Universal Serial Bug — a tale of spontaneous modem resets

Sebastian Krzyszkowiak

FOSDEM 2024, FOSS on Mobile Devices

February 3, 2024

whoami

I'm dos. I have many hobbies.

```
# dosowisko.net
```

whoami

Mobile GNU/Linux:

- Openmoko, SHR, freesmartphone.org
- Neo900
- Librem 5

The problem

Broadmobi BM818
Connected via USB 2.0

Modem resets

```
21:55:44 info kernel: usb 1-1.2: USB disconnect, device number 20
21:55:44 info kernel: option1 ttyUSB0: GSM modem (1-port) converter now
disconnected from ttyUSB0
21:55:44 info kernel: option 1-1.2:1.0: device disconnected
21:55:44 info kernel: option1 ttyUSB1: GSM modem (1-port) converter now
disconnected from ttyUSB1
21:55:44 info kernel: option 1-1.2:1.1: device disconnected
21:55:44 info kernel: option1 ttyUSB2: GSM modem (1-port) converter now
disconnected from ttyUSB2
21:55:44 info kernel: option 1-1.2:1.2: device disconnected
21:55:44 info kernel: option1 ttyUSB3: GSM modem (1-port) converter now
disconnected from ttyUSB3
21:55:44 info kernel: option 1-1.2:1.3: device disconnected
```

Modem resets

```
21:55:44 info kernel: usb 1-1.2: New USB device found, idVendor=2020,
idProduct=2060, bcdDevice= 0.00
21:55:44 info kernel: usb 1-1.2: New USB device strings: Mfr=3,
Product=2, SerialNumber=4
21:55:44 info kernel: usb 1-1.2: Product: Qualcomm CDMA Technologies MSM
21:55:44 info kernel: usb 1-1.2: Manufacturer: Qualcomm, Incorporated
21:55:44 info kernel: option 1-1.2:1.0: GSM modem (1-port) converter
detected
21:55:44 info kernel: usb 1-1.2: GSM modem (1-port) converter now
attached to ttyUSB0
21:55:44 info kernel: option 1-1.2:1.1: GSM modem (1-port) converter
detected
(...)
```

xHCI dying

```
[ 3025.568529] xhci-hcd xhci-hcd.4.auto: xHCI host not responding to
stop endpoint command.
[ 3025.568586] xhci-hcd xhci-hcd.4.auto: USBSTS:
[ 3025.576712] xhci-hcd xhci-hcd.4.auto: xHCI host controller not
responding, assume dead
[ 3025.584983] xhci-hcd xhci-hcd.4.auto: HC died; cleaning up
[ 3025.595358] usb 1-1: USB disconnect, device number 2
[ 3025.595435] usb 1-1.1: USB disconnect, device number 3
(...)

[98615.418433] usb 1-1-port2: connect-debounce failed
```

Debugging case-study

(not a groundbreaking thing)

Librem 5 USB topology

- two M.2 slots
  1. cellular modem
  2. WiFi/Bluetooth
- two USB controllers (DWC3)
  1. host-only (hub)
  2. role-swapping (USB-C)
- USB2642 hub with three ports
  1. microSD reader (internal)
  2. modem's M.2 slot
  3. WiFi/BT's M.2 slot (unused)

Universal Serial Bus


708
F510 NJE

Properties of USB 2.0:

- devices can be suspended
  - selective suspend
  - global suspend
- no device speaks unsolicited
  - except for remote wakeup
- all hubs are “smart”
- differential pair (D+, D-)

Differential pair (full-speed)

	D+	D-
J	1	0
K	0	1
SE0	0	0
SE1	1	1

- SE0 by default (pull-down).
- SE1 is illegal.
- High-speed uses lower voltages & negotiation chirps.
- Low-speed swaps J and K around.

The bug

The bug

Modem resets

Movement and signal strength dependent.

Dynamic Debug

```
echo 'file drivers/usb/core/*.c +p' >
              /sys/kernel/debug/dynamic_debug/control
```

```
14:08:08.512712: usb 1-1: usb wakeup-resume
14:08:08.532673: usb 1-1: Waited 0ms for CONNECT
14:08:08.535249: usb 1-1: finish resume
14:08:08.536903: hub 1-1:1.0: hub_resume
14:08:08.538503: usb 1-1-port1: status 0507 change 0000
14:08:08.538684: usb 1-1-port2: status 0101 change 0005
14:08:08.644711: usb usb1-port1: resume, status 0
14:08:08.645710: hub 1-1:1.0: state 7 ports 3 chg 0004 evt 0000
14:08:08.668652: usb 1-1.2: usb wakeup-resume
14:08:08.688651: usb 1-1.2: Waited 0ms for CONNECT
14:08:08.691032: usb 1-1.2: finish reset-resume
14:08:08.796917: usb 1-1.2: reset high-speed USB device number 6
                             using xhci-hcd
14:08:08.932995: usb 1-1-port2: resume, status 0
14:08:08.936984: usb 1-1-port2: status 0101, change 0005, 12 Mb/s
14:08:08.938337: usb 1-1.2: USB disconnect, device number 6
14:08:08.947333: usb 1-1.2: unregistering device
```

```
14:08:06.176782: usb usb1: usb wakeup-resume
14:08:06.188259: usb usb1-port1: status 0507 change 0000
14:08:06.240669: usb 1-1: usb wakeup-resume
14:08:06.260659: usb 1-1: Waited 0ms for CONNECT
14:08:06.263021: usb 1-1: finish resume
14:08:06.264559: hub 1-1:1.0: hub_resume
14:08:06.266101: usb 1-1-port1: status 0507 change 0000
14:08:06.266482: usb 1-1-port2: status 0503 change 0004
14:08:06.266715: usb usb1-port1: resume, status 0
14:08:06.266926: hub 1-1:1.0: state 7 ports 3 chg 0000 evt 0004
14:08:06.280247: usb 1-1.2: usb wakeup-resume
14:08:06.300565: usb 1-1.2: Waited 0ms for CONNECT
14:08:06.306261: usb 1-1.2: finish resume
14:08:06.308109: usb 1-1-port2: resume, status 0
14:08:08.405615: usb 1-1.2: usb auto-suspend, wakeup 1
14:08:08.424018: hub 1-1:1.0: hub_suspend
14:08:08.444694: usb 1-1: usb auto-suspend, wakeup 1
```

usbmon

- https://docs.kernel.org/usb/usbmon.html
- Can be used with Wireshark
- Didn't tell us anything new :(

Initial state

- the first phones ship in December 2019
- issue filled in August 2020

Initial state

- USB power management was enabled...

Initial state

- USB power management was enabled...
  - but SD card reader kept the hub active

Initial state

- USB power management was enabled...
  - but SD card reader kept the hub active
    - fixed in August 2020: https://lore.kernel.org/linux-scsi/20200623111018.31954-1-martin.kepplinger@puri.sm/T/

Initial state

- USB power management was enabled...
  - but SD card reader kept the hub active
    - fixed in August 2020: https://lore.kernel.org/linux-scsi/20200623111018.31954-1-martin.kepplinger@puri.sm/T/
- ModemManager polls every 30 seconds for signal strength...

Initial state

- USB power management was enabled...
  - but SD card reader kept the hub active
    - fixed in August 2020: https://lore.kernel.org/linux-scsi/20200623111018.31954-1-martin.kepplinger@puri.sm/T/
- ModemManager polls every 30 seconds for signal strength...
  - started listening to unsolicited messages instead

Initial state

- USB power management was enabled...
  - but SD card reader kept the hub active
    - fixed in August 2020: https://lore.kernel.org/linux-scsi/20200623111018.31954-1-martin.kepplinger@puri.sm/T/
- ModemManager polls every 30 seconds for signal strength...
  - started listening to unsolicited messages instead
    - first implementation in August 2020

Initial state

- USB power management was enabled...
  - but SD card reader kept the hub active
    - fixed in August 2020: https://lore.kernel.org/linux-scsi/20200623111018.31954-1-martin.kepplinger@puri.sm/T/
- ModemManager polls every 30 seconds for signal strength...
  - started listening to unsolicited messages instead
    - first implementation in August 2020
    - made the resets show up more often

Power management

- Disabling modem suspend made the issue go away! \o/

Power management

- Disabling modem suspend made the issue go away! \o/
- Doing so eats almost 0.5W /o\

Power management

- Disabling modem suspend made the issue go away! \o/
- Doing so eats almost 0.5W /o\
- Power management is essential and must be kept on.

Debugging efforts

- turns out the issue *only* ever happens if the hub has just been suspended
  - observed first by Elias Rudberg (I think?)
- race condition?

Debugging efforts

Playing with timeouts!

Based on Martin Kepplinger's work.

Ping reproducer

Pinging the phone with packet interval set *just right* can trigger the issue.

USB M.2 adapter

- identifying the culprit
- reproducer: blacklisted modules, timeouts set to 0
- worked with some hubs, didn't with others
- always worked with no hub
- regardless of host controller interface (xhci, ehci)

Reading the spec

Reading the spec

Device enters Suspend state after 3 ms of no activity on the bus.

- SetPortFeature(PORT_SUSPEND) sent to the hub
  - selective suspend
- host stopping communications
  - global suspend

Suspend signaling (section 7.1.7.6)

> *When a device operating in high-speed mode* ***detects that the data lines have been in the high-speed Idle state for at least 3.0 ms****, it must revert to the full-speed configuration no later than 3.125 ms (T_{WTREV}) after the start of the idle state.*

High-speed idle = SE0

Full-speed idle = J

Suspend signaling (section 7.1.7.6)

> *No earlier than 100 μs and no later than 875 μs ($T_{WTRSTHS}$) after reverting to full-speed,* ***the device must sample the state of the line. If the state is a full-speed J, the device continues with the Suspend process****. (SE0 would have indicated that the downstream facing port was driving reset, and the device would have gone into the "High-speed Detection Handshake" as described in Section 7.1.7.5.)*

The device asserts J after sampling the state of the line.

Resume signaling (section 7.1.7.7) - host

> *The host may signal resume (T_{DRSMDN}) at any time.* ***It must send the resume signaling for at least 20 ms*** *(. . .) If the port was operating in high-speed when it was suspended, the resume signaling must be ended with a transition to the high-speed idle state.*
> *(. . .)*
> ***After resuming the bus, the host must begin sending bus traffic (at least the SOF token) within 3 ms of the start of the idle state*** *to keep the system from going back into the Suspend state.*

Resume signaling = K

Resume signaling (section 7.1.7.7) - device

> ***A device with remote wakeup capability may not generate resume signaling unless the bus has been continuously in the Idle state for 5 ms (T_{WTRSM}).*** *This allows the hubs to get into their Suspend state and prepare for propagating resume signaling.* ***The remote wakeup device must hold the resume signaling for at least 1 ms but for no more than 15 ms (T_{DRSMUP})****. At the end of this period, the device stops driving the bus (puts its drivers into the high-impedance state and does not drive the bus to the J state).*

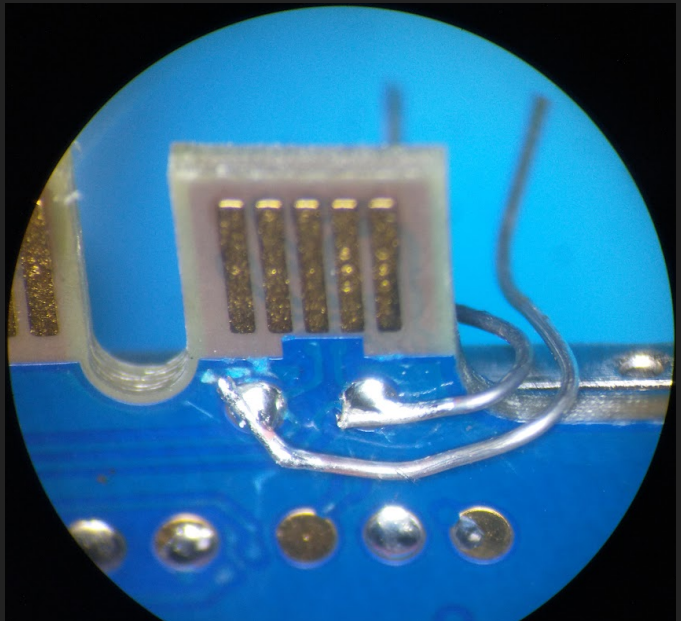
Resume signaling = K

Resume signaling (section 7.1.7.7)

The controlling hub must rebroadcast the resume signaling within 1 ms (T_{URSM}) and ensures that resume is signaled for at least 20 ms (T_{DRSMDN}). *The hub may then begin normal operation (. . .).*
The USB System Software must provide a 10 ms resume recovery time (T_{RSMRCY}) during which it will not attempt to access any device connected to the affected (just-activated) bus segment.

Getting dirty

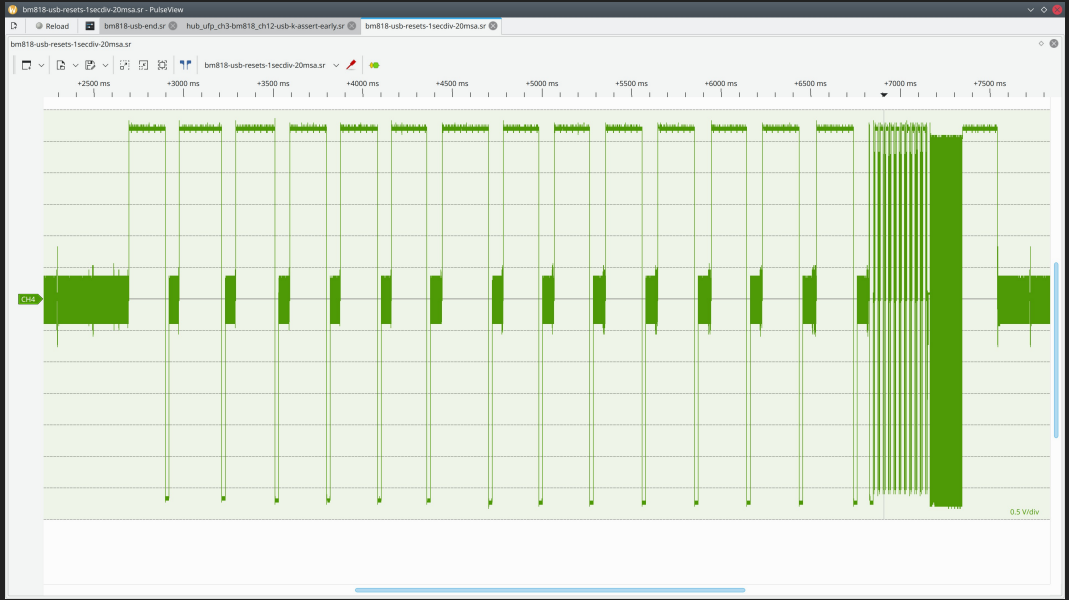
Photos by Eric Kuzmenko

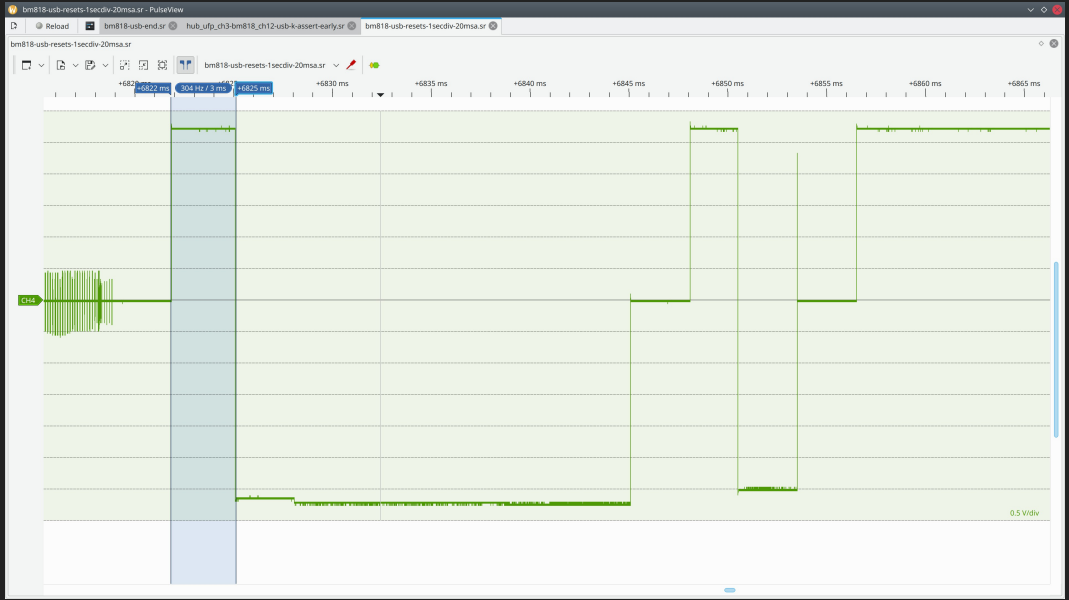



S&E
科技股份有限公司
dMobi Communication
ology Co., Ltd.
中国制造
Made in China

MAIN
AUX
BroadMesh
BM818
上海博通通信科技股份有限公司
Shanghai BroadMesh Communication Technology Co., Ltd.


Utility
Display
Acquire
Trigger
Cursors
Measure
Math
Analysis
SIGLENT
Stop
f(C4) < 2.0Hz
TRIGGER
Y2= 0.0V
C4
Y2
ΔY= 3.28V
Y1= -3.28V
Y1
C4
DC50
10X
1.00V/
FULL
0.00V
Timebase
35.0ms
10.0ms/div
200Mpts
2.00GSa/s
Trigger
C4 DC
Stop
1.00V
Edge
Falling
19:35:29
2023/3/23


bm818-usb-resets-1secdiv-20msa.sr - PulseView
Reload
bm818-usb-end.sr
hub_ufp_ch3-bm818_ch12-usb-k-assert-early.sr
bm818-usb-resets-1secdiv-20msa.sr
bm818-usb-resets-1secdiv-20msa.sr
bm818-usb-resets-1secdiv-20msa.sr
+2500 ms
+3000 ms
+3500 ms
+4000 ms
+4500 ms
+5000 ms
+5500 ms
+6000 ms
+6500 ms
+7000 ms
+7500 ms
CH4
0.5 V/div


bm818-usb-resets-1secdiv-20msa.sr - PulseView
Reload
bm818-usb-end.sr
hub_ufp_ch3-bm818_ch12-usb-k-assert-early.sr
bm818-usb-resets-1secdiv-20msa.sr
bm818-usb-resets-1secdiv-20msa.sr
bm818-usb-resets-1secdiv-20msa.sr
+6820 ms
+6822 ms
304 Hz / 3 ms
+6825 ms
+6830 ms
+6835 ms
+6840 ms
+6845 ms
+6850 ms
+6855 ms
+6860 ms
+6865 ms
CH4
0.5 V/div

Two Idle States Hypothesis

The specification requires a grace period of 5 ms before sending a remote wakeup request.

- the wording in the spec seems ambiguous

Two Idle States Hypothesis

The specification requires a grace period of 5 ms before sending a remote wakeup request.

- the wording in the spec seems ambiguous
  - is going through two separate idle states being “continuously in the Idle state”?

Two Idle States Hypothesis

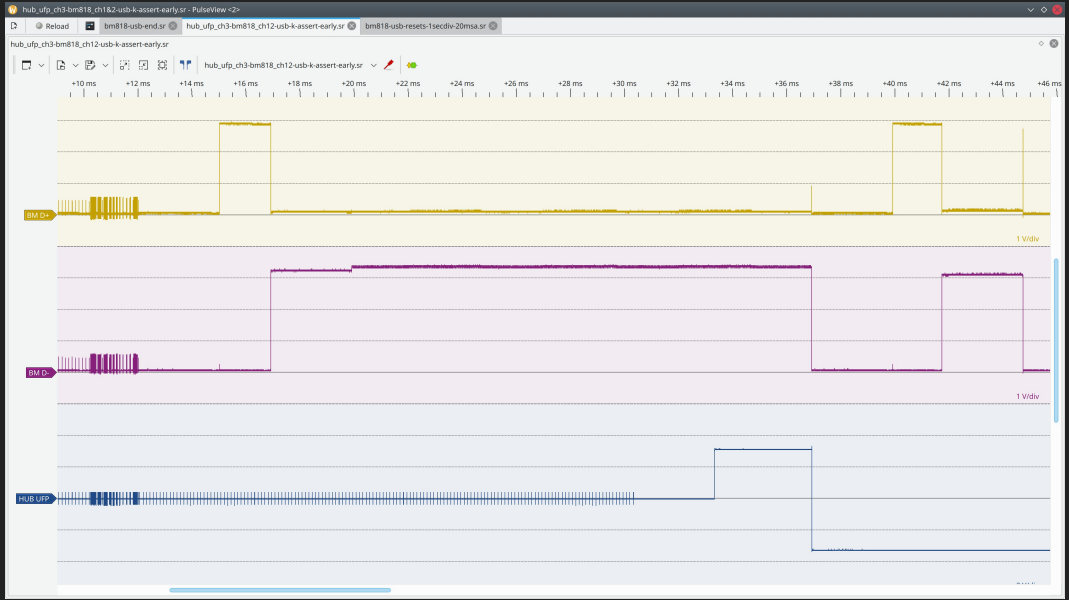
The specification requires a grace period of 5 ms before sending a remote wakeup request.

- the wording in the spec seems ambiguous
  - is going through two separate idle states being “continuously in the Idle state”?
- but the state machine description is clear


AUX
MAIN
宽翼通信
BroadMobi
S/N
IME
上海宽翼通信科技股份有限公司
Shanghai BroadMobi Communication Technology Co., Ltd.
中国制造
Made in China
202007
203202


ROHM
BD71837A
SMSC


ADJUST
QuadHands


hub_ufp_ch3-bm818_ch1&2-usb-k-assert-early.sr - PulseView <2>
Reload
bm818-usb-end.sr
hub_ufp_ch3-bm818_ch12-usb-k-assert-early.sr
bm818-usb-resets-1secdiv-20msa.sr
hub_ufp_ch3-bm818_ch12-usb-k-assert-early.sr
+10 ms
+12 ms
+14 ms
+16 ms
+18 ms
+20 ms
+22 ms
+24 ms
+26 ms
+28 ms
+30 ms
+32 ms
+34 ms
+36 ms
+38 ms
+40 ms
+42 ms
+44 ms
+46 ms
BM D+
1 V/div
BM D-
1 V/div
HUB UFP

Race condition

- When the Upstream Facing Port goes into *Suspending* state while a Downstream Facing Port is in *Resuming* state, the state machine falls apart.

Race condition

- When the Upstream Facing Port goes into *Suspending* state while a Downstream Facing Port is in *Resuming* state, the state machine falls apart.
- At that point in time, host has no way to know that a Downstream Facing Port is already attempting to wake itself up.

LKMS e-mail

> *It turns out that the USB-2 spec does not take into account the possibility of the race between a hub being suspended and one of its ports receiving a remote wakeup request from downstream. The flowcharts and discussions in Chapter 11 ignore the possibility completely. Depending on how closely a particular hub's implementation follows the treatment in the spec, if the two events happen at about the same time then it is possible for the hub to lose the wakeup request, and it is even possible for the hub to think the child device has been disconnected. I believe (though I'm not certain) that people have observed both these things happen during testing.*

Alan Stern, September 2012

https://linux-usb.vger.kernel.narkive.com/YnBDJuuP/remote-wakeup-vs-hub-suspend

LKMS e-mail

I don't know what we should do. Suggestions, anybody?

USB 2.0 Errata

Add host software requirement to suspend downstream ports before suspending a hub:

Background: There is a very unlikely possibility of a race condition between a remote wakeup event and the process of suspending a hub that could create unintended signaling events. This issue can be avoided if system software suspends a downstream facing hub ports before suspending a hub.

Change: p. 282, section 10.2.7, Add new paragraph, “The USB system should suspend all downstream facing ports of a hub before suspending the hub.”

USB 2.0 Errata

WTF???

The workaround

- add a port quirk
- don't suspend that port selectively, only pretend to
- keep the port active whenever a sibling port is active
- resume quirked downstream ports when the hub gets resumed
- rely on global suspend
- works with L5's topology
- https://source.puri.sm/Librem5/linux/-/merge_requests/672/diffs?commit_id=9cceaf04aded144a18760f1c57ea8e9741830d09

Aftermath

- Mainlining?
  - Needs to be reworked to be suitable, haven't done that yet.
  - Ensure that no downstream wakeup-capable port is suspended while the hub goes into suspend.
- Suspend/resume latency
  - Skipping selective suspend lets us suspend faster and more often.

Consulting

I can turn your money into code.

https://dosowisko.net/

Questions?