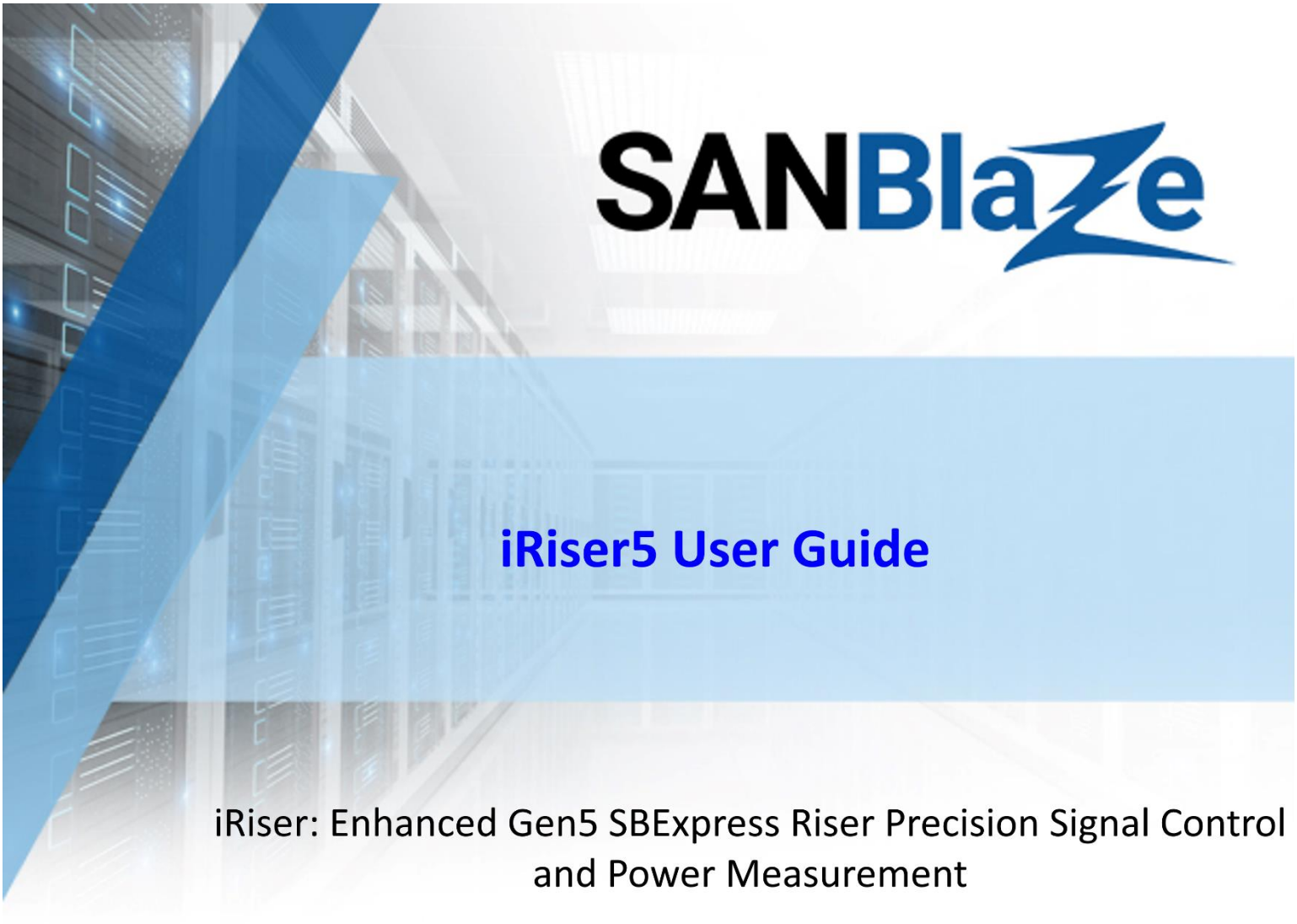


iRiser5 User Guide

iRiser: Enhanced Gen5 SBExpress Riser Precision Signal Control and Power Measurement

Author

Vince Asbridge

President, SANBlaze

Version V1.8 | July 2024

Document Versions

V0.1 - Original July 28th, 2023 – Created document – Vince Asbridge.
V0.2 - December 13th 2023 - Added information on mirroring support.
V0.3 – December 15th 2023 – Formatting (PB).
V0.4 - December 30th 2023 - Added init command to software and documented. (VA)
V0.5 - Updated Edit Action and document sequences and actions. Added FAQs. (VA)
V0.6 – Added SG and TM's info (PB)
V0.7 - Replaced pics w/ SG's new ones; removed the installing DT5 riser section (keep separate) (PB)
V0.8 – Added how dump is used (PB)
V0.9 – Vince's changes + extensive edits (PB)
V1.0 – Vince's new commands and other edits (PB)
V1.1 – Added SH and TM's edits (PB)
V1.2 – Added BK's clocking info (PB)
V1.3 – Added SG's edits (PB)
V1.4 – Added read and write command per TM; slot population per VA & SH (PB)
V1.5 – Changed FPGA from VC5 to VC8, added Showing Sequence Status and removed note (PB)
V1.6 – Made BK's changes for RM5 V2/V3 (A0) SBRs and updated table (PB)
V1.7 – Fixed verbiage on when sequence is a loop per TM (PB)
V1.8 – Updated install steps, revision ca, SBR image table, references to other docs. 7/8/24 (PB)

Table of Contents

Introduction

iRiser5 is a member of the iRiser family of precision NVMe test tools from SANBlaze. The iRiser family provides precision control of PCIe/NVMe power and control signals while continuously monitoring the power of each device under test (DUT).

The iRiser5 allows you to sample power at 1 million (M) samples per second, placing data in host memory with zero host CPU overhead.

A sequence of events can be scheduled on each signal line with up to 10 nanoseconds (nS) precision, each with intervals from 80nS to hours. Simple or complex sequences can be defined and loaded to the iRiser from the host system.

This document describes how to install iRiser cards into the SBExpress-DT5 chassis, building and loading sequences to the iRiser and executing them, and provides examples of typical sequences.

Overview of the Installation and Configuration Steps

The minimum software revision supporting the iRiser5 is 10.7. With this, the system software will need to be upgraded to 10.7 before installing the iRiser5 hardware.

The software upgrade requires the following software components to be upgraded in the following order:

1. Update the system software to the latest version of 10.7.
2. Verify that the Serial Boot Records (SBR) bridge firmware is upgraded to the latest, as shown in the Verifying the SBR Images section of the iRiser5 User Guide.
3. Reboot the machine and verify the Software and SBR versions, using the following commands:
   - **`grep Version= /proc/vlun/config`** to check the software version.
   - **`sb_flash show`** to check the SBR version.
4. Power off the machine using the `poweroff` command.
5. Prepare to install the iRiser5 into the RM5 or DT5 chassis, noting the following: the iRiser5 devices can be installed in Slots 6, 7, 8, or 9 of the SBExpress-RM5 for a total of four. The SBExpress-DT5 accepts three iRiser5 devices.
6. Install the iRiser5 into the appropriate RM5 or DT5 chassis. Refer to the SB-Express-RM5 or DT5 Installation Guides that came in your shipping box. You can also access them here.
7. Reboot the RM5 or DT5.
8. Verify the correct operation of the iRiser5, using the following commands:
   - **`lspci |grep SANBlaze`**
     - `iRiser5 is Device 2015, revision ca or higher`
     - `DT5 Motherboard is Device 2004`
     - `RM5 Motherboard is Device 2005`

   For example:

   `lspci |grep SANBlaze`

```
20:00.0 Signal processing management: SANBlaze Technology, Inc.
Device 2004 (rev ae) <- DT5

22:00.0 Signal processing management: SANBlaze Technology, Inc.
Device 2015 (rev ca) <- iRiser5

23:00.0 Signal processing management: SANBlaze Technology, Inc.
Device 2015 (rev ca) <- iRiser5
```

9. This completes the iRiser5 installation.

Updating and Configuring the DT5/RM5 System

This section describes how to update and configure the DT5/RM5 software and firmware for iRiser5 testing.

Updating the System Software

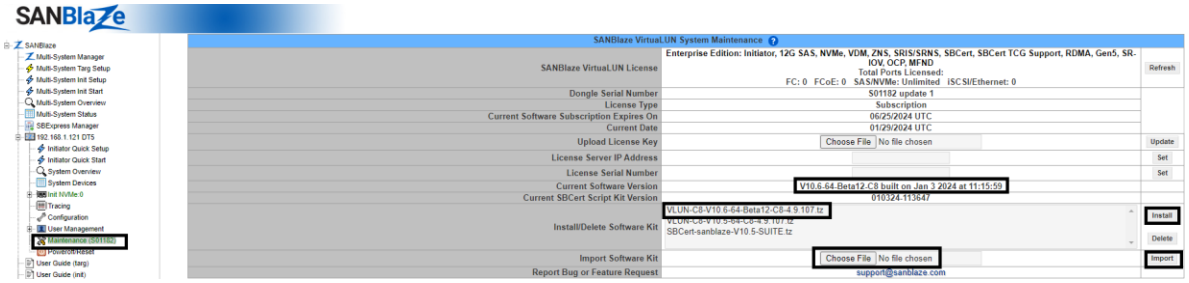
You must use 10.7 software with the iRiser5 hardware. As noted earlier, the iRiser5s are shipped with the latest software installed, no software update is required.

If you need to update your system software:

1. Download the share file provided to you by SANBlaze**.** This file has a **.tz** extension. For example:

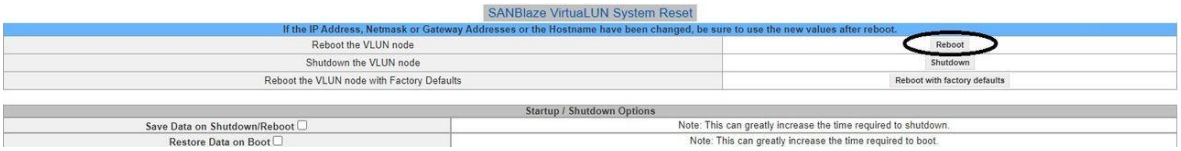
   `VLUN-C8-V10.7-64-Beta1-C8-4.9.107.tz`

2. Open the SANBlaze SBExpress-DT5 GUI by entering the **IP address** displayed on the front of your system into a web browser.
3. Within the GUI, select the **Maintenance** page located on the left-hand side menu, as shown below.



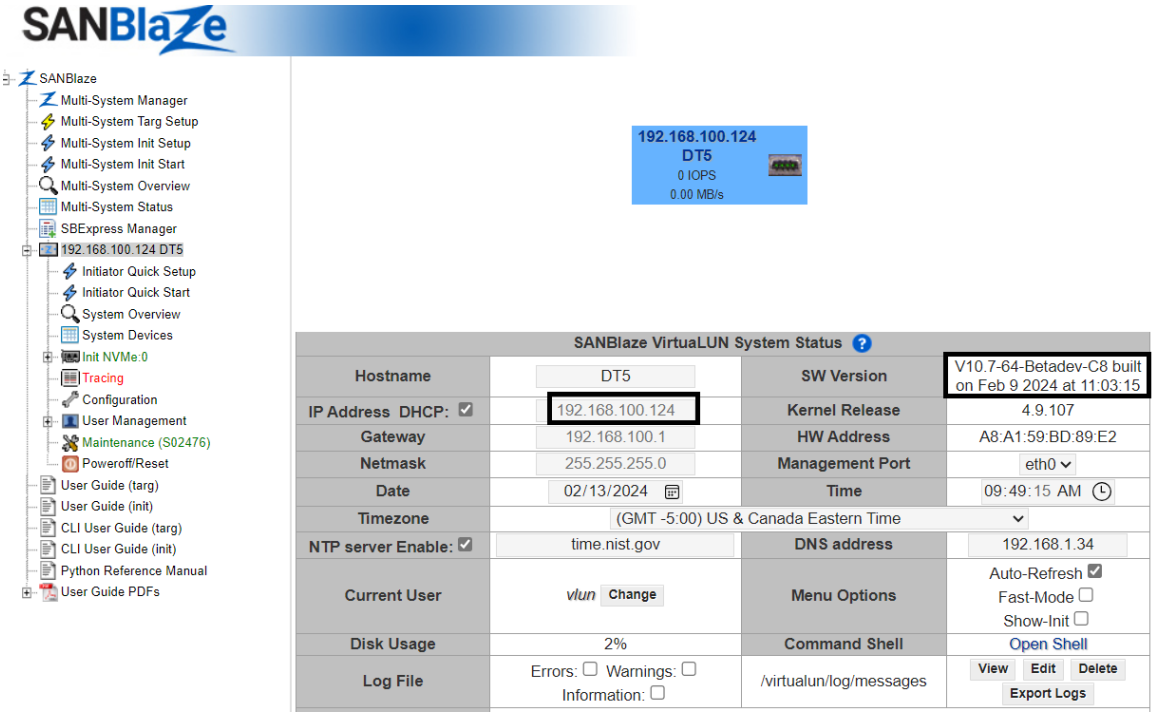
4. Within the Maintenance page, click the **Choose File** button located on the **Import Software Kit** row.
5. Select the **.tz** file provided to you by SANBlaze and then select **Import**. Importing the file may take a few minutes.
6. When the import has finished, highlight the new filename that is listed on the screen and select **Install**; a pop-up appears. Select **OK**. The install begins, this may take a few minutes.

7. Upon completion, a **SANBlaze VirtuaLUN System Reset** page appears. For example:



8. Click on the **Reboot** button. After the system has rebooted, the SBExpress page is displayed**,** displaying the current software version that is associated with the IP address of the system. In the GUI example below, the software version is shown on the top of the right-hand column from the IP address (left-hand column) on the main page. In this example, the software version is V10.7-64-Betadev-C8, your version may vary but must be Version 10.7 or later.

   NOTE: If you have Version 10.6 or lower on your system, the iRiser5 will not work, update to the current 10.7 version as previously described in this section.



9. To check the software version from the CLI, use the following command:

```
grep Version= /proc/vlun/config
```

Verifying the SBR Images

Using your telnet or ssh session, verify that the required Serial Boot Records (SBRs) are present on the system. The SBR images required for iRiser5 are shown in the following table.

RM5 V2/V3 (A0)	RM5 V2/V3 (B0)	RM5 V4	DT5	Description
03b60505	03b71009	03b72105	03c01016	Default
03B60201	03b71200	03b72204	03c02004	SRIS/SRNS

To determine your system version, use the following pair of commands:

```
sb_i2c2 -d -3 -e | grep Device
lspci | egrep "(DT|RM)[45]" -m 1
```

The last digit of the first command is the version number: i.e., 1, 2, 3, etc.

The last few characters of the second command will show if the rev is a0 or b0.

To display the SBR image configuration for the system use: **`sb_flash show`**

```
[root@DT5 ~]# sb_flash show
  INFO: Found 3c01016 flash image 0 at available_images 57
  INFO: Found 3c01016 flash image 1 at available_images 57
  INFO: Found 3c01016 flash image 2 at available_images 57
  INFO: Found 3c01016 flash image 3 at available_images 57
  INFO: System = DT5B0 sdb port=/dev/ttyACM0 Selected SBR Flash=0 Showing compatible images
     Index  ImageID  Type  Ver  Mode  UpLNK  SSC/CFC  Clock  System  Description
Active: 58 03c01016    10   16  Base     10      CFC     CC   DT5B0  DT5 Default
        60 03c01202    12   02  Base     10      CFC     CC   DT5B0  DT5 Default
        64 03c02004    20   04  Base     10      SSC   SRIS   DT5B0  DT5 SRIS
        66 03c02100    21   00  Base     10      CFC     CC   DT5B0  DT5 DPR Off; Dual Port
        67 03c02200    22   00  Base     10      CFC     CC   DT5B0  DT5 DPR Off; Single Port
  INFO: Current Active Images:
     Index  ImageID  Type  Ver  Mode  UpLNK  SSC/CFC  Clock  System  Description
Active:  0 03c01016    10   16  Base     10      CFC     CC   DT5B0  DT5 Default
         1 03c01016    10   16  Base     10      CFC     CC   DT5B0  DT5 Default
         2 03c01016    10   16  Base     10      CFC     CC   DT5B0  DT5 Default
         3 03c01016    10   16  Base     10      CFC     CC   DT5B0  DT5 Default
[root@DT5 ~]#
```

In the above example, the DT5 default **SBR 03c01016** is in eeprom index 0 and is active. This is the SBR image that is needed for the DT5. If your SBR image is not the default, follow the steps below.

If you want to upgrade the default SBR from **03c01016** to **03c01202**:

```
[root@DT5 ~]# sb_flash select 0 image 60
  INFO: Selecting flashindex=0
  INFO: Found 3c01016 flash image 0 at available_images 57
  INFO: Found 3c01016 flash image 1 at available_images 57
  INFO: Found 3c01016 flash image 2 at available_images 57
  INFO: Found 3c01016 flash image 3 at available_images 57
  INFO: System = DT5B0 sdb port=/dev/ttyACM0 Selected SBR Flash=0 Showing compatible images
  INFO: Selecting Flash Index=0, Image Index=59 Write File
filename=RDK_B0.RC.pre_15_Sblz_DT5_03C01202.signe
d.bin
  INFO: readFlash successfully read 2156 bytes
```

```
  INFO: Doing cmd=diff -q /tmp/firmware/0/write_test.sbr /tmp/firmware/0/
  write_test.sbr.unsigned
  INFO: Successfully uploaded Flash at Index=0 from file /virtualun/firmware/atlas/
  sbr_images/B0/DT5/RDK_B0.
RC.pre_15_Sblz_DT5_03C01202.signed.bin
  INFO: Found 3c01202 flash image 0 at available_images 59
  INFO: Found 3c01202 flash image 1 at available_images 59
  INFO: Found 3c01202 flash image 2 at available_images 59
  INFO: Found 3c01202 flash image 3 at available_images 59
  INFO: Current Active Images:
     Index  ImageID  Type  Ver  Mode  UpLNK  SSC/CFC  Clock  System  Description
Active:  0 03c01202    12   02  Base     10      CFC     CC   DT5B0  DT5 Default
         1 03c01202    12   02  Base     10      CFC     CC   DT5B0  DT5 Default
         2 03c01202    12   02  Base     10      CFC     CC   DT5B0  DT5 Default
         3 03c01202    12   02  Base     10      CFC     CC   DT5B0  DT5 Default
[root@DT5 ~]#
```

The word **Next** appears near the selected Default SBR.

1. After the Default SBR has been selected, power cycle the system using the following command:

   [root@DT5 ~**]#** **poweroff**

2. Wait 30 seconds, then turn the unit back on by pressing the checkmark button on the front of the DT5.

Verify that the Default SBR is Active

After the SBExpress-DT5 is back up and running, use the **sb_flash show** command to verify that the active SBR image 03c01016 is the Default SBR. For example:

```
[root@DT5 ~]# sb_flash show
  INFO: Found 3c01202 flash image 0 at available_images 59
  INFO: Found 3c01202 flash image 1 at available_images 59
  INFO: Found 3c01202 flash image 2 at available_images 59
  INFO: Found 3c01202 flash image 3 at available_images 59
  INFO: System = DT5B0 sdb port=/dev/ttyACM0 Selected SBR Flash=0 Showing compatible images
     Index  ImageID  Type  Ver  Mode  UpLNK  SSC/CFC  Clock  System  Description
Active: 58 03c01016    10   16  Base     10      CFC     CC   DT5B0  DT5 Default
        60 03c01202    12   02  Base     10      CFC     CC   DT5B0  DT5 Default
        64 03c02004    20   04  Base     10      SSC   SRIS   DT5B0  DT5 SRIS
        66 03c02100    21   00  Base     10      CFC     CC   DT5B0  DT5 DPR Off; Dual Port
        67 03c02200    22   00  Base     10      CFC     CC   DT5B0  DT5 DPR Off;
Single Port
  INFO: Current Active Images:
     Index  ImageID  Type  Ver  Mode  UpLNK  SSC/CFC  Clock  System  Description
  Next:  0 03c01202    12   02  Base     10      CFC     CC   DT5B0  DT5 Default
         1 03c01202    12   02  Base     10      CFC     CC   DT5B0  DT5 Default
         2 03c01202    12   02  Base     10      CFC     CC   DT5B0  DT5 Default
         3 03c01202    12   02  Base     10      CFC     CC   DT5B0  DT5 Default
  WARN: System is booted using unknown SBR image, next boot will select SBR flash 0
[root@DT5 ~]#
```

NOTE: Using **-3** in this command specifies that changes should be written to the motherboard.

You can issue any of the following commands to return the gpios back to their default settings:

```
iriser -d -3 set sw7
iriser -d -3c clear sw6
iriser -d -3 set sw2
iriser -d -3c clear sw1
```

Getting Started

This section describes:

- Identifying the slots containing an iRiser.
- Populating slots with the iRiser5
- Command Structure used.
- Displaying the iRiser current state of signals.
- Manually setting or clearing a control signal.
- How to Save, Store, and Share .cfg and .sh Files.
- Find iRisers on PCIe.
- Using Read and Write commands.
- Setting the GPIOs to Initial Default State.
- Writing the GPIO Registers as 32 bit Values.
- Setting I/O and GPIO Values from a File.
- Verify the Device Under Test is Linked Correctly.

Identifying Slots Containing an iRiser

Because iRiser5 can coexist with standard SANBlaze risers, use the following command to identify which slots contain an iRiser:

sb_i2c2 -d a -i

```
INFO: System 01[MB] SBExpress SN=64fc8ad9 Rev=R01 i2c=/dev/i2c-0 SDB=/dev/ttyACM0  VLUN=0
INFO:  Slot Main  MI  Priv   p   t   l    PCIe Prsnt Latch Width Speed Power BlueLED   RiserID
INFO:  ---- ---- ---- ---- --- --- --- ------- ----- ----- ----- ----- ----- ------- ---------
INFO:     0    0    1 none   0 100   1 09:00.0     1     0     4     5     1       1 600952000
INFO:     1    0    1 none   0 101   1 07:00.0     1     0     4     4     1       0 600952000
INFO:     2    0    1    3   0 102   1 05:00.0     1     0     4     4     1       0 600957003
```

In the previous example, the iRisers are located in slots 0 and 1, and are identified by the RiserID of 600952xxx.

Populating Slots with the iRiser5

The SBExpress-RM5 allows up to four iRiser5 devices and the SBExpress-DT5 allows three iRiser5 devices. For the SBExpress-RM5, place up to four iRiser5 devices in these slots only: 6, 7, 8, or 9 for the best signal integrity.

Command Structure

iRiser commands use the following structure:

```
iriser <-d N> <stop|start|dump|show|init> [action[s]|sequence[s]]
```

Where: <> = required and [] = optional

Note that the parameter *dump* prints the fpga registers and is mostly for internal use.

Help is available and is context sensitive to the -d # of the device:

```
iriser -d 0 help
```

Specific commands and examples are shown below.

Checking the Current State of Signals

The following command shows the names and state of the signals under iRiser control:

iriser -d 1 show gpio

```
00000204: 0ab900ff
Direction: I/O (UC) user writable signal, i/o (LC) signal locked
GPIO|31           24|23           16|15            8|7             0|
    |USR3TST|USR2TST|USR1TST|USR0TST|SBGPIO3|SBGPIO2|SBGPIO1|SBGPIO0|STATUSL|LEDEDSF|RFUEDSF|PLNEDSF|PLAEDSF|MFGEDSF|DUALPTL|PRSNTL|Tx3P|Tx3N|Tx2P|Tx2N|Tx1P|Tx1N|Tx0P|Tx0N|P1CLKL|P0CLKL|PWRDIS|SMRSTL|3V3|12V|CLKREQL|PERST0L|
    | O   | O   | O   | O   | I   | I   | I   | I   | I   | I   | I   | I   | I   | O   | I   | O   | O   | O   | O   | O   | O   | O   | O   | O   | O   | I   | I   | O   | O   | I   | I   | Dir:[0x214]0xf002ffcc
    | 0   | 0   | 0   | 0   | 1   | 1   | 1   | 1   | 0   | 1   | 0   | 1   | 0   | 0   | 1   | 0   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 0   | 0   | 0   | 1   | 1   | 1   | 0   | 1   | Val:[0x210]0x0f52ff1d
```

Alternative format for the same output:

iriser -d 1 show gpio2 (or gpio1)

```
Bit         Name Val Description
---         ---- --- -----------
00[I]    PERST0L   1 Setting low asserts perst
01[I]    CLKREQL   0 Setting low asserts clkreq
02[O]        12V   1 Setting low disables 12V
03[O]        3V3   1 Setting low disables 3V3
04[I]     SMRSTL   1 Setting low resets SMBus
05[I]     PWRDIS   0 Setting high sets PWRDIS signal to EDSFF
06[O]     P0CLKL   0 Setting low enables P0 clock
07[O]     P1CLKL   0 Setting low enables P1 clock
08[O]       Tx0N   1 Setting low disables Tx[0]-
09[O]       Tx0P   1 Setting low disables Tx[0]+
10[O]       Tx1N   1 Setting low disables Tx[0]-
11[O]       Tx1P   1 Setting low disables Tx[0]+
12[O]       Tx2N   1 Setting low disables Tx[0]-
13[O]       Tx2P   1 Setting low disables Tx[0]+
14[O]       Tx3N   1 Setting low disables Tx[0]-
15[O]       Tx3P   1 Setting low disables Tx[0]+
16[I]     PRSNTL   0 Setting low forces PRSNT, normally input from drive
17[O]    DUALPTL   1 Setting low forces Dual Port signal to drive
18[I]    MFGEDSF   0 EDSFF MFG signal
19[I]    PLAEDSF   0 EDSFF PLA (Power Loss Ack) normally input from drive
20[I]    PLNEDSF   1 EDSFF PLN (Power Loss Notification)
21[I]    RFUEDSF   0 EDSFF RFU (Reserved Future Use) pin
22[I]    LEDEDSF   1 EDSFF LED low enables LED
23[I]    STATUSL   0 Setting low enables blue status LED at front
24[I]    SBGPIO0   1 Spare GPIO wired to TP on DT5 "A" board
```

```
25[I]   SBGPIO1   1 Spare GPIO wired to TP on DT5 "A" board
26[I]   SBGPIO2   1 Spare GPIO wired to TP on DT5 "A" board
27[I]   SBGPIO3   1 Spare GPIO wired to TP on DT5 "A" board
28[O]   USR0TST   0 User monitor connector on iRiser5
29[O]   USR1TST   0 User monitor connector on iRiser5
30[O]   USR2TST   0 User monitor connector on iRiser5
31[O]   USR3TST   0 User monitor connector on iRiser5
```

Notes from the previous output:

For Direction (0x214), certain signals may be locked by SANBlaze. These cannot be changed or sequenced.

- O – A signal that can be sequenced.
- o – A signal that cannot be sequenced.
- 1 – Signal is "set".
- 0 – Signal is "clear".

Note also that reading and writing the iRiser register at 0x210 is the Setting Register for the GPIO; the actual signal value (which may differ from the SET value in the case of a wired or signal) can be read from register 0x200.

The following is an example of reading a SET value versus a Current IO value:

Set Value

```
iriser -d 1 0x210
00000210: ebfdffff
```

Current Value

```
iriser -d 1 0x200
00000210: ebfdffff
```

Manually Setting or Clearing a Signal Control

Each signal can be "set" or "cleared" by bit location or by signal name. Use either of the following syntax examples to set or clear a signal control:

- Setting by signal name (from the show GPIO table above):

  ```
  iriser -d 1 set dual001
  ```

- Clearing by bit location [0:31]

  ```
  iriser -d 1 clear 0
  ```

Note that signal names are not case sensitive. The commands above both operate on the same bit[0].

How to Save, Store, and Share .cfg and .sh Files

To save iRiser information to the current directory, store to a specified path, or share the .cfg and .sh files, use the following command:

```
iriser -d <to slot> -f /etc/iRiser/<system>/<from slot>/active.cfg
```

For example, if you want slot 1 to have the same active.cfg file as slot 2:

```
iriser -d 1 -f /etc/iRiser/<system>/2/active.cfg
```

Find iRisers on PCIe

To find an iRiser on PCIe, use the following command:

```
lspci |grep SAN
C5:00.0 Signal processing management: SANBlaze Technology, Inc. Device 2005 (rev 68)20:00.0
Signal processing management: SANBlaze Technology, Inc. Device 2004 (rev ac)
22:00.0 Signal processing management: SANBlaze Technology, Inc. Device 2015 (rev c6)
23:00.0 Signal processing management: SANBlaze Technology, Inc. Device 2015 (rev c6)
```

The device ID for iRiser5 is 2015 (rev c6)
The device ID for RM5 is 2005 (rev 68)
The device ID for DT5 is 2004 (rev ac)

The program "find_iRiser5_USB" can be used to locate the USB connection to the iRiser. This program is needed by the openocd flash procedure, which now finds the iRiser5 to program by slot/USB.

Using Read and Write Functions

You can write to a script using -s or write to a file using -f. Note that the read function however, can only be used on a file. For example:

```
[root@DT5 ~]# iriser -d 1 write -s testcfg.sh
```

This builds a new script *testcfg.sh* from the current configuration.

```
[root@VLUN-QA-100-104-IPMI-149 ~]# iriser -d 2 write -f testcfg.txt
```

Write user configuration file *testcfg.txt* and exit.

```
[root@VLUN-QA-100-104-IPMI-149 ~]# iriser -d 3 read -f testcfg.txt
```

Restores iRiser from configuration file *testcfg.txt* and activates.

Setting the GPIOs to Initial Default State

There are two commands which reset the GPIOs to the default state:

- init
- reset

To set the iRiser5 to the factory default state, use the init command.

iriser -d <slot> init

This:

- Stops the sequencer at <slot> if running.
- Writes the default value to the GPIO output register at 0x210.
- Writes the default value to the GPIO direction register at 0x214.

- Clears the sequence memory to zero.

After the init command is used, the drive under test should re-link on PCIe and power up ready and the sequencer is stopped.

iriser -d <slot> reset

Note: The reset command is not fully functional yet, please refrain from using it at this time.

To set the GPIOs to default state but leave the iRiser5 sequencer memory intact, use the reset command.

This:

- Stops the sequencer at <slot> if running.
- Writes the default value to the GPIO output register at 0x210.
- Writes the default value to the GPIO direction register at 0x214.
- Leaves the sequencer memory as currently programmed.

After the reset command is used, the drive under test should re-link on PCIe and power up ready and the sequencer is stopped.

You can verify the current status at any time withcthe following command:

```
iriser -d N show gpio
```

Writing the GPIO Registers as 32 bit Values

You can set the I/O direction register (at 0x214) as a 32 bit write, as follows:

```
iriser -d 1 214 0x00000000
```

Where zero is input, 1 is output. The command above sets all GPIOs to input and is a good debug tool to prove the hardware sets the correct default state.

You can set the I/O Value register (at 0x210) as a 32 bit write, as follows:

```
iriser -d 1 210 0xffffffff
```

Where zero is clear, 1 is set. The command above sets all GPIOs to level 1.

The above commands will render the device under test (DUT) disconnected. You can use the init command to recover any unwanted configuration.

NOTE: Some GPIOs are masked off by SANBlaze and are not settable by the user.

Setting I/O and GPIO Values from a File

An example config file exists under */etc/iRiser* containing the default state for each FPGA based design and can be loaded with the following command:

```
iriser -d 1 -f /etc/iRiser/iRiser5_GPIO_default.cfg
```

Your GPIOs should now look like this:

```
iriser -d 1 show gpio
```

```
Direction: I/O (UC) user writable signal, i/o (LC) signal locked
GPIO|31           24|23           16|15            8|7             0|
    |USR3TST|USR2TST|USR1TST|USR0TST|SBGPIO3|SBGPIO2|SBGPIO1|SBGPIO0|STATUSL|LEDESDSF|RFUEDSF|PLNEDSF|PAEDSF|MFGEDSF|DUALPTL|PRSNTL|Tx3P|Tx3N|Tx2P|Tx2N|Tx1P|Tx1N|Tx0P|Tx0N|P1CLKL|P0CLKL|PWRDIS|SMRSTL|3V3|12V|CLKREQL|PERST0L|
    |O  O  O  O  I  I  I  I  I  I  I  I  I  I  O  I  O  O  O  O  O  O  O  O  O  O  I  I  O  O  I  I  Dir:[0x214]0xf002ffcc
    |0  0  0  0  1  1  1  1  0  1  0  1  0  0  1  0  1  1  1  1  1  1  1  1  0  0  0  1  1  1  0  1  Val:[0x210]0x0f52ff1d
```

Because all are still in the default state, none of the above should have affected the connection to the device.

Verifying the DUT is Linked Correctly

Use the following command to verify that the DUT is linked correctly:

```
sb_sdb -d 1 -5 -l
```

```
Slot PSN(hex) PCIeDev Pres MRL Wid Spd Pwr 0x0FAC(TLP) 0x0FB0(DLLP) 0x0BF4(RERR) 0x0BC4(RCVY)
0x0BB0(LTSSM) State.Substate
   1   4(x04) 07:00.0    1   0   4   4   1           0            0            0            0
1301h L0.Active
Time since counter reset 02:30:23
```

Description of the GPIO Signals for iRiser5

Signal names should always be checked by the **iriser show gpio** command. Currently, iRiser5 signals are named as described below, but may change during qualification. Bit numbers are fixed in hardware but may vary between designs and revisions.

All signals below are accessible by the sequencer and can be changed in 80nS intervals. This will be described later in this document. You can manually set or clear any bit by number or name using the following command:

```
iriser -d N <name || bit> <set || clear>
```

Showing the State of a Signal

GPIO bits can be inspected by name or bit number using the `show` command:

```
iriser -d 1 show perst
200:[00] 1

iriser -d 1 show 0
200:[00] 1

iriser -d 1 show perst -q
1

iriser -d 1 show 12v
200:[02] 1

iriser -d 1 show 2
200:[02] 1

iriser -d 1 show 12v -q
1
```

In the output above, the syntax used is:

```
input Reg 0x200:[bit] value
```

Adding the -q (quiet) parameter will return only the state of the bit and may be more helpful for scripting.

show <signal> -qq

Using the -qq flag to the show signal command will suppress all output and exit with 0, if the signal is clear or 1 if the signal is set. The exit value can then be used by the caller. Examples:

```
iriser -d 1 show 12v -qq

echo $?
1

iriser -d 1 show 12v -qq && echo signal is clear || echo signal is set
signal is set

iriser -d 1 clear 12v

iriser -d 1 show 12v -qq && echo signal is clear || echo signal is set
signal is clear
```

Signals Available to Control

The signals under control are as follows:

```
iriser -d 1 show gpio2

Bit          Name Val Description
---          ---- --- -----------
00[I]     PERST0L   1 Setting low asserts perst
01[I]     CLKREQL   0 Setting low asserts clkreq
02[O]         12V   0 Setting low disables 12V
03[O]         3V3   1 Setting low disables 3V3
04[I]      SMRSTL   1 Setting low resets SMBus
05[I]      PWRDIS   0 Setting high sets PWRDIS signal to EDSFF
06[O]      P0CLKL   0 Setting low enables P0 clock
07[O]      P1CLKL   0 Setting low enables P1 clock
08[O]        Tx0N   1 Setting low disables Tx[0]-
09[O]        Tx0P   1 Setting low disables Tx[0]+
10[O]        Tx1N   1 Setting low disables Tx[0]-
11[O]        Tx1P   1 Setting low disables Tx[0]+
12[O]        Tx2N   1 Setting low disables Tx[0]-
13[O]        Tx2P   1 Setting low disables Tx[0]+
14[O]        Tx3N   1 Setting low disables Tx[0]-
15[O]        Tx3P   1 Setting low disables Tx[0]+
16[I]      PRSNTL   0 Setting low forces PRSNT, normally input from drive
17[O]     DUALPTL   1 Setting low forces Dual Port signal to drive
18[I]     MFGEDSF   0 EDSFF MFG signal
19[I]     PLAEDSF   0 EDSFF PLA (Power Loss Ack) normally input from drive
20[I]     PLNEDSF   1 EDSFF PLN (Power Loss Notification)
21[I]     RFUEDSF   0 EDSFF RFU (Reserved Future Use) pin
22[I]     LEDEDSF   1 EDSFF LED low enables LED
23[I]     STATUSL   0 Setting low enables blue status LED at front
24[I]     USBDISL   1 Spare GPIO wired to TP on DT5 "A" board
25[I]     SB0GPIO   1 Spare GPIO wired to TP on DT5 "A" board
```

```
26[I]     SB1GPIO   1 Spare GPIO wired to TP on DT5 "A" board
27[I]     SB2GPIO   1 Spare GPIO wired to TP on DT5 "A" board
28[O]     USR0TST   1 User monitor connector on iRiser5
29[O]     USR1TST   0 User monitor connector on iRiser5
30[O]     USR2TST   0 User monitor connector on iRiser5
31[O]     USR3TST   0 User monitor connector on iRiser5
```

Setting GPIO Direction and Value

You can set any bit with the name of the signal, and you only need enough text to be unique. You can set the direction or the value of each bit. Note that the case used is insensitive.

Also, if an action sequence is running, the GPIOs are under control of the action sequence and may change.

Use the `show gpio` command to read signal names and determine if the sequencer is running or stopped. For example:

```
iriser -d 1 show gpio
```

GPIO	31							24	23							16	15							8	7							0	
	USR3TST	USR2TST	USR1TST	USR0TST	SB2GPIO	SB1GPIO	SB0GPIO	USBDISL	STATUSL	LEDEDSF	RFUEDSF	PLNEDSF	PLAEDSF	MFGEDSF	DUALPTL	PRSNTL	Tx3P	Tx3N	Tx2P	Tx2N	Tx1P	Tx1N	Tx0P	Tx0N	P1CLKL	P0CLKL	PWRDISL	SMRSTL	3V3	12V	CLKREQL	PERST0L	
Dir:[0x214]0xfe02ffcd	O	O	O	O	O	O	O	I	I	I	I	I	I	I	O	I	O	O	O	O	O	O	O	O	O	O	I	I	O	O	I	O	
Val:[0x200]0x3f52ff1d	0	0	1	1	1	1	1	1	0	1	0	1	0	0	1	0	1	1	1	1	1	1	1	1	0	0	0	1	1	1	0	1	

```
Mirroring GPIO: perst0l[0]->usr0tst[28] 12v[2]->usr1tst[29]
```
Sequencer: Stopped

Signal Access by Name

```
iriser -d 1 out usr0
iriser -d 1 set usr0
iriser -d 1 clear usr0
iriser -d 1 in usr0
```

Signal Access by Bit Location

```
iriser -d 1 out 28
iriser -d 1 set 28
iriser -d 1 clear 28
iriser -d 1 in 28
```

Note: While it may seem easier to use bit number, SANBlaze recommends that you use names when scripting because the bit locations may change from one design to the next, whereas the names will stay consistent.

Measuring Power

There are two distinct ways iRiser5 measures power:

- Backward Compatible Power Measurement
- Power Measurement via AtoD and DMA to Host Memory

Backward Compatible Power Measurement

For backward compatibility, the iRiser5 contains the same power circuitry as the 959 EDSFF riser. Power currently displayed on the GUI and recorded in the .csv files is based on the INA226 I2C AtoD and is accessed with the following command:

```
sb_i2c -d 0 -m
```

```
INFO: System 01[00] SBExpress SN=64fc8ad9 Rev=R01 i2c_main=i2c-0 i2c_mi=i2c-1 SDB=/dev/ttyACM0
INFO: 01[00] Vload=11896.250mV, Iload=310.362mA, Pload=3692.142mW
```

Power Measurement via AtoD and DMA to Host Memory

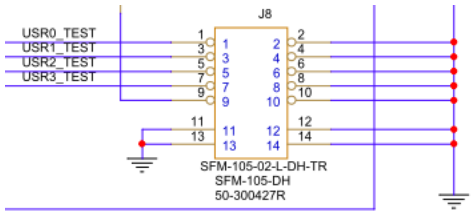
The iRiser5 can sample power 1M times/second or average up to 1M samples. The software to support this feature is under development and will be released and documented in a future software release.

Mirroring GPIO Bits

USR0TST - USR3TST can be mapped to any other GPIO signal for the purpose of connecting a scope to monitor the signals or to use any signal as a trigger. Up to four signals can be "mirrored" in this manner.

J8 - User Signal Connector

A ten-pin connector is available on the front of iRiser5 for the purpose of accessing the four mirror signals. The connector is wired as follows:



- Pin1 - USR0TST GPIO28
- Pin3 - USR1TST GPIO29
- Pin5 - USR2TST GPIO30
- Pin7 - USR3TST GPIO31
- Pin9 - N/C (no connect)
- All other pins are ground

A cable is included in the iRiser5 kit for access to these signals.

Mirroring Example:

A scope is placed on USR0 and USR1 which is used to monitor POWER (12V) and PCIe RESET (PERST).

Mirroring can be set up using bit location or name, and the current configuration is displayed using the show gpio command. For example:

show gpio

GPIO	31							24	23							16	15							8	7							0	
	USR3TST	USR2TST	USR1TST	USR0TST	SB2GPIO	SB1GPIO	SB0GPIO	USBDISL	STATUSSL	LEDEDSF	RFUEDSF	PLNEDSF	PLAEDSF	MFGEDSF	DUALPTL	PRSNTL	Tx3P	Tx3N	Tx2P	Tx2N	Tx1P	Tx1N	Tx0P	Tx0N	P1CLKL	P0CLKL	PWRDISL	SMRSTL	3V3	12V	CLKREQL	PERST0L	
	O	O	O	O	I	I	I	I	I	I	I	I	I	I	O	I	O	O	O	O	O	O	O	O	O	O	I	I	O	O	I	I	Dir:[0x214]0xf002ffcc
	0	0	0	0	1	1	1	1	0	1	0	1	0	0	1	0	1	1	1	1	1	1	1	1	0	0	0	1	1	1	0	1	Val:[0x200]0x0f52ff1d

In the above configuration, no mirroring has been established.

To set up the example of mirroring PERST to USR0TST and 12V to USR1TST, use the following commands (assumes iRiser5 is in slot 1):

iriser -d 1 mirror perst usr0

```
INFO: Mirror[0] from perst0l[0] to usr0tst[28]
```

iriser -d 1 mirror 12v usr1

```
INFO: Mirror[1] from 12v[2] to usr1tst[29]
```

Issue the show GPIO command to inspect the mirror configuration:

iriser -d 1 show gpio

GPIO	31							24	23							16	15							8	7							0	
	USR3TST	USR2TST	USR1TST	USR0TST	SB2GPIO	SB1GPIO	SB0GPIO	USBDISL	STATUSSL	LEDEDSF	RFUEDSF	PLNEDSF	PLAEDSF	MFGEDSF	DUALPTL	PRSNTL	Tx3P	Tx3N	Tx2P	Tx2N	Tx1P	Tx1N	Tx0P	Tx0N	P1CLKL	P0CLKL	PWRDISL	SMRSTL	3V3	12V	CLKREQL	PERST0L	
	O	O	O	O	I	I	I	I	I	I	I	I	I	I	O	I	O	O	O	O	O	O	O	O	O	O	I	I	O	O	I	I	Dir:[0x214]0xf002ffcc
	0	0	1	1	1	1	1	1	0	1	0	1	0	0	1	0	1	1	1	1	1	1	1	1	0	0	0	1	1	1	0	1	Val:[0x200]0x3f52ff1d

Mirroring GPIO: perst0l[0]->usr0tst[28] 12v[2]->usr1tst[29]

unmirror Command

To remove the mirroring, issue the unmirror command:

iriser -d 1 unmirror

Monitoring GPIO Signals with Mirroring

Once mirroring is established between the USRnTST signals and any of the lower 28 GPIO bits, a change to the "from" bit will be reflected by the "to" bit for the purpose of externally monitoring that signal.

Sequences and Actions

The iRiser5 is programmed by means of loading Actions and Sequences of actions to the device which are then executed changing the specified GPIO signals. Sequences are simply defined as groups of actions that are linked together and either end in a "stop" (sequence will run once and stop) or the index of another action. If an action points to the first action in the sequence, the sequence will loop until the "stop" command is issued at the CLI.

If the sequence is a loop and the system reboots, after the system comes back up, the sequence will continue running. The blue status LED on the front of the iRiser5 will be blinking. If the sequence is not a loop, after it stops running, the blue status LED on the front of the iRiser5 will stop blinking.

There are 128 "slots", each of which can hold an action, and a Sequence can be one or more actions; up to 128 actions can be defined in a sequence.

Examples of a Sequence of Actions

To illustrate the use case of Actions and Sequences, see the two example test scenarios below. Note the following:

s = seconds
ms = milliseconds
us = microseconds
ns = nanoseconds

Example 1: Run Once and Stop Sequence

In this example, a user wants to see the effect of a glitch on PERST (reset) on the device under test exactly 5S after a clean power up. You may want to describe the following actions:

- Action 0 - Starting state: Deassert 12V Power, assert PERST, sleep 10 seconds
- Action 1 - Assert 12v Power, sleep 250mS
- Action 2 - Deassert PERST, sleep 5S
- Action 3 - Assert PERST, sleep 100nS
- Action 4 - Deassert PERST, and stop

Example 2: Run until Stopped (loop)

In this example, a user wants to repeat the test created above, but run it continuously every 15 seconds until stopped. Add an additional action to the actions above that loops back to Action 0.

- Action 0 - Starting state: Deassert 12V Power, assert PERST, sleep 10 seconds
- Action 1 - Assert 12v Power, sleep 250mS
- Action 2 - deassert PERST, sleep 5S
- Action 3 - assert PERST, sleep 100nS
- Action 4 - deassert PERST, sleep 15S then return to Action 0

Defining Actions from CLI Command

The two examples above are built using the 'iriser -d <slot> edit action <action>' command, which can be run from a single CLI command or interactively. First build the first example using the CLI command and then modify it interactively to demonstrate both methods.

Start with the iRiser5 in Initial State:

```
iriser -d 0 init
```

Program the 5 actions described for Example 1:

```
#
# Commands to configure iRiser5 sequencer
#
iriser -d 0 edit action   0 0xf002ffcd 0x0f52ff18 10.000000s      1 Deassert 12V Power assert PERST sleep 10 seconds
iriser -d 0 edit action   1 0xf002ffcd 0x0f52ff1c 250.000000ms    2 Assert 12v Power sleep 250
iriser -d 0 edit action   2 0xf002ffcd 0x0f52ff1d 5.000000s       3 Deassert PERST sleep 5S
iriser -d 0 edit action   3 0xf002ffcd 0x0f52ff1c 100.000000ns    4 Assert PERST sleep 100nS
iriser -d 0 edit action   4 0xf002ffcd 0x0f52ff1d 100.000000ns stop Deassert PERST sleep 100nS and stop sequencer
```

iRiser -d 0 Edit Action

The iRiser5 is designed to be able to control signals to a device under test at precise intervals. For example, an action sequence could enable power to a device, 100mS later release PERST, and then 100mS after that assert PERST for 100nS and release it. Any combination of controlled signals can be programmed in this manner.

```
iRiser -d <slot> edit action <action number> <direction> <value> <time>
<next action> [name]
```

Where:

- <slot> - Physical slot number of the iRiser.
- <action number> - Action number, use show actions for list.
- <direction> - 32 bits of GPIO direction 1=output,0=input (use show gpio for signal names).
  - Certain I/O direction bits may be locked by SANBlaze depending on configuration.
- <value> - 32 bits of GPIO signal values.
  - All signals MUST be defined, for a signal that will not change on this action you must specify same value as the preceding action.
- <time> - Time specified as s (seconds), ms (milliseconds), us (microseconds), ns (nanoseconds) (e.g. 25.2ms).
  - Minimum time between actions is 80nS.

- <next> - Next action in the sequence.
  - If next = Usually current action number +1 - Continue execution at that action.
  - If next = "stop" - Stop execution (values in GPIO will remain at last action values).
  - Valid range [0 - 127].
  - If next points to existing action execution will jump to that action.
  - If next points to an action within the same sequence, will begin looping.
- <name> - Optional name for action. These are stored in the .cfg and .sh files described elsewhere in this document. Special characters are only allowed in the action name if you use quotation marks. Currently, 126 characters is the limit in length.
  - Optional, if not provided will be assigned to "Action <action number>".

Initializing the Action Memory

All actions can be cleared with a single command.

CAUTION - Use with caution.

```
iRiser -d 1 init
```

For the scenario described above, start with power disabled and PERST enabled and wait 1s.

Show Current Actions

Once the actions in Example 1 above have been programmed, you may show any one action, all the actions, or all the actions in a given sequence with the following commands:

```
iriser -d <slot> show action <number>
iriser -d <slot> show actions
iriser -d <slot> show sequence <number>
iriser -d <slot> show sequences
```

For Example 1:

```
  INFO: Action 0 Sequence 0
Direction: I/O (UC) user writable signal, i/o (LC) signal locked
GPIO|31           24|23           16|15            8|7             0|

     |U|U|U|U|S|S|S|U|S|L|R|P|P|M|D|P|T|T|T|T|T|T|T|T|P|P|P|S|3|1|C|P|
     |S|S|S|S|B|B|B|S|T|E|F|L|L|G|U|R|x|x|x|x|x|x|x|x|1|0|W|M|V|2|L|E|
     |R|R|R|R|2|1|0|B|A|D|U|N|A|E|A|S|3|3|2|2|1|1|0|0|C|C|R|R|3|V|K|R|
     |3|2|1|0|G|G|G|D|T|E|N|E|E|D|L|N|P|N|P|N|P|N|P|N|L|L|D|S| | |R|S|
     |T|T|T|T|P|P|P|I|U|D|E|E|D|G|P|T| | | | | | | | |K|K|I|T| | |E|T|
     |S|S|S|S|I|I|I|S|S|S|D|D|S|E|T|L| | | | | | | | |L|L|S|L| | |Q|0|
     |T|T|T|T|O|O|O|L|L|F|F|F|F|D|L| | | | | | | | | | | | | | | |L|L|
     |O|O|O|O|I|I|I|I|I|I|I|I|I|O|I|O|O|O|O|O|O|O|O|O|I|I|O|O|I|O| Dir:[0x804]0xf002ffcd
Act  |                                                               | Next|GPIO Value|Time
000  |0|0|0|0|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|0|0|0|  001|0x0f52ff18|10.000000s
001  |0|0|0|0|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|1|0|0|  002|0x0f52ff1c|250.000000ms
002  |0|0|0|0|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|1|0|1|  003|0x0f52ff1d|5.000000s
003  |0|0|0|0|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|1|0|0|  004|0x0f52ff1c|100.000000ns
004  |0|0|0|0|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|1|0|1|  004|0x0f52ff1d|100.000000ns
Stop
```

Starting a Sequence

Sequences are groups of actions that terminate (stop) or loop (jump back to start). Sequence numbers begin at zero and are assigned automatically. You cannot re-number sequences. To find all currently programmed sequences, use the show sequences command.

Start the sequence defined in Example 1 above with the command:

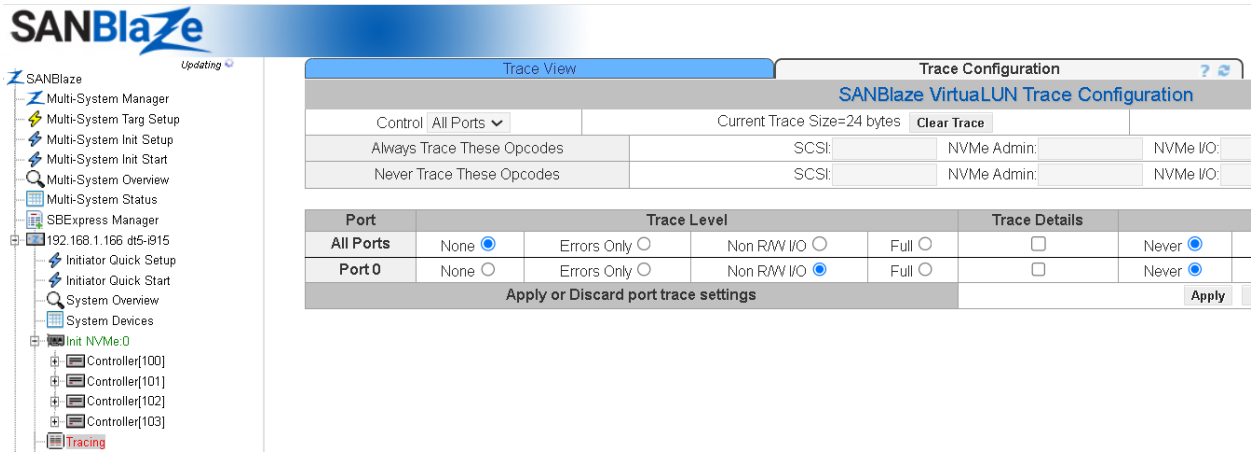
```
iRiser -d 0 start sequence 0
```

Because the sequence is set to stop, using the show sequence 0 command will first show the sequence as running, and then stopped.

What happens while the sequence is running?

Monitoring your device using the SANBlaze built-in "tracing" will show the NVMe activity as the sequence executes.

Monitor Activity with Tracing




```
Jan 22 05:40:05.693219 P0:T100: 0000:09:00.0: trigger with error 09061ae3
Jan 22 05:40:05.693222 P0:T100: 0000:09:00.0: trigger with error 09001ae3
Jan 22 05:40:05.693223 P0:T100: 0000:09:00.0: Device is gone, deleting
Jan 22 05:40:05.693224 P0:T100: controller gone reported
Jan 22 05:40:05.715258 P0:T100: logout with 0000:09:00.0
Jan 22 05:40:16.015336 P0: adding NVMe device 0000:09:00.0
Jan 22 05:40:16.117096 P0:T100: 0000:09:00.0: reset start
Jan 22 05:40:16.453719 P0:T100: 0000:09:00.0: Controller Probe to Controller Enable took 102265 usec
Jan 22 05:40:16.453732 P0:T100: 0000:09:00.0: Controller Enable (CC.EN=1) to Controller Ready (CSTS.RDY=1) took 336310 usec
Jan 22 05:40:16.453995 P0:T100: qid:0 cid:14a IdentifyController
Jan 22 05:40:16.454196 P0:T100: qid:0 cid:14a IdentifyController success (192 usec)
Jan 22 05:40:16.454201 P0:T100: 0000:09:00.0: Controller Ready to Identify Successful took 484 usec
Jan 22 05:40:16.454237 P0:T100: qid:0 cid:14a SetFeatures NSID=0x0 FID=0xe SV=0x0
Jan 22 05:40:16.454265 P0:T100: qid:0 cid:14a SetFeatures NSID=0x0 FID=0xe SV=0x0 success, result 0h (23 usec)
Jan 22 05:40:16.454269 P0:T100: qid:0 cid:14a GetFeatures NSID=0x0 FID=0x7 SEL=0x0
Jan 22 05:40:16.454302 P0:T100: qid:0 cid:14a GetFeatures NSID=0x0 FID=0x7 SEL=0x0 success, result 7F007Fh (23 usec)
Jan 22 05:40:16.454305 P0:T100: qid:0 cid:14a SetFeatures NSID=0x0 FID=0x7 SV=0x0
Jan 22 05:40:16.454331 P0:T100: qid:0 cid:14a SetFeatures NSID=0x0 FID=0x7 SV=0x0 success, result 7F007Fh (17 usec)
Jan 22 05:40:16.457972 P0:T100: qid:0 cid:14a CreateCompletionQueue QID=0x1 QSIZE=0x3fff
```

Monitoring Activity with sb_logger

A CLI program sb_logger is provided with all SANBlaze systems, which continuously monitors all relevant log files on the system.

It is common (and best) practice to leave the sb_logger program running in a CLI window while executing sequences to see how the sequence is affecting your device.

The following is an example of sb_logger session:

sb_logger

```
==> /virtualun/log/iriser <==
Mon Jan 22 05:45:31 2024:   INFO: Starting sequencer at action location action=0

==> /virtualun/log/sb_i2c2 <==
Mon Jan 22 05:45:31 2024: sb_i2c2 -d 0 -f STATUS_BLUE_LED_L -w 3 -q

==> /var/log/messages <==
Jan 22 05:45:31 dt5-i915 kernel: [81541.435244] nvme: DPC: link to bus 09 is now DOWN
Jan 22 05:45:31 dt5-i915 kernel: [81541.435250] nvme: HP: Slot(0): status 0108/4001 for bus 09
Jan 22 05:45:31 dt5-i915 kernel: [81541.435252] nvme: HP: Slot(0): Presence Not Detected on bus
09
Jan 22 05:45:31 dt5-i915 kernel: [81541.435253] nvme: HP: Slot(0): Link Down on bus 09
Jan 22 05:45:31 dt5-i915 kernel: [81541.435258] nvme: DPC status 0009 for bus 09
Jan 22 05:45:31 dt5-i915 kernel: [81541.435267] nvme: HP: Slot(0): link to bus 09 is DOWN
Jan 22 05:45:31 dt5-i915 kernel: [81541.435270] nvme_vlun_hp_down@15229: 0000:04:08.0
0000:09:00.0
Jan 22 05:45:31 dt5-i915 kernel: [81541.435272] nvme_vlun_hp_remove@15212: echo 1 >
/sys/bus/pci/devices/0000:09:00.0/remove
Jan 22 05:45:31 dt5-i915 kernel: [81541.435291] remove PCI device 0000:09:00.0 start
Jan 22 05:45:31 dt5-i915 kernel: [81541.435292] pci_stop_bus_device@101: 0000:09:00.0:
ffff88107241d800 0000:09
Jan 22 05:45:31 dt5-i915 kernel: [81541.435297] nvme: 0000:09:00.0: code 6
Jan 22 05:45:31 dt5-i915 kernel: [81541.435304] nvme_remove@7993: 0000:09:00.0: cfg rd 0004 ffff
Jan 22 05:45:31 dt5-i915 kernel: [81541.435305] nvme_remove@7994: 0000:09:00.0: cfg rd 0006 ffff
Jan 22 05:45:31 dt5-i915 kernel: [81541.435312] 0000:09:00.0: trigger with error 09061ae3
Jan 22 05:45:31 dt5-i915 kernel: [81541.435314] 0000:09:00.0: trigger with error 09001ae3
Jan 22 05:45:31 dt5-i915 kernel: [81541.435315] 0000:09:00.0: Device is gone, deleting
Jan 22 05:45:31 dt5-i915 kernel: [81541.435317] pci 0000:09:00.0: vendor/device ffffffff
Jan 22 05:45:31 dt5-i915 kernel: [81541.435321] nvme_dev_dis@6091: 0000:09:00.0: rd 1c ffffffff
Jan 22 05:45:31 dt5-i915 kernel: [81541.435531] nvme_disable_admin_queue@4253: 0000:09:00.0: rd
00 ffffffffffffffff
Jan 22 05:45:31 dt5-i915 kernel: [81541.435552] nvme_pci_disable@6055: 0000:09:00.0: cfg rd 0004
ffff
Jan 22 05:45:31 dt5-i915 kernel: [81541.456799] nvme: removing device 0000:09:00.0
Jan 22 05:45:31 dt5-i915 kernel: [81541.457152] nvme_pci_free_ctrl@6181: dev<ffff88106019e000>
pdev<ffff880f6e8d6000>
Jan 22 05:45:31 dt5-i915 kernel: [81541.457153] nvme_pci_free_ctrl@6190: dev<ffff88106019e000>
Jan 22 05:45:31 dt5-i915 kernel: [81541.457166] nvme: 0000:09:00.0: code 7
Jan 22 05:45:31 dt5-i915 kernel: [81541.457174] _pci_remove_bus_device@136: 0000:09:00.0:
ffff88107241d800 0000:09
Jan 22 05:45:31 dt5-i915 kernel: [81541.457174] nvme: 0000:09:00.0: code 2
Jan 22 05:45:31 dt5-i915 kernel: [81541.457183] nvme: 0000:09:00.0: code 3
Jan 22 05:45:31 dt5-i915 kernel: [81541.457193] remove PCI device 0000:09:00.0 end
Jan 22 05:45:31 dt5-i915 kernel: [81541.457196] pci: free ffff880f6e8d6000 0000:09:00.0 1bb1:5018
```

Interactive Action Editing

An interactive method for editing actions is built into the system. Editing actions interactively provides built-in help and makes the process easier to visualize.

To illustrate this, we will modify Example 1 above using interactive edits to change the sequence from a "run once and stop" to a "loop" sequence, which then runs until stopped by the **iriser -d <slot> stop** command.

Starting Point

As shown above, we created sequence 0 which runs once and stops. For example:

```
 INFO: Action 0 Sequence 0
Direction: I/O (UC) user writable signal, i/o (LC) signal locked
GPIO|31          24|23          16|15           8|7             0
    |U|U|U|U|S|S|S|U|S|L|R|P|P|M|D|P|T|T|T|T|T|T|T|T|P|P|P|S|3|1|C|P|
    |S|S|S|S|B|B|B|S|T|E|F|L|L|F|U|R|x|x|x|x|x|x|x|x|1|0|W|M|V|2|L|E|
    |R|R|R|R|2|1|0|B|A|D|U|N|A|G|A|S|3|3|2|2|1|1|0|0|C|C|R|R|3|V|K|R|
    |3|2|1|0|G|G|G|D|T|E|E|E|E|E|L|N|P|N|P|N|P|N|P|N|L|L|D|S| | |R|S|
    |T|T|T|T|P|P|P|I|U|D|D|D|D|D|P|T| | | | | | | | |K|K|I|T| | |E|T|
    |S|S|S|S|I|I|I|S|S|S|S|S|S|S|T|L| | | | | | | | |L|L|S|L| | |Q|0|
    |T|T|T|T|O|O|O|L|L|F|F|F|F|F|L| | | | | | | | | | | | | | | |L|L|
    |O|O|O|O|I|I|I|I|I|I|I|I|I|I|O|I|O|O|O|O|O|O|O|O|O|O|I|I|O|O|I|O|Dir:[0x804]0xf002ffcd
Act                                                                  Next|GPIO Value|Time
000 |0|0|0|0|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|0|0|0| 001|0x0f52ff18|10.000000s
001 |0|0|0|0|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|1|0|0| 002|0x0f52ff1c|250.000000ms
002 |0|0|0|0|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|1|0|1| 003|0x0f52ff1d|5.000000s
003 |0|0|0|0|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|1|0|0| 004|0x0f52ff1c|100.000000ns
004 |0|0|0|0|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|1|0|1| 004|0x0f52ff1d|100.000000ns Stop

     Sequencer: Stopped
```

We modify action 4 to sleep for 10 seconds and then loop back to action 0, thus looping indefinitely until stopped.

The interactive session is shown below:

```
iRiser -d 0 edit action 4
Prior action points to me, use GPIO_dir from prior action
Direction: I/O (UC) user writable signal, i/o (LC) signal locked
GPIO|31          24|23          16|15           8|7             0
    |U|U|U|U|S|S|S|U|S|L|R|P|P|M|D|P|T|T|T|T|T|T|T|T|P|P|P|S|3|1|C|P|
    |S|S|S|S|B|B|B|S|T|E|F|L|L|F|U|R|x|x|x|x|x|x|x|x|1|0|W|M|V|2|L|E|
    |R|R|R|R|2|1|0|B|A|D|U|N|A|G|A|S|3|3|2|2|1|1|0|0|C|C|R|R|3|V|K|R|
    |3|2|1|0|G|G|G|D|T|E|E|E|E|E|L|N|P|N|P|N|P|N|P|N|L|L|D|S| | |R|S|
    |T|T|T|T|P|P|P|I|U|D|D|D|D|D|P|T| | | | | | | | |K|K|I|T| | |E|T|
    |S|S|S|S|I|I|I|S|S|S|S|S|S|S|T|L| | | | | | | | |L|L|S|L| | |Q|0|
    |T|T|T|T|O|O|O|L|L|F|F|F|F|F|L| | | | | | | | | | | | | | | |L|L|
    |O|O|O|O|I|I|I|I|I|I|I|I|I|I|O|I|O|O|O|O|O|O|O|O|O|O|I|I|O|O|I|O|Dir:[0x844]0xf002ffcd
Act                                                                  Next|GPIO Value|Time
004 |0|0|0|0|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|1|0|0| 004|0x0f52ff1c|100.000000ns

Enter 32bit hex or bit name/number or h for help. DIRECTION:[0xf002ffcd]:
```

©2024 Copyright, SANBlaze. *All Rights Reserved.*

We do not want to change DIRECTION and therefore we type <cr> to take the default which advances the editor to ask for VALUE.

```
Direction: I/O (UC) user writable signal, i/o (LC) signal locked
GPIO|31            24|23            16|15             8|7              0|
    |U|U|U|U|S|S|S|U|S|L|R|P|P|M|D|P|T|T|T|T|T|T|T|T|P|P|P|S|3|1|C|P|
    |S|S|S|S|B|B|B|S|T|E|F|L|L|F|U|R|x|x|x|x|x|x|x|x|1|0|W|M|V|2|L|E|
    |R|R|R|R|2|1|0|B|A|D|U|N|A|G|A|S|3|3|2|2|1|1|0|0|C|C|R|R|3|V|K|R|
    |3|2|1|0|G|G|G|D|T|E|E|E|E|E|L|N|P|N|P|N|P|N|P|N|L|L|D|S| | |R|S|
    |T|T|T|T|P|P|P|I|U|D|D|D|D|D|P|T| | | | | | | | |K|K|I|T| | |E|T|
    |S|S|S|S|I|I|I|S|S|S|S|S|S|S|T|L| | | | | | | | |L|L|S|L| | |Q|0|
    |T|T|T|T|O|O|O|L|L|F|F|F|F|F|L| | | | | | | | | | | | | | | |L|L|
    |O|O|O|O|I|I|I|I|I|I|I|I|I|I|O|I|O|O|O|O|O|O|O|O|O|O|I|I|O|O|I|O|Dir:[0x844]0xf002ffcd
Act                                                                  Next|GPIO Value|Time
004 |0|0|0|0|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|1|0|0| 000|0x0f52ff1c|10.000000s
Prior action points to me, use GPIO_out from prior action
Direction: I/O (UC) user writable signal, i/o (LC) signal locked
GPIO|31            24|23            16|15             8|7              0|
    |U|U|U|U|S|S|S|U|S|L|R|P|P|M|D|P|T|T|T|T|T|T|T|T|P|P|P|S|3|1|C|P|
    |S|S|S|S|B|B|B|S|T|E|F|L|L|F|U|R|x|x|x|x|x|x|x|x|1|0|W|M|V|2|L|E|
    |R|R|R|R|2|1|0|B|A|D|U|N|A|G|A|S|3|3|2|2|1|1|0|0|C|C|R|R|3|V|K|R|
    |3|2|1|0|G|G|G|D|T|E|E|E|E|E|L|N|P|N|P|N|P|N|P|N|L|L|D|S| | |R|S|
    |T|T|T|T|P|P|P|I|U|D|D|D|D|D|P|T| | | | | | | | |K|K|I|T| | |E|T|
    |S|S|S|S|I|I|I|S|S|S|S|S|S|S|T|L| | | | | | | | |L|L|S|L| | |Q|0|
    |T|T|T|T|O|O|O|L|L|F|F|F|F|F|L| | | | | | | | | | | | | | | |L|L|
    |O|O|O|O|I|I|I|I|I|I|I|I|I|I|O|I|O|O|O|O|O|O|O|O|O|O|I|I|O|O|I|O|Dir:[0x844]0xf002ffcd
Act                                                                  Next|GPIO Value|Time
004 |0|0|0|0|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|1|0|0| 004|0x0f52ff1c|100.000000ns
Enter 32bit hex or bit name/number or h for help. VALUE:[0x0f52ff1c]: perst
```

We want to leave PERST de-asserted, and see that the default for Action 4 is zero (asserted). At this point we have three options in the editor. We can toggle a single bit by name, by bit location, or we can write the entire 32bit output value. We choose toggle by name. The three input options are as follows:

- Change bit value by name - perst
- Change bit value by location - 0
- Change entire GPIO register - 0x0f52ff1d

```
Direction: I/O (UC) user writable signal, i/o (LC) signal locked
GPIO|31            24|23            16|15             8|7              0|
    |U|U|U|U|S|S|S|U|S|L|R|P|P|M|D|P|T|T|T|T|T|T|T|T|P|P|P|S|3|1|C|P|
    |S|S|S|S|B|B|B|S|T|E|F|L|L|F|U|R|x|x|x|x|x|x|x|x|1|0|W|M|V|2|L|E|
    |R|R|R|R|2|1|0|B|A|D|U|N|A|G|A|S|3|3|2|2|1|1|0|0|C|C|R|R|3|V|K|R|
    |3|2|1|0|G|G|G|D|T|E|E|E|E|E|L|N|P|N|P|N|P|N|P|N|L|L|D|S| | |R|S|
    |T|T|T|T|P|P|P|I|U|D|D|D|D|D|P|T| | | | | | | | |K|K|I|T| | |E|T|
    |S|S|S|S|I|I|I|S|S|S|S|S|S|S|T|L| | | | | | | | |L|L|S|L| | |Q|0|
    |T|T|T|T|O|O|O|L|L|F|F|F|F|F|L| | | | | | | | | | | | | | | |L|L|
    |O|O|O|O|I|I|I|I|I|I|I|I|I|I|O|I|O|O|O|O|O|O|O|O|O|O|I|I|O|O|I|O|Dir:[0x844]0xf002ffcd
Act                                                                  Next|GPIO Value|Time
004 |0|0|0|0|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|1|0|1| 000|0x0f52ff1d|10.000000s
Enter 32bit hex or bit name/number or h for help. VALUE:[0x0f52ff1d]:
```

You can toggle as many bits as you want using this method. When satisfied with the VALUE bits, simply accept the current value using <cr>, and the editor will move on to ask for TIME.

```
Direction: I/O (UC) user writable signal, i/o (LC) signal locked
GPIO|31            24|23            16|15             8|7             0|
    |U|U|U|U|S|S|S|U|S|L|R|P|P|M|D|P|T|T|T|T|T|T|T|T|P|P|P|S|3|1|C|P|
    |S|S|S|S|B|B|B|S|T|E|F|L|L|F|U|R|x|x|x|x|x|x|x|x|1|0|W|M|V|2|L|E|
    |R|R|R|R|2|1|0|B|A|D|U|N|A|G|A|S|3|3|2|2|1|1|0|0|C|C|R|R|3|V|K|R|
    |3|2|1|0|G|G|G|D|T|E|E|E|E|E|L|N|P|N|P|N|P|N|P|N|L|L|D|S| | |R|S|
    |T|T|T|T|P|P|P|I|U|D|D|D|D|D|P|T| | | | | | | | |K|K|I|T| | |E|T|
    |S|S|S|S|I|I|I|S|S|S|S|S|S|S|T|L| | | | | | | | |L|L|S|L| | |Q|0|
    |T|T|T|T|O|O|O|L|L|F|F|F|F|F|L| | | | | | | | | | | | | | | |L|L|
    |O|O|O|O|I|I|I|I|I|I|I|I|I|I|O|I|O|O|O|O|O|O|O|O|O|O|I|I|O|O|I|O| Dir:[0x844]0xf002ffcd
Act                                                                  Next|GPIO Value|Time
004 |0|0|0|0|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|1|0|1|  000|0x0f52ff1d|10.000000s
Specify time in s, ms, us, ns (e.g. 25.2ms). Time [10.00s]:
```

We want to leave the device under test (DUT) time to stabilize before staring the loop over, so we enter a reasonable time of 10 seconds. The editor will move to ask for label.

Note: The minimum time between actions, and therefore the minimum time between changing the state of a signal is 80ns. If you enter a value less than 80ns you will be prompted to correct.

Also, the hardware is capable of glitching PCIe lanes at 10nS, the functionality will be enabled in a future SW release.

`Sequence label for Action 4 [Action 4]: Sleep 10s and loop to action 0`

We reject the default label of "Action 4" and provide our own label. The editor moves to ask for "next action".

`Next Action (0=loop, s=stop, w=write, q=quit) [  5]: 0`

The editor provides us with four choices and a default. The default is action + 1, so in our case 5. If you take the default, the editing session will continue and you can define action 5, but we want to loop at this point, so select 0. The "loop" index will be the first index in the current sequence and may not be zero.

Next Action Options

- loop - An action within the sequence will loop back into the sequence and run until manually stopped. The loop default is the first action in the current sequence.
- s - Stop sequence leaving signal states at the currently defined I/O and value.
- w - Write the changes to the given action or actions and exit interactive editor.
- q - Abandon the changes to the give action or actions and quit the interactive editor.

At this point entering "s" will make the current action last and stop the sequencer. Be sure to leave all GPIO bits in the state you wish them to remain once the sequence stops.

©2024 Copyright, SANBlaze. *All Rights Reserved.*

For example, if you leave power (12V) at zero or PERST at 0 your DUT will not be accessible.

```
Direction: I/O (UC) user writable signal, i/o (LC) signal locked
GPIO|31           24|23           16|15            8|7             0|
    |U|U|U|U|S|S|S|U|S|L|R|P|P|M|D|P|T|T|T|T|T|T|T|T|P|P|P|S|3|1|C|P|
    |S|S|S|S|B|B|B|S|T|E|F|L|L|F|U|R|x|x|x|x|x|x|x|x|1|0|W|M|V|2|L|E|
    |R|R|R|R|2|1|0|B|A|D|U|N|A|G|A|S|3|3|2|2|1|1|0|0|C|C|R|R|3|V|K|R|
    |3|2|1|0|G|G|G|D|T|E|E|E|E|E|L|N|P|N|P|N|P|N|P|N|L|L|D|S| | |R|S|
    |T|T|T|T|P|P|P|I|U|D|D|D|D|D|P|T| | | | | | | | |K|K|I|T| | |E|T|
    |S|S|S|S|I|I|I|S|S|S|S|S|S|S|T|L| | | | | | | | |L|L|S|L| | |Q|0|
    |T|T|T|T|O|O|O|L|L|F|F|F|F|F|L| | | | | | | | | | | | | | | |L|L|
    |O|O|O|O|I|I|I|I|I|I|I|I|I|I|O|I|O|O|O|O|O|O|O|O|O|O|I|I|O|O|I|O|Dir:[0x844]0xf002ffcd
Act |                                                               |Next|GPIO Value|Time
004 |0|0|0|0|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|1|0|1| 000|0x0f52ff1d|10.000000s

     Sequencer: Stopped
  INFO: Action edit complete, use iRiser -d 0 start action 4 to start
[root@dt5-i915 iRiser]# iRiser -d 0 show sequence 0

  INFO: Action 0 Sequence 0
Direction: I/O (UC) user writable signal, i/o (LC) signal locked
GPIO|31           24|23           16|15            8|7             0|
    |U|U|U|U|S|S|S|U|S|L|R|P|P|M|D|P|T|T|T|T|T|T|T|T|P|P|P|S|3|1|C|P|
    |S|S|S|S|B|B|B|S|T|E|F|L|L|F|U|R|x|x|x|x|x|x|x|x|1|0|W|M|V|2|L|E|
    |R|R|R|R|2|1|0|B|A|D|U|N|A|G|A|S|3|3|2|2|1|1|0|0|C|C|R|R|3|V|K|R|
    |3|2|1|0|G|G|G|D|T|E|E|E|E|E|L|N|P|N|P|N|P|N|P|N|L|L|D|S| | |R|S|
    |T|T|T|T|P|P|P|I|U|D|D|D|D|D|P|T| | | | | | | | |K|K|I|T| | |E|T|
    |S|S|S|S|I|I|I|S|S|S|S|S|S|S|T|L| | | | | | | | |L|L|S|L| | |Q|0|
    |T|T|T|T|O|O|O|L|L|F|F|F|F|F|L| | | | | | | | | | | | | | | |L|L|
    |O|O|O|O|I|I|I|I|I|I|I|I|I|I|O|I|O|O|O|O|O|O|O|O|O|O|I|I|O|O|I|O|Dir:[0x804]0xf002ffcd
Act |                                                               |Next|GPIO Value|Time
000 |0|0|0|0|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|0|0|0| 001|0x0f52ff18|10.000000s
001 |0|0|0|0|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|1|0|0| 002|0x0f52ff1c|250.000000ms
002 |0|0|0|0|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|1|0|1| 003|0x0f52ff1d|5.000000s
003 |0|0|0|0|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|1|0|0| 004|0x0f52ff1c|100.000000ns
004 |0|0|0|0|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|1|0|1| 000|0x0f52ff1d|10.000000s

     Sequencer: Stopped
```

Now when you start sequence 0 it will continue running and will loop back, turn off the DUT and start again.

```
iriser -d 0 start sequence 0
iriser -d 0 show sequence 0
```

```
  INFO: Action 0 Sequence 0
Direction: I/O (UC) user writable signal, i/o (LC) signal locked
GPIO|31           24|23           16|15            8|7             0|
    |U|U|U|U|S|S|S|U|S|L|R|P|P|M|D|P|T|T|T|T|T|T|T|T|P|P|P|S|3|1|C|P|
    |S|S|S|S|B|B|B|S|T|E|F|L|L|F|U|R|x|x|x|x|x|x|x|x|1|0|W|M|V|2|L|E|
    |R|R|R|R|2|1|0|B|A|D|U|N|A|G|A|S|3|3|2|2|1|1|0|0|C|C|R|R|3|V|K|R|
    |3|2|1|0|G|G|G|D|T|E|E|E|E|E|L|N|P|N|P|N|P|N|P|N|L|L|D|S| | |R|S|
    |T|T|T|T|P|P|P|I|U|D|D|D|D|D|P|T| | | | | | | | |K|K|I|T| | |E|T|
    |S|S|S|S|I|I|I|S|S|S|S|S|S|S|T|L| | | | | | | | |L|L|S|L| | |Q|0|
    |T|T|T|T|O|O|O|L|L|F|F|F|F|F|L| | | | | | | | | | | | | | | |L|L|
    |O|O|O|O|I|I|I|I|I|I|I|I|I|I|O|I|O|O|O|O|O|O|O|O|O|O|I|I|O|O|I|O|Dir:[0x804]0xf002ffcd
Act |                                                               |Next|GPIO Value|Time
000 |0|0|0|0|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|0|0|0| 001|0x0f52ff18|10.000000s
001 |0|0|0|0|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|1|0|0| 002|0x0f52ff1c|250.000000ms
002 |0|0|0|0|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|1|0|1| 003|0x0f52ff1d|5.000000s
003 |0|0|0|0|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|1|0|0| 004|0x0f52ff1c|100.000000ns
004 |0|0|0|0|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|1|0|1| 000|0x0f52ff1d|10.000000s

     Sequencer: Running
iriser -d 0 show sequences
```

Showing Status of Sequences

To show sequences use the command:

iriser -d status

```
INFO: Action 0 Sequence 0

Direction: I/O (UC) user writable signal, i/o (LC) signal locked

GPIO|31           24|23           16|15            8|7             0|
    ├─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┤
    |U|U|U|U|S|S|S|U|S|L|R|P|P|M|D|P|T|T|T|T|T|T|T|T|P|P|P|S|3|1|C|P|
    |S|S|S|S|B|B|B|S|T|E|F|L|L|F|U|R|x|x|x|x|x|x|x|1|0|W|M|V|2|L|E|
    |R|R|R|R|2|1|0|B|A|D|U|N|A|G|A|S|3|3|2|2|1|1|0|0|C|C|R|R|3|V|K|R|
    |3|2|1|0|G|G|G|D|T|E|E|E|E|L|N|P|N|P|N|P|N|P|N|L|L|D|S| | |R|S|
    |T|T|T|T|P|P|P|I|U|D|D|D|D|P|T| | | | | | | |K|K|I|T| | |E|T|
    |S|S|S|S|I|I|I|S|S|S|S|S|S|T|L| | | | | | | |L|L|S|L| | |Q|0|
    |T|T|T|T|O|O|O|L|L|F|F|F|F|L| | | | | | | | | | | | | | |L|L|
    ├─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┤
    |O|O|O|O|I|I|I|I|I|I|I|I|I|O|I|O|O|O|O|O|O|O|O|O|O|I|I|O|O|I|I|Dir:[0x804]0xf002ffcc
Act ├─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┤Next|GPIO Value|Time
000 |0|0|0|0|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|1|0|1| 001|0x0f52ff1d|250.000000ms
001 |0|0|0|1|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|1|0|1| 002|0x1f52ff1d|250.000000ms
002 |0|0|1|0|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|1|0|1| 003|0x2f52ff1d|250.000000ms
003 |0|0|1|1|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|1|0|1| 004|0x3f52ff1d|250.000000ms
004 |0|1|0|0|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|1|0|1| 005|0x4f52ff1d|250.000000ms
005 |0|1|0|1|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|1|0|1| 006|0x5f52ff1d|250.000000ms
006 |0|1|1|0|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|1|0|1| 007|0x6f52ff1d|250.000000ms
007 |0|1|1|1|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|1|0|1| 008|0x7f52ff1d|250.000000ms
008 |1|0|0|0|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|1|0|1| 009|0x8f52ff1d|250.000000ms
009 |1|0|0|1|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|1|0|1| 010|0x9f52ff1d|250.000000ms
010 |1|0|1|0|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|1|0|1| 011|0xaf52ff1d|250.000000ms
011 |1|0|1|1|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|1|0|1| 012|0xbf52ff1d|250.000000ms
012 |1|1|0|0|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|1|0|1| 013|0xcf52ff1d|250.000000ms
013 |1|1|0|1|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|1|0|1| 014|0xdf52ff1d|250.000000ms
014 |1|1|1|0|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|1|0|1| 015|0xef52ff1d|250.000000ms
015 |1|1|1|1|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|1|0|1| 000|0xff52ff1d|250.000000ms
    └─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┘
```

```
INFO: Action 20 Sequence 1

Direction: I/O (UC) user writable signal, i/o (LC) signal locked

GPIO|31           24|23           16|15            8|7             0|
    ├─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┤
    |U|U|U|U|S|S|S|U|S|L|R|P|P|M|D|P|T|T|T|T|T|T|T|P|P|P|S|3|1|C|P|
    |S|S|S|S|B|B|B|S|T|E|F|L|L|F|U|R|x|x|x|x|x|x|x|1|0|W|M|V|2|L|E|
    |R|R|R|2|1|0|B|A|D|U|N|A|G|A|S|3|3|2|2|1|1|0|0|C|C|R|R|3|V|K|R|
    |3|2|1|0|G|G|G|D|T|E|E|E|E|L|N|P|N|P|N|P|N|P|N|L|L|D|S| | |R|S|
    |T|T|T|T|P|P|P|I|U|D|D|D|D|D|P|T| | | | | | | | |K|K|I|T| | |E|T|
    |S|S|S|S|I|I|I|S|S|S|S|S|S|S|T|L| | | | | | | | |L|L|S|L| | |Q|0|
    |T|T|T|T|O|O|O|L|L|F|F|F|F|F|L| | | | | | | | | | | | | | |L|L|
    ├─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┤
    |O|O|O|O|I|I|I|I|I|I|I|I|I|O|I|O|O|O|O|O|O|O|O|O|O|I|I|O|O|I|I|Dir:[0x944]0xf002ffcc
Act ├─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┤Next|GPIO Value|Time
020 |0|0|0|0|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|1|0|1| 021|0x0f52ff1d|125.000000ms
021 |0|0|0|1|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|1|0|1| 022|0x1f52ff1d|125.000000ms
022 |0|0|1|0|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|1|0|1| 023|0x2f52ff1d|125.000000ms
023 |0|0|1|1|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|1|0|1| 024|0x3f52ff1d|125.000000ms
024 |0|1|0|0|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|1|0|1| 025|0x4f52ff1d|125.000000ms
025 |0|1|0|1|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|1|0|1| 026|0x5f52ff1d|125.000000ms
026 |0|1|1|0|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|1|0|1| 027|0x6f52ff1d|125.000000ms
027 |0|1|1|1|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|1|0|1| 028|0x7f52ff1d|125.000000ms
028 |1|0|0|0|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|1|0|1| 029|0x8f52ff1d|125.000000ms
029 |1|0|0|1|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|1|0|1| 030|0x9f52ff1d|125.000000ms
030 |1|0|1|0|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|1|0|1| 031|0xaf52ff1d|125.000000ms
031 |1|0|1|1|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|1|0|1| 032|0xbf52ff1d|125.000000ms
032 |1|1|0|0|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|1|0|1| 033|0xcf52ff1d|125.000000ms
033 |1|1|0|1|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|1|0|1| 034|0xdf52ff1d|125.000000ms
034 |1|1|1|0|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|1|0|1| 035|0xef52ff1d|125.000000ms
035 |1|1|1|1|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|1|0|1| 020|0xff52ff1d|125.000000ms
    └─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┘
```

```
INFO: Action 40 Sequence 2

Direction: I/O (UC) user writable signal, i/o (LC) signal locked

GPIO|31           24|23           16|15            8|7             0|
    ├─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┤
    |U|U|U|U|S|S|S|U|S|L|R|P|P|M|D|P|T|T|T|T|T|T|T|P|P|P|S|3|1|C|P|
    |S|S|S|S|B|B|B|S|T|E|F|L|L|F|U|R|x|x|x|x|x|x|x|1|0|W|M|V|2|L|E|
    |R|R|R|R|2|1|0|B|A|D|U|N|A|G|A|S|3|3|2|2|1|1|0|0|C|C|R|R|3|V|K|R|
    |3|2|1|0|G|G|G|D|T|E|E|E|E|L|N|P|N|P|N|P|N|P|N|L|L|D|S| | |R|S|
    |T|T|T|T|P|P|P|I|U|D|D|D|D|D|P|T| | | | | | | |K|K|I|T| | |E|T|
    |S|S|S|S|I|I|I|S|S|S|S|S|S|S|T|L| | | | | | | |L|L|S|L| | |Q|0|
    |T|T|T|T|O|O|O|L|L|F|F|F|F|L| | | | | | | | | | | | | | |L|L|
    ├─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┤
    |O|O|O|O|I|I|I|I|I|I|I|I|I|O|I|O|O|O|O|O|O|O|O|O|O|I|I|O|O|I|I|Dir:[0xa84]0xf002ffcc
Act ├─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┼─┤Next|GPIO Value|Time
040 |0|0|0|0|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|1|0|1| 041|0x0f52ff1d|1.000000s
041 |0|0|0|1|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|1|0|1| 042|0x1f52ff1d|1.000000s
042 |0|0|1|0|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|1|0|1| 043|0x2f52ff1d|1.000000s
043 |0|0|1|1|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|1|0|1| 044|0x3f52ff1d|1.000000s
044 |0|1|0|0|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|1|0|1| 045|0x4f52ff1d|1.000000s
045 |0|1|0|1|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|1|0|1| 046|0x5f52ff1d|1.000000s
046 |0|1|1|0|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|1|0|1| 047|0x6f52ff1d|1.000000s
047 |0|1|1|1|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|1|0|1| 048|0x7f52ff1d|1.000000s
048 |1|0|0|0|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|1|0|1| 049|0x8f52ff1d|1.000000s
049 |1|0|0|1|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|1|0|1| 050|0x9f52ff1d|1.000000s
050 |1|0|1|0|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|1|0|1| 051|0xaf52ff1d|1.000000s
051 |1|0|1|1|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|1|0|1| 052|0xbf52ff1d|1.000000s
052 |1|1|0|0|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|1|0|1| 053|0xcf52ff1d|1.000000s
053 |1|1|0|1|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|1|0|1| 054|0xdf52ff1d|1.000000s
054 |1|1|1|0|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|1|0|1| 055|0xef52ff1d|1.000000s
055 |1|1|1|1|1|1|1|1|0|1|0|1|0|0|1|0|1|1|1|1|1|1|1|1|0|0|0|1|1|1|0|1| 055|0xff52ff1d|1.000000s
Stop
    └─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┘

    Sequencer is running current action 33 sequence 1
```

Appendix A: Programming the FPGA Code

If your hardware is not up to date or not showing up, here is how to program the FPGA (you need a tip kit).

```
iRiser5_flash_VC8_SPIx4.sh -d 0
Open On-Chip Debugger 0.12.0+dev-01243-g24b656bff (2023-06-09-12:18)
...
Info : sector 16 took 122 ms
...
shutdown
Using config file /etc/iRiser/iRiser5_flash_VC8_SPIx4.cfg
INFO: Wrote image iRiser5_VC8_SPIx4.bin SPI set to x4
```

Appendix B: Programming the EEPROM

The model number, revision, and serial number is stored in an EERPOM on the main I2C bus on the unit.

Programming the eeprom (This will be done in manufacturing, but here is how it is done).

sb_i2c2 -d 0 -Z -D 600952000

```
  INFO: Successfully initialized EEPROM, exiting
```

sb_i2c2 -d 0 -S 952A3100004

```
  INFO: Successfully initialized EEPROM, exiting
```

sb_i2c2 -d 0 -R R01

```
  INFO: Successfully initialized EEPROM, exiting
```

sb_i2c2 -d 0 -e

```
  INFO: System 01[00] SBExpress SN=64fc8ad9 Rev=R01 i2c_main=i2c-0 i2c_mi=i2c-1 SDB=/dev/ttyACM0
VLUN=0
  INFO: Vendor=SANBlaze Technology, Inc.
  INFO: Device=600952000
  INFO: SN=952A3100004
  INFO: Rev=R01
```

The first 5 boards (SNs 001 - 005) should already be up to date, but if you follow the information above you can bring an uninitialized board up to this level.

Appendix C: FAQs

Currently the FAQs are simply a current list of questions the developers think MAY come up but have not technically been frequently asked as the iRiser hardware and software are new.

Question: Can I glitch or shut off PCIe lanes to my device to test its response to losing or noisy PCIe lanes?

Answer: Yes. Use the sequencer as described above to set the state of any of the following signals to zero (clear):

- TX0N
- TX0P
- TX1N
- TX1P
- TX2N
- TX2P
- TX3N
- TX3P

Question: Early documentation for iRiser5 stated it can glitch a PCIe line as quickly as 10nS, but actions take 80nS to load?

Answer: Glitching down to 10nS is supported in the hardware using a special "Glitch" action type which sets the GPIO to the "glitch" state, then back to previous state in any of 10 - 70nS increments. The Glitch functionality will be implemented in a future release of the iRiser program.

Question: How are sequence numbers assigned?

Answer: Sequence numbers start at zero, are contiguous, and are assigned in order to a maximum of 127.

Question: Can I change sequence numbers?

Answer: No, but you can use show sequences to list them, and for any given group of actions (restored from a file for example), they will always be assigned in the same order.

Question: Early documentation mentioned power measurements at 1M/second, where is that functionality?

Answer: Power is currently measured via the **sb_i2c2** program (*sb_i2c2 -d <slot> -m*) or (*sb_i2c2 -d <slot> -m -U for M.2*). The iRiser hardware is capable of sampling 12V power at 1M samples per second and DMA-ing the data directly to host memory. The current hardware is fully capable of this function. The software to implement DMA power sampling will be implemented in a future release.

Question: I'm doing SRIS and SRNS testing on my machine. Will iRiser5 continue to function in a SRIS/SRNS configuration?

Answer: Yes. But certain clock/clock spectrum combinations will cause the iRiser5 to lose the private link from the FPGA to the host and must be avoided. Future SW releases will guard against allowing the user to select combinations that won't work, but currently please contact SANBlaze technical support before changing the SBR image to SRIS or the spread spectrum/common clock and clock source switches.

On an SBExpress- DT5 system, the iRiser5 cannot be configured to perform signal glitching under the hardware configuration Altclock (SW6) = 1, SSC (SW7) = 1. This means that for 2 of the 4 clocking modes – SRNS, and common clocking with no spreading (no SSC) – signal glitching is not supported. The other 2 clocking modes – common clocking with spreading (SSC) and SRIS – are fully supported by the iRiser5. To overcome this limitation, we recommend separating clocking tests from signal glitching tests or using one of our other systems such as the SBEpress-RM5 test system which supports this specific combination of hardware settings.

Question: What is the maximum number of iRiser cards you can support in a DT5 or RM5 system?

Answer: Technically three in a DT5 (one per slot) and 16 in an RM5, but due to the complexity of such a configuration we are currently limiting the number of iRisers in a system to three. Also, some system BIOSes hang when the total number of PCIe devices exceeds a threshold. We are working with BIOS companies on this issue, but it will restrict the number of iRisers in a given system to three for the time being.

Question: Can the FPGA FW code be updated in the field by the user as functionality is added?

Answer: Yes. It is a simple process to update the FPGA code in place in the field. There is no need at this point as there is only one version shipping, but if the need arises firmware will be provided along with an update procedure.