White Paper

NVMe Power and Reset Testing with the SBExpress-RM4

Controlling PERST and POWER with the sb_sdb Program

Author:

Vince Asbridge, Founder and President, SANBlaze

Table of Contents

Table of Figures

Introduction

The timing of PCIe Reset (PERST) in relation to when power is asserted can vary from platform to platform and is a source for potential issues with NVMe devices as they execute start-of-day code and firmware initialization.

While it is not practical to expose all NVMe devices to all possible implementations by various hardware server vendors and various BIOS vendors, it is possible to simulate power and reset timing using synthetic tests on the SANBlaze SBExpress Gen4 hardware.

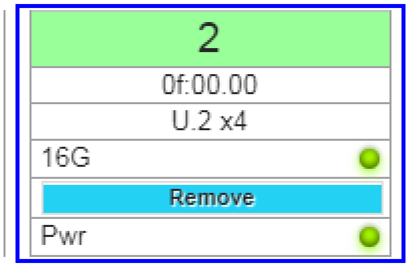
A tool accessed from the SANBlaze CLI, **sb_sdb**, can control the assertion of Power and PERST, and can vary the timing of the two signals in relation to one another to simulate complex Power and Reset timing combinations. The **sb_sdb** tool is available in Version 8.1 Beta 6 or later.

This document demonstrates the syntax for controlling Power and Reset and gives examples of typical timing scenarios.

Power Control from the SBExpress GUI

The SBExpress system GUI provides easy access to hot plug functionality for each slot on the system. The SBExpress Manager page will show the current state of Power, Presence, and Link as shown in the diagram below.

Within a glance, the GUI shows that the NVMe device present in Slot 2 is at PCIe address 0f:00.0, is in a U.2 riser, linked x4, Gen4 (16G) speed, and has power.



Selecting the **Remove** button will trigger a HotPlug event on the slot, which will cause the device to be gracefully removed from the system.

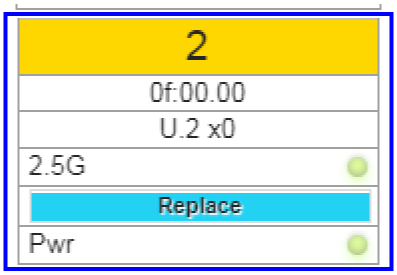
Selecting the Green **Pwr** LED will remove power from the slot creating a "surprise" removal situation, where the device is removed from the system without notification.

If you are "tailing" the */var/log/messages* file, while pressing the Green **Pwr** LED you will see:

```
==> /var/log/messages <==
Jun  5 12:01:17 DT4-1 kernel: [236402.742294] pciehp
0000:09:14.0:pcie204: Slot(20): Link Down
Jun  5 12:01:17 DT4-1 kernel: [236402.742312] unconfigure PCI device
0000:0f:00 start
```

```
Jun  5 12:01:17 DT4-1 kernel: [236402.750418] nvme_dev_disable@4185:
0000:0f:00.0: rd 1c ffffffff
Jun  5 12:01:17 DT4-1 kernel: [236402.750439] nvme_dev_disable@4210:
0000:0f:00.0: rd 14 ffffffff
Jun  5 12:01:17 DT4-1 kernel: [236402.750442]
nvme_disable_admin_queue@2880: 0000:0f:00.0: rd 00 ffffffffffffffff
Jun  5 12:01:17 DT4-1 kernel: [236402.750444] nvme_disable_ctrl@2138:
0000:0f:00.0: rd 14 ffffffff
Jun  5 12:01:17 DT4-1 kernel: [236402.750445] _nvme_disable_ctrl@2162:
0000:0f:00.0: wr 14 00460000
Jun  5 12:01:17 DT4-1 kernel: [236402.750491] 0000:0f:00.0: Controller
Disable (CC.EN=0) failed (no such device)
```

And the GUI will reflect the new state (Offline) for the NVMe device, as shown:



The Yellow background indicates that the device in Slot 2 is still present, but is not linked to PCIe, which is also indicated by the lane width now reported as x0.

The Green **Pwr** LED button provides the simplest way of removing power from a slot using only the software.

Control of Power and Reset Signals from CLI

Power, Reset and Clock can also be controlled from the CLI for use in scripting. In order to certify that an NVMe device will survive less predictable events which may happen in the field it is critical that NVMe devices be tested against events that can occur in the field, but are difficult to reproduce in the lab. These scenarios include:

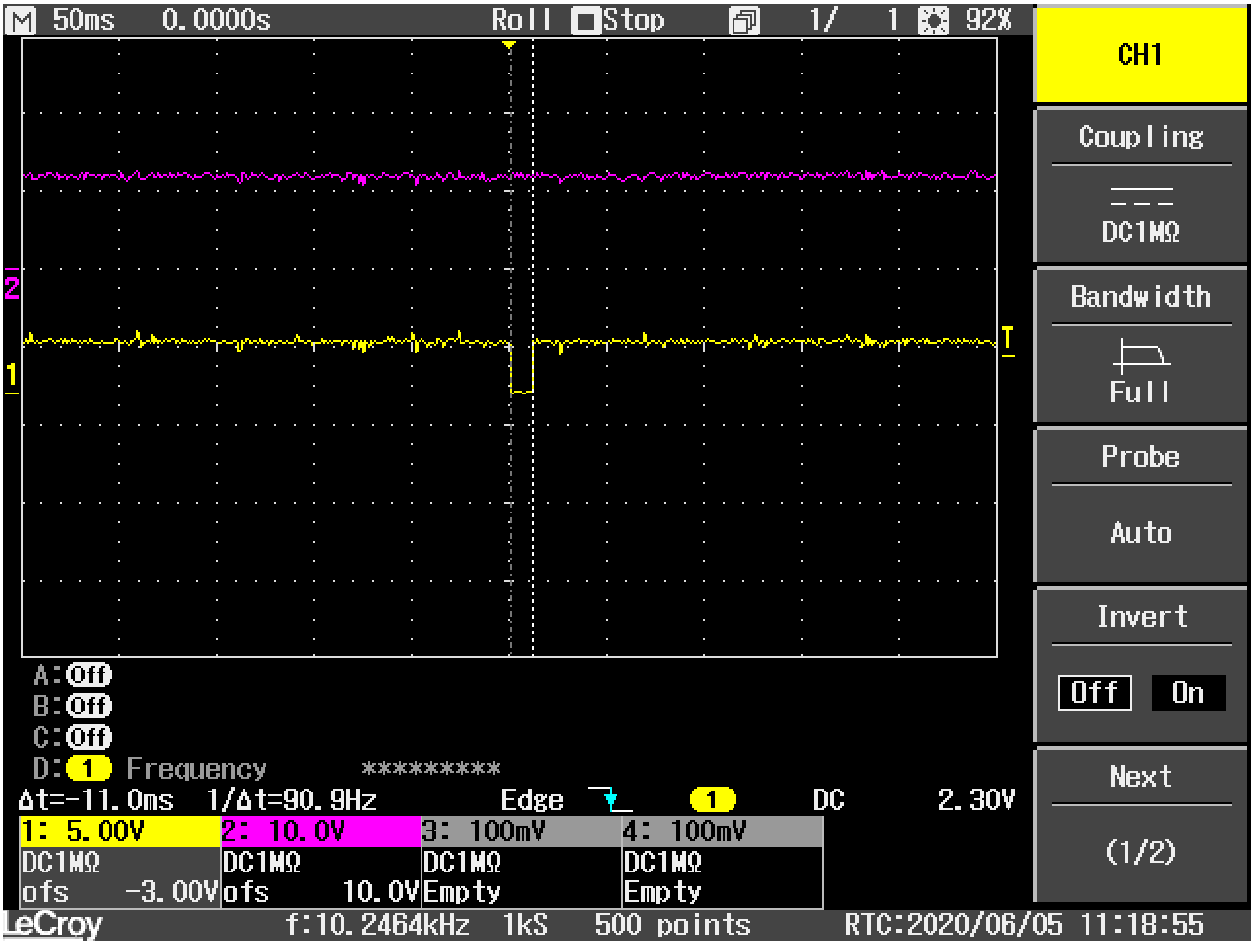
- Unexpected loss of power
- Unexpected loss of power while IO is occurring (Read and Write)
- PCIe Reset (PERST)
- Out of specification PERST timing (multiple assertions)
- Out of specification PERST in relation to Power

Using the **sb_sdb** program from the CLI, simple and complex reset and power sequences can be generated.

The following example timing diagrams were generated by the **sb_sdb** program. Other custom timing can also be achieved by varying the time between transitions.

Single PERST Assertion

The PERST signal is asserted low. Writing zero to the feature HP_PERST_ will assert the reset.



M 50ms 0.0000s Roll Stop 1/ 1 92%

A: Off
B: Off
C: Off
D: 1 Frequency **********

Δt=-11.0ms 1/Δt=90.9Hz Edge 1 DC 2.30V

1: 5.00V DC1MΩ ofs -3.00V
2: 10.0V DC1MΩ ofs 10.0V
3: 100mV DC1MΩ Empty
4: 100mV DC1MΩ Empty

f:10.2464kHz 1kS 500 points RTC:2020/06/05 11:18:55

CH1
Coupling: DC1MΩ
Bandwidth: Full
Probe: Auto
Invert: Off On
Next (1/2)

Figure 1: Single PERST Assertion

In the scope screenshot above:

Source2 (Pink) = 12V Power
Source1 (Yellow) = PERST

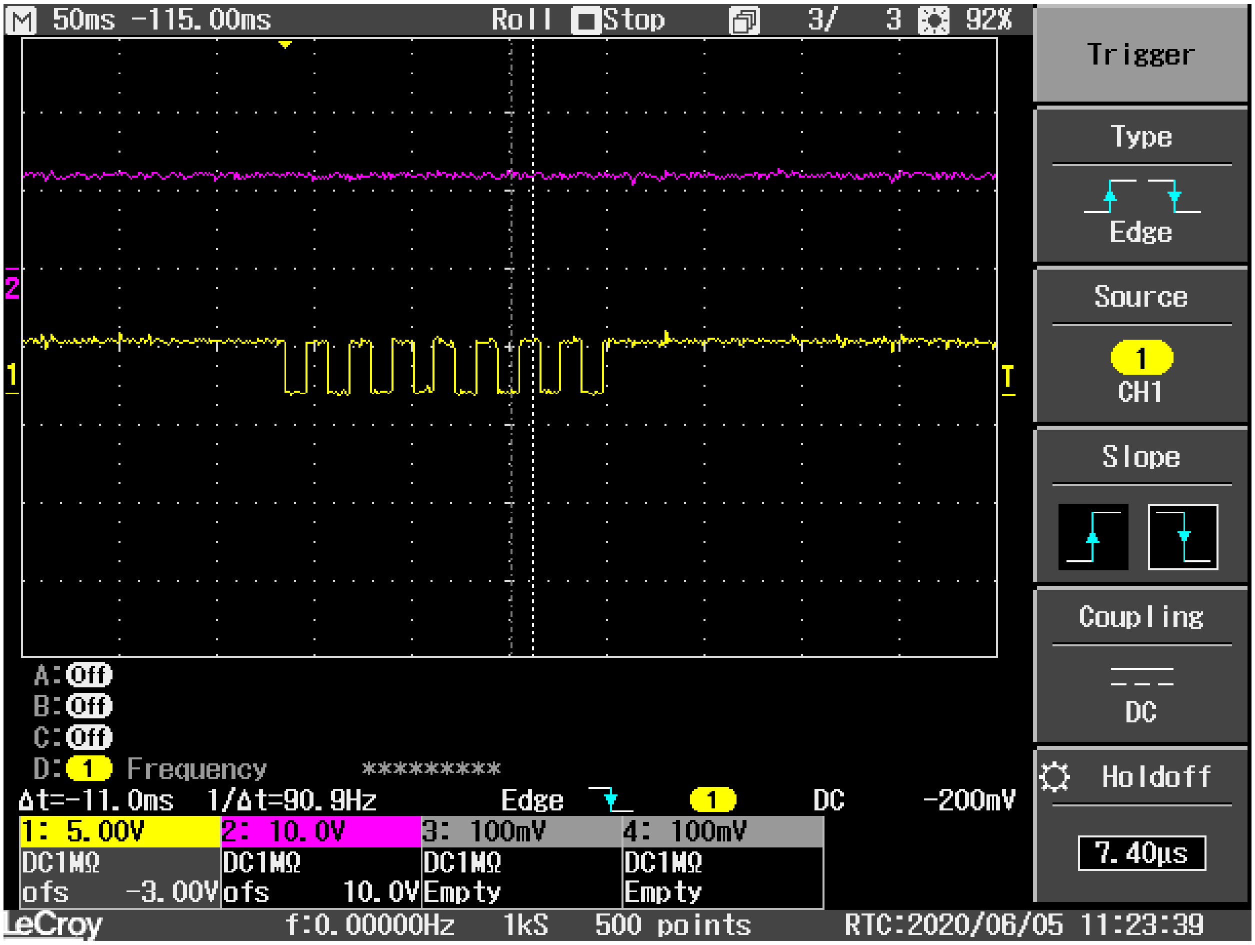
The signals are measured in Slot 0 at the U.2 connector where the NVMe UUT is connected.

The waveform above is generated with the following command:

```
sb_sdb -d 0 \
-f HP_PERST_ -w 0 -g 0 \
-f HP_PERST_ -w 1 -g 0
```

Multiple PERST Assertion

Multiple PERST assertions can be accomplished by subsequently writing 0 and 1 followed by a delay in uS (microseconds). The delay is specified with the **-g N** (glitch) argument, where N is number of uS to pause between commands. Specifying **-g 0** will produce approximately an 11mS (millisecond) assertion.



M 50ms -115.00ms Roll Stop 3/ 3 92%

A: Off
B: Off
C: Off
D: 1 Frequency **********
Δt=-11.0ms 1/Δt=90.9Hz Edge 1 DC -200mV

1: 5.00V	2: 10.0V	3: 100mV	4: 100mV
DC1MΩ	DC1MΩ	DC1MΩ	DC1MΩ
ofs -3.00V	ofs 10.0V	Empty	Empty

LeCroy f:0.00000Hz 1kS 500 points RTC:2020/06/05 11:23:39

Figure 2: Multiple PERST Assertions

In the scope screenshot above:

Source2 (Pink) = 12V Power
Source1 (Yellow) = PERST

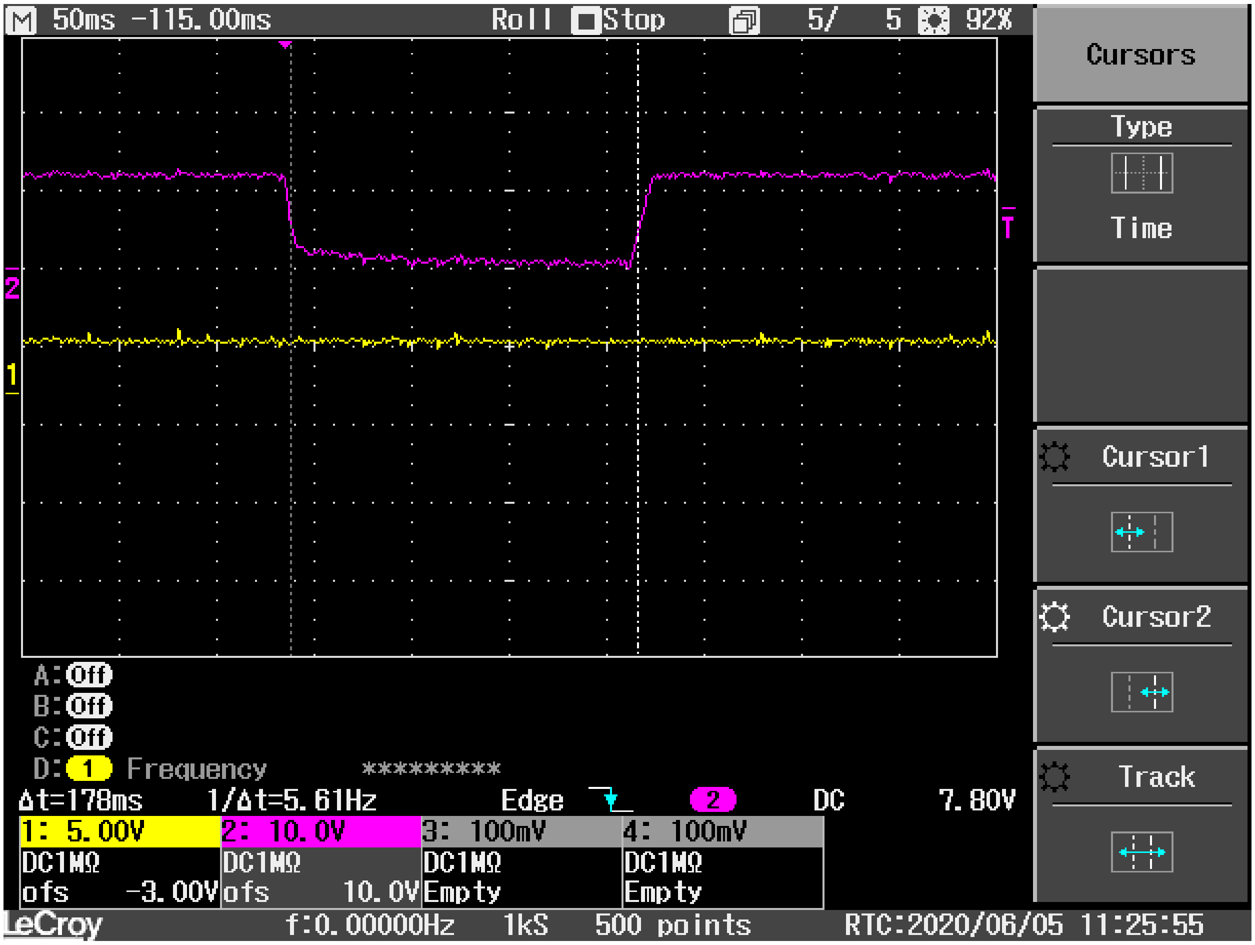
The signals are measured in Slot 0 at the U.2 connector where the NVMe UUT is connected.

The waveform above is generated with the following command:

```
sb_sdb -d 0 \
-f HP_PERST_ -w 0 -g 0 \
-f HP_PERST_ -w 1 -g 0 \
-f HP_PERST_ -w 0 -g 0 \
-f HP_PERST_ -w 1 -g 0 \
-f HP_PERST_ -w 0 -g 0 \
-f HP_PERST_ -w 1 -g 0 \
-f HP_PERST_ -w 0 -g 0 \
-f HP_PERST_ -w 1 -g 0 \
-f HP_PERST_ -w 0 -g 0 \
-f HP_PERST_ -w 1 -g 0 \
-f HP_PERST_ -w 0 -g 0 \
-f HP_PERST_ -w 1 -g 0 \
-f HP_PERST_ -w 0 -g 0 \
-f HP_PERST_ -w 1 -g 0 \
-f HP_PERST_ -w 0 -g 0 \
-f HP_PERST_ -w 1 -g 0
```

Power Control

Power is controlled with the feature HP_PWRST, it is asserted high, such that writing 1 will enable power and 0 will disable power. Like PERST, single or multiple power cycles can be specified on a single line, separated by **-g N** where N is time between commands. Specifying **-g 0** between the deassertion of power and assertion of power will produce the minimum power off cycle which is approximately 178mS.



```
M 50ms -115.00ms            Roll  Stop      5/   5   92%
A: Off
B: Off
C: Off
D: 1 Frequency          **********
Δt=178ms   1/Δt=5.61Hz       Edge      2      DC       7.80V
1: 5.00V         2: 10.0V        3: 100mV    4: 100mV
DC1MΩ            DC1MΩ           DC1MΩ       DC1MΩ
ofs      -3.00V  ofs      10.0V  Empty       Empty
LeCroy      f:0.00000Hz    1kS    500 points    RTC:2020/06/05 11:25:55
```

Figure 3: Minimum Power Deassertion

In the scope screenshot above:

Source2 (Pink) = 12V Power
Source1 (Yellow) = PERST

The signals are measured in Slot 0 at the U.2 connector where the NVMe UUT is connected.

The waveform above is generated with the following command:

```
sb_sdb -d 0 \
-f HP_PWREN -w 0 -g 0 \
-f HP_PWREN -w 1 -g 0
```

Longer power off times are achieved by increasing the **-g** time between the deassertion and the assertion as in the screen shot below:

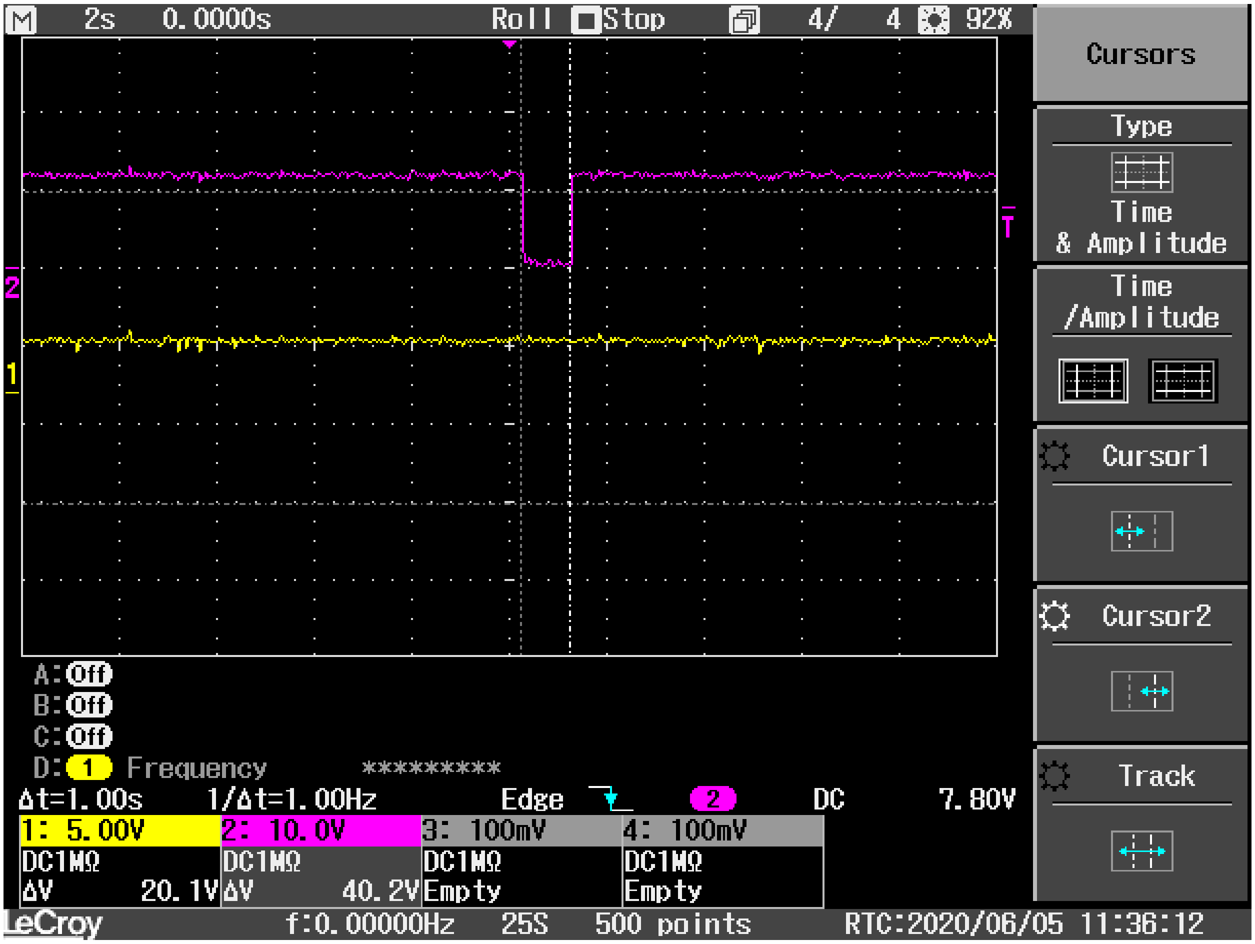


Figure 4: 1 Second Power Deassertion

```
sb_sdb -d 0 \
-f HP_PWREN -w 0 -g 830000 \
-f HP_PWREN -w 1 -g 0
```

The command above produces a 1S power deassertion.

Complex Power and Reset Timing

Using the primitives above, complex power and reset timing diagrams can be constructed. For example, power cycling followed by single or multiple PERST assertion. Some examples are given below.

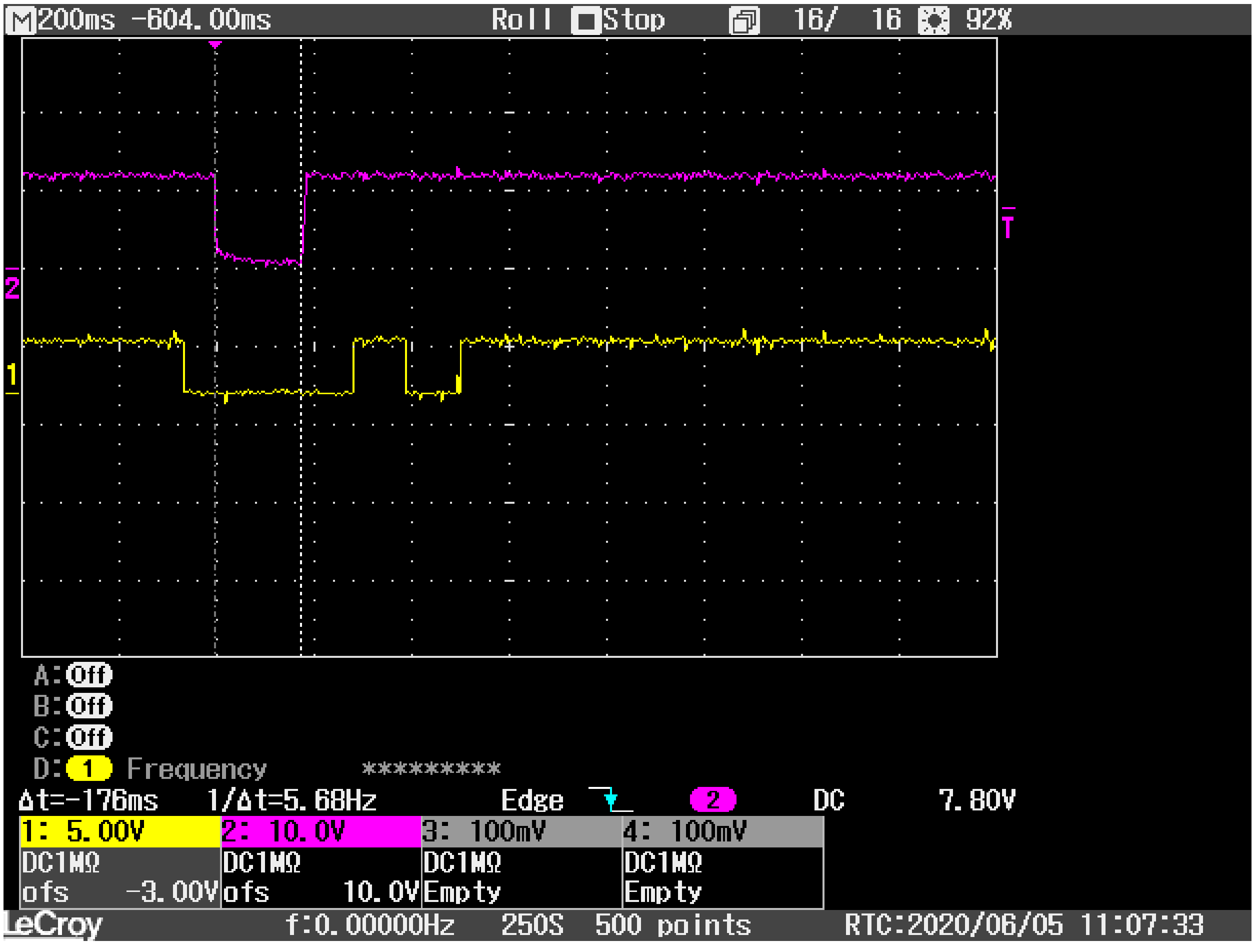


Figure 5: Power Deassert while PERST Asserted

In the scope screenshot above:

Source2 (Pink) = 12V Power
Source1 (Yellow) = PERST

The signals are measured in Slot 0 at the U.2 connector where the NVMe UUT is connected.

The waveform above is generated with the following command:

```
sb_sdb -d 0 \
-f HP_PERST_ -w 0 -g 50000 \
-f HP_PWREN -w 0 -g 0 \
-f HP_PWREN -w 1 -g 50000 \
-f HP_PERST_ -w 0 -g 200000 \
-f HP_PERST_ -w 1 -g 100000  \
-f HP_PERST_ -w 0 -g 100000  \
-f HP_PERST_ -w 1
```

For this example, PERST is asserted, then Power is removed from the device under test. Power is restored before PERST is released, and finally PERST is asserted and released again.

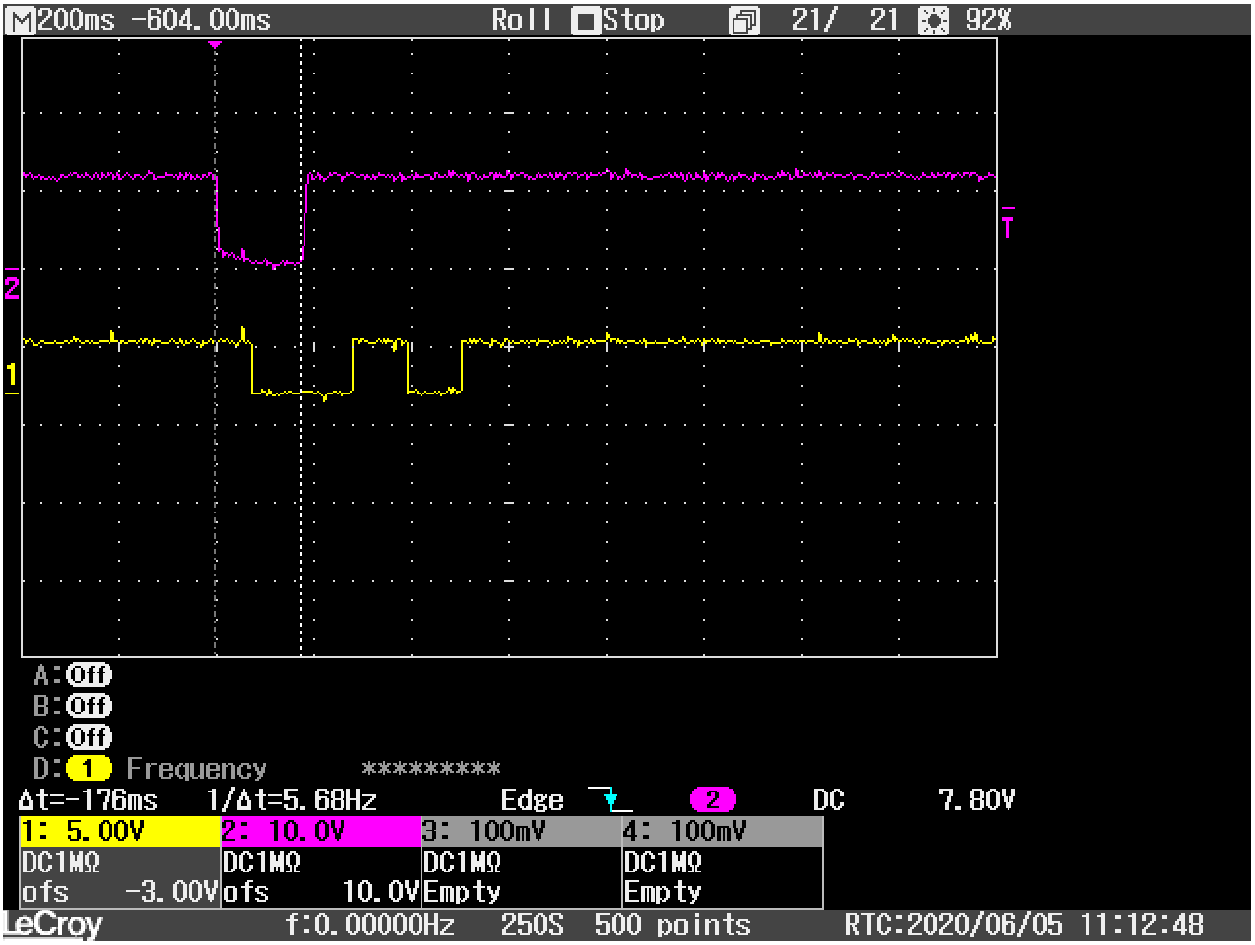


Figure 6: Power Asserted while PERST Asserted

In the scope screenshot above:

Source2 (Pink) = 12V Power
Source1 (Yellow) = PERST

The signals are measured in Slot 0 at the U.2 connector where the NVMe UUT is connected.

The waveform above is generated with the following command:

```
sb_sdb -d 0 \
-f HP_PWREN -w 0 -g 0 \
-f HP_PWREN -w 1 -g 50000 \
-f HP_PERST_ -w 0 -g 200000 \
-f HP_PERST_ -w 1 -g 100000 \
-f HP_PERST_ -w 0 -g 100000 \
-f HP_PERST_ -w 1
```

For this example, Power is deasserted, then PERST is asserted. Power is asserted then PERST is deasserted. Finally, PERST is asserted and released again.

Help Command

To view all of the commands and flags for the sb_sdb program, type **sb_sdb help** from the command line.

```
Info from the sb_sdb help

Error parsing input line arg count = 1, expected 1.  See help
  USAGE: sb_sdb [-n SYSTEM] [-d SLOT] [DEVICE (ex. /dev/ttyACM0)] ADDRESS
[DATA] [-b] [-B] [-T] [-v] [-f FEATURE] [-g uS] [-?]
     ex: sb_sdb -n 1 -d 0 6081C098 001E0002
     ex: sb_sdb -n 1 -d 0 60800000 b (return one byte)
     ex: sb_sdb -n 1 -d 0 0 (read from address at offset zero from base address
of slot 0)
     ex: sb_sdb -n 1 -d a 0FAC Read the 0xFAC register from all slots
   NOTE: DEVICE is optional.  Will be read from /tmp/NVMe/sdb_nr_N where N is
system number.  If none, will use first /dev/ttyACM* found
   NOTE: Write data must be four bytes example 001E0002
         Use -d SLOT to specify base register address for physical slot
         Address will be modified to point to the given slot base address
     ex: sb_sdb /dev/ttyACM0 0 -d 1 (will read register 0 for slot 1 at address
0x60844000
Options: -d SLOT   SBExpress-RM4 slot number.  Use '-d a' for ALL Slots
         -i        Info show all slot related information and TLP(0x0FAC),
DLLP(0x0FB0) and RCV(0x0BF4) Errors (hex)
         -l        Info show all slot related information and TLP(0x0FAC),
DLLP(0x0FB0), RCV(0x0BF4) and LTSSM (0x0BB0) counters (hex)
         -n        System number, default = 1
         -b        Single byte output and input
         -B        Also show binary register values
         -T        Temperature of the Atlas chip
         -v        Verbose
         -g        Time in uS to sleep between -f FEATURE commands
         -f        Feature to send to the HotPlug controller (See below)
         -?        Help
   NOTE: Reset counter notes:
         -r        Reset counters read by -i command
     ex: sb_sdb -d a -i -r (reset all counters all slots)
     ex: sb_sdb -d 1 -i -r (reset all counters slot 1)
     ex: sb_sdb -d a fac -r (reset TLP counter all slots)
     ex: sb_sdb -d 1 fac -r (reset TLP counter slot 1)
   NOTE: Hot Plug Controller Features:
         Commands are available to toggle Hot Plug features, currently
HP_PWREN, HP_CLKEN_, HP_PERST_ are available
         These signals can be asserted or deasserted as follows (note CLKEN_
and PERST_ are asserted low)
     ex: sb_sdb -d 0 -f HP_PERST_ -w 0 (device is now held in reset)
         A time to wait before issuing the next feature is specified with the
-g uS option.  For example, Assrt PERST_, wait 500uS, deassert PERST_
     ex: sb_sdb -d 0 -f HP_PERST_ -w 0 -g 500 -f HP_PERST_ -w 1
         Using combinations of time and signals, complex reset timing can be
achieved as explained in this whitepaper.
```

Summary

Power and reset type issues can be difficult to reproduce in the lab, and certain servers in the field have been found to have unspecified behavior with regards to Power and Reset timing.

The SBExpress platform and **sb_sdb** tool provide a simple means to reproduce complex timing sequences in a predictable manner reducing the time needed to replicate issues seen in the field and to produce repeatable test scenarios to be used during qualification of NVMe devices to avoid seeing issues in the field.

Contact SANBlaze for the latest SBExpress software and the *Certified by SANBlaze* test suite.