
May 28 08:15:14 odroid watchdog[2720]: still alive after 122
interval(s)
May 28 08:15:15 odroid watchdog[2720]: still alive after 123
interval(s)
May 28 08:15:16 odroid watchdog[2720]: still alive after 124
interval(s)
May 28 08:15:17 odroid watchdog[2720]: still alive after 125
interval(s)
May 28 08:15:19 odroid watchdog[2720]: still alive after 126
```

```
interval(s)
May 28 08:15:20 odroid watchdog[2720]: still alive after 127
interval(s)
May 28 08:15:21 odroid watchdog[2720]: still alive after 128
interval(s)
May 28 08:15:22 odroid watchdog[2720]: still alive after 129
interval(s)
May 28 08:15:23 odroid watchdog[2720]: still alive after 130
interval(s)
lines 1-19/19 (END)
```

Once the watchdog demon is configured it tries to continuously reset the watchdog timer. When/if it fails to do it (because of unresponsive system), the timer will expire and the board will reboot.

Another way to test watchdog device is killing the watchdog demon after it has started.

target

```
root@odroid64:~#
root@odroid64:~# pkill -9 watchdog
```

Test Watchdog daemon

To test watchdog daemon.

Be careful when using these commands.

The commands below will cause the kernel to crash.

Use caution when following these steps, and by no means use them on a production machine.

```
echo c > /proc/sysrq-trigger
```

This will force the Linux kernel to crash. If the watchdog works properly, it will reboot the system.

target

```
root@odroid:~# echo c > /proc/sysrq-trigger
[   46.497202@3] sysrq: SysRq : Trigger a crash
[   46.497523@0] Unable to handle kernel NULL pointer dereference at
virtual address 00000000
[   46.510196@0] pgd = ffffffc0c6d62000
[   46.510356@0] [0000000000000000] *pgd=0000000000000000,
```

```
*pud=0000000000000000
[   46.517274@0] Internal error: Oops: 96000045 [#1] PREEMPT SMP
[   46.521024@0] Modules linked in: fuse squashfs cpufreq_ondemand
cpufreq_powersave cpufreq_userspace cpufreq_conservative rtc_pcf8563
i2c_meson_master sch_6
[   46.556125@0] CPU: 0 PID: 2906 Comm: bash Not tainted 4.9.177-28 #1
[   46.562356@0] Hardware name: Hardkernel ODROID-N2 (DT)
[   46.567472@0] task: ffffffc0c8de8000 task.stack: ffffffc0c7b68000
[   46.573563@0] PC is at sysrq_handle_crash+0x28/0x38
[   46.578393@0] LR is at sysrq_handle_crash+0x14/0x38
[   46.583244@0] pc : [<ffffff80094e3698>] lr : [<ffffff80094e3684>]
pstate: 60000145
[   46.590782@0] sp : ffffffc0c7b6bcd0
[   46.594248@0] x29: ffffffc0c7b6bcd0 x28: ffffffc0c8de8000
[   46.599709@0] x27: ffffff8009c12000 x26: 0000000000000040
[   46.605168@0] x25: 0000000000000123 x24: 0000000000000000
[   46.610628@0] x23: 0000000000000004 x22: ffffff800a656000
[   46.616088@0] x21: ffffff800a656488 x20: 0000000000000063
[   46.621549@0] x19: ffffff800a5f9000 x18: ffffffffffffffff
[   46.627008@0] x17: 0000007f93178028 x16: ffffff800923a770
[   46.632468@0] x15: ffffff800a5d7e90 x14: ffffff808a77b11f
[   46.637928@0] x13: 0000000000000000 x12: 0000000000000007
[   46.643388@0] x11: 0000000000000006 x10: 0000000000000358
[   46.648848@0] x9 : 0000000000000001 x8 : 0000000000000000
[   46.654308@0] x7 : ffffff800a640130 x6 : 0000000000000000
[   46.659768@0] x5 : 0000000000000000 x4 : 0000000000000000
[   46.665228@0] x3 : 0000000000000000 x2 : 00000000000409b1
[   46.670688@0] x1 : 0000000000000000 x0 : 0000000000000001
[   46.676150@0]
[   46.676150@0] SP: 0xffffffc0c7b6bc50:
[   46.681435@0] bc50  0a656000 ffffff80 00000004 00000000 00000000
00000000 00000123 00000000
[   46.689755@0] bc70  00000040 00000000 09c12000 ffffff80 c8de8000
ffffffc0 c7b6bcd0 ffffffc0
[   46.698074@0] bc90  094e3684 ffffff80 c7b6bcd0 ffffffc0 094e3698
ffffff80 60000145 00000000
[   46.706394@0] bcb0  c7b6bcd0 ffffffc0 094e3684 ffffff80 ffffffff
0000007f 00000000 00000000
[   46.714714@0] bcd0  c7b6bce0 ffffffc0 094e3d58 ffffff80 c7b6bd20
ffffffc0 094e4328 ffffff80
[   46.723034@0] bcf0  00000002 00000000 8da7b0d0 00000055 8da7b0d0
00000055 00000002 00000000
[   46.731354@0] bd10  c7b6beb0 ffffffc0 00000015 00000000 c7b6bd40
ffffffc0 092b9070 ffffff80
[   46.739674@0] bd30  3d0b0b40 ffffffc0 3cd44700 ffffffc0 c7b6bd80
ffffffc0 09238058 ffffff80
[   46.748007@0]
[   46.748007@0] X28: 0xffffffc0c8de7f80:
[   46.753368@0] 7f80  00000000 00000000 00000000 00000000 c8de7fc0
```

```
ffffffc0 00000000 00000000
[   46.761688@0] 7fa0  00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000
[   46.770008@0] 7fc0  00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000
[   46.778328@0] 7fe0  00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000
[   46.786648@0] 8000  00000008 00000000 ffffffff ffffffff 00000001
00000000 00000000 00000000
[   46.794967@0] 8020  c7b68000 ffffffc0 00000002 00400100 00000000
00000000 00000000 00000000
[   46.803288@0] 8040  00000001 00000000 00000005 00000000 ffff0866
00000000 3d284600 ffffffc0
[   46.811608@0] 8060  00000000 00000001 00000078 00000078 00000078
00000000 09c19458 ffffff80
[   46.819929@0]
[   46.819929@0] X29: 0xffffffc0c7b6bc50:
[   46.825301@0] bc50  0a656000 ffffff80 00000004 00000000 00000000
00000000 00000123 00000000
[   46.833621@0] bc70  00000040 00000000 09c12000 ffffff80 c8de8000
ffffffc0 c7b6bcd0 ffffffc0
[   46.841941@0] bc90  094e3684 ffffff80 c7b6bcd0 ffffffc0 094e3698
ffffff80 60000145 00000000
[   46.850261@0] bcb0  c7b6bcd0 ffffffc0 094e3684 ffffff80 ffffffff
0000007f 00000000 00000000
[   46.858581@0] bcd0  c7b6bce0 ffffffc0 094e3d58 ffffff80 c7b6bd20
ffffffc0 094e4328 ffffff80
[   46.866901@0] bcf0  00000002 00000000 8da7b0d0 00000055 8da7b0d0
00000055 00000002 00000000
[   46.875221@0] bd10  c7b6beb0 ffffffc0 00000015 00000000 c7b6bd40
ffffffc0 092b9070 ffffff80
[   46.883541@0] bd30  3d0b0b40 ffffffc0 3cd44700 ffffffc0 c7b6bd80
ffffffc0 09238058 ffffff80
[   46.891861@0]
[   46.893510@0] Process bash (pid: 2906, stack limit =
0xffffffc0c7b68000)
[   46.900185@0] Stack: (0xffffffc0c7b6bcd0 to 0xffffffc0c7b6c000)
[   46.906078@0] bcc0:
ffffffc0c7b6bce0 ffffff80094e3d58
[   46.914052@0] bce0: ffffffc0c7b6bd20 ffffff80094e4328
0000000000000002 000000558da7b0d0
[   46.922025@0] bd00: 000000558da7b0d0 0000000000000002
ffffffc0c7b6beb0 0000000000000015
[   46.929999@0] bd20: ffffffc0c7b6bd40 ffffff80092b9070
ffffffc03d0b0b40 ffffffc03cd44700
[   46.937972@0] bd40: ffffffc0c7b6bd80 ffffff8009238058
ffffff800a5d7000 ffffffc03cd44700
[   46.945946@0] bd60: 0000000000000002 ffffffc0c7b6beb0
000000558da7b0d0 ffffffc0c9bb7a80
[   46.953919@0] bd80: ffffffc0c7b6be30 ffffff8009239084
0000000000000002 ffffffc03cd44700
```

```
[   46.961892@0] bda0: 0000000000000000 000000558da7b0d0
ffffffc0c7b6beb0 0000000000000002
[   46.969866@0] bdc0: ffffffc0c7b6bdf0 ffffff800923c88c
0000000000000000 ffffffc0ca86ca80
[   46.977839@0] bde0: ffffffc0ca86ca80 0000000000000002
ffffffc0c7b6be30 ffffff8009239174
[   46.985812@0] be00: 0000000000000002 ffffffc03cd44700
0000000000000000 000000558da7b0d0
[   46.993786@0] be20: ffffffc03cd44700 00000000000409b1
ffffffc0c7b6be70 ffffff800923a7dc
[   47.001759@0] be40: ffffff800a5d7000 ffffffc03cd44700
ffffffc03cd44700 000000558da7b0d0
[   47.009732@0] be60: 0000000000000002 0000000000000000
0000000000000000 ffffff80090839c0
[   47.017705@0] be80: fffffffffffff1d 00000040c5119000
ffffffffffffffff 0000007f931cfbac
[   47.025679@0] bea0: 0000000020000000 0000000000000400
0000000000000000 00000000000409b1
[   47.033652@0] bec0: 0000000000000001 000000558da7b0d0
0000000000000002 0000007f932651a8
[   47.041626@0] bee0: 0000000000000000 0000000155510004
0000000000000000 0000000000000001
[   47.049599@0] bf00: 0000000000000040 0000007f932f3700
0000000000000010 0000000000000000
[   47.057572@0] bf20: 0000000000000001 000000000000270f
0000000000000002 0000000000000000
[   47.065545@0] bf40: 000000556d115bf0 0000007f93178028
0000007f93260a70 0000000000000001
[   47.073526@0] bf60: 000000558da7b0d0 0000007f93261560
0000000000000002 000000558da7b0d0
[   47.081494@0] bf80: 0000000000000002 0000007f93261648
000000556d0fe000 000000556d0eb000
[   47.089466@0] bfa0: 000000558da7ae60 0000007ff070e2d0
0000007f9317b398 0000007ff070e2d0
[   47.097438@0] bfc0: 0000007f931cfbac 0000000020000000
0000000000000001 0000000000000040
[   47.105412@0] bfe0: 0000000000000000 0000000000000000
0000000000000000 0000000000000000
[   47.113386@0] Call trace:
[   47.115988@0] Exception stack(0xffffffc0c7b6bae0 to
0xffffffc0c7b6bc10)
[   47.122572@0] bae0: ffffff800a5f9000 0000007fffffffff
ffffffc0c7b6bcd0 ffffff80094e3698
[   47.130545@0] bb00: 0000000060000145 ffffff800a77a000
ffffffc0c7b6bb30 ffffff800911278c
[   47.138519@0] bb20: ffffff8009f1c628 0000000100000000
ffffffc0c7b6bbd0 ffffff8009112938
[   47.146492@0] bb40: ffffffc0c7b6bc30 ffffff8009f53a98
ffffff800a656488 ffffff800a656000
```

```
[   47.154464@0] bb60: 0000000000000004 0000000000000000
0000000000000123 0000000000000040
[   47.162439@0] bb80: ffffff8009c12000 ffffffc0c8de8000
ffffffc0ca408240 00000000000409b1
[   47.170412@0] bba0: 0000000000000001 0000000000000000
00000000000409b1 0000000000000000
[   47.178385@0] bbc0: 0000000000000000 0000000000000000
0000000000000000 ffffff800a640130
[   47.186358@0] bbe0: 0000000000000000 0000000000000001
0000000000000358 0000000000000006
[   47.194330@0] bc00: 0000000000000007 0000000000000000
[   47.199371@0] [<ffffff80094e3698>] sysrq_handle_crash+0x28/0x38
[   47.205255@0] [<ffffff80094e3d58>] __handle_sysrq+0xb0/0x1a8
[   47.210887@0] [<ffffff80094e4328>] write_sysrq_trigger+0x90/0xa0
[   47.216871@0] [<ffffff80092b9070>] proc_reg_write+0x90/0xd0
[   47.222416@0] [<ffffff8009238058>] __vfs_write+0x60/0x150
[   47.227785@0] [<ffffff8009239084>] vfs_write+0xac/0x1b0
[   47.232985@0] [<ffffff800923a7dc>] SyS_write+0x6c/0xd8
[   47.238104@0] [<ffffff80090839c0>] el0_svc_naked+0x34/0x38
[   47.243562@0] Code: 52800020 b90a9820 d5033e9f d2800001 (39000020)
[   47.249813@0] ---[ end trace a309fd0bed7660d7 ]---
[   47.266264@0] Kernel panic - not syncing: Fatal exception
[   47.266368@0] SMP: stopping secondary CPUs
[   47.270152@0] Kernel Offset: disabled
[   47.273744@0] Memory Limit: none
[   47.288644@0] Rebooting in 5 seconds..
[   52.288923@0] reboot reason 12
bl31 reboot reason: 0xd
bl31 reboot reason: 0xc
system cmd  1.
G12B:BL:6e7c85:7898ac;FEAT:E0F83180:2000;POC:F;RCY:0;EMMC:0;READ:0;0.4
bl2_stage_init 0x01
bl2_stage_init 0x81
hw id: 0x0000 - pwm id 0x01
bl2_stage_init 0xc1
bl2_stage_init 0x02

L0:00000000
L1:00000703
L2:00008067
L3:04000000
B2:00002000
B1:e0f83180

TE: 303140

BL2 Built : 10:47:19, Jan 14 2019. g12b g152d217 - guotai.shen@droid11-
sz

Board ID = 4
Set A53 clk to 24M
```

```
Set A73 clk to 24M
Set clk81 to 24M
A53 clk: 1200 MHz
A73 clk: 1200 MHz
CLK81: 166.6M
smccc: 0004e8b4
eMMC boot @ 0
```

Watchdog on Android

To test watchdog daemon.

If the watchdog daemon does not send ping anymore, Android will be rebooted after 30 sec.

```
130|odroidn2:/ # ps -A | grep watchdogd
root          1393     2       0      0 rescuer_thread      0 S [watchdogd]
root          2357     1    6496   1596 hrtimer_nanosleep   0 S watchdogd
odroidn2:/ # pkill -9 watchdogd && stop watchdogd
```

To test watchdog for kernel panic.

Be careful when using these commands.

The commands below will cause the kernel to crash.

Use caution when following these steps, and by no means use them on a production machine.

```
echo c > /proc/sysrq-trigger
```

2020/03/19 17:30 · luke.go