Quarch Technology Ltd

Gen6 AIC PCIe Breaker Modules

Technical Manual

For use with:

QTL3240 – Gen6 PCIe x16-0 Breaker +Triggering Module
QTL3148 – Gen6 PCIe x16-1 Breaker +Triggering Module
QTL3185 – Gen6 PCIe x4-4 Breaker +Triggering Module
QTL3074 – Gen6 PCIe x16-16 Breaker +Triggering Module

Tuesday, 16 September 2025

Change History

1.0	27th August 2025	First Release.
1.1	16th September 2025	Updated manual to include more variations of the x16 AIC Breakers. Updated switch characteristics. Removed note about FW and power injection (outdated comment).

Contents

Introduction

The different variations of the **Torridon Gen6 AIC PCIe Breaker Modules** allow remote switching of PCIe data, power and supporting control pins in a PCIe x4 Gen 6 slot for test automation or fault injection purposes. The main differences between the variations of these modules are the amount of data lanes connected and switchable, more information found here: https://quarch.com/news/which-gen6-breaker-do-i-choose/

The module supports PCIe Gen 2, Gen 3, Gen 4, Gen 5 and Gen 6 devices at data rates up to 64 GT/s.

Each pin is individually switched, allowing complete control over the mating of the card connector. Any individual pin(s) can be disconnected or glitched to simulate failures or configuration changes. If the devices support hot-swap, a sequenced hot-swap event can be run, connecting pins in sequence and simulating pin-bounce if required.

WARNING: Some systems DO NOT support hot swap of a PCIe card while the system is powered up. You should verify that your hardware supports this feature before using it.

The module can disrupt power, data, and control signals down to a minimum glitch length of 50nS.

The module provides a power injection port to optionally power the PCIe add-in card from an external Quarch Programmable Power Module (QTL1999) instead of the PCIe host.

(*The practical minimum time for some pins, such as power pins, may be longer than this)

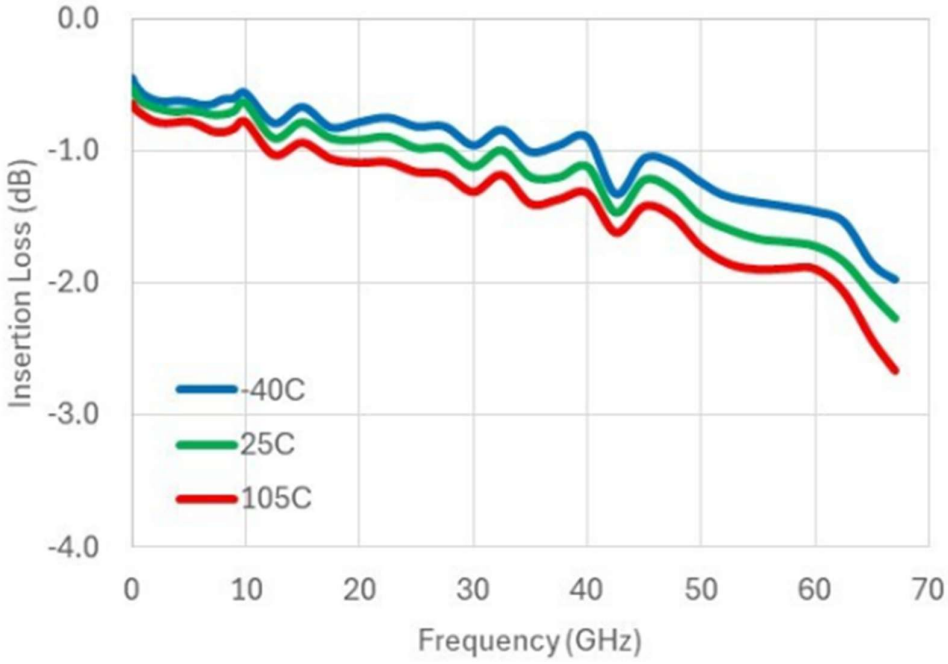
Technical Specifications

Switching Characteristics

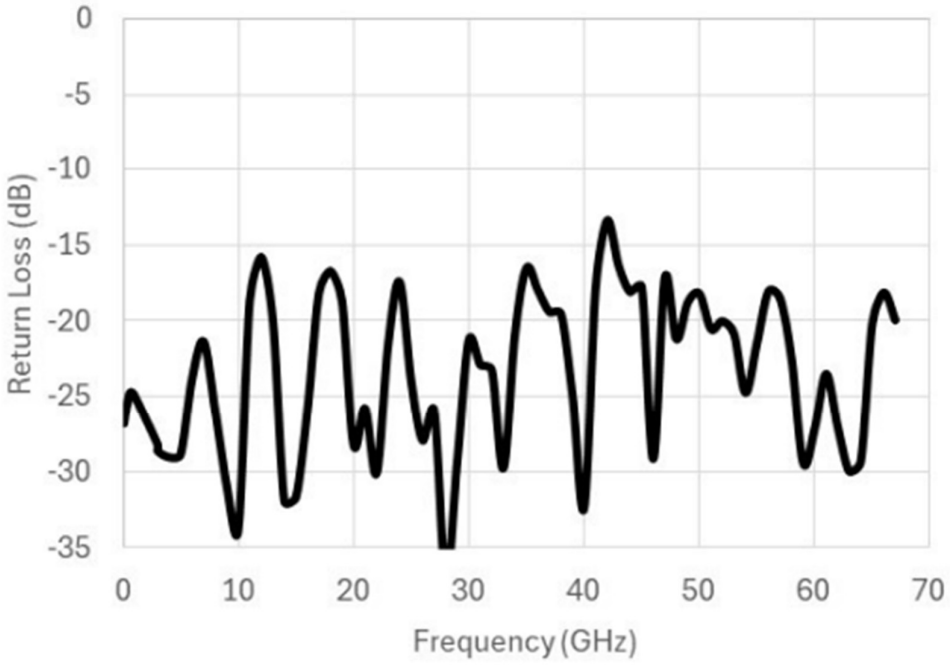
PCIe Connector Pin	Description	Switching Action
A4, A12, A15, A18, A20, A23, A24, A27, A28, A31, A34, A37, A38, A41, A42, A45, A46, A49, A51, A54, A55, A58, A59, A62, A63, A66, A67, A70, A71, A74, A75, A78, A79, A82, B4, B7, B13, B16, B18, B21, B22, B25, B26, B29, B32	PCIe Data and Power Ground Pins	All connected to ground on the Module
A16, A17, A21, A22, A25, A26, A29, A30, A35, A36, A39, A40, A43, A44, A47, A48, A52, A53, A56, A57, A60, A61, A64, A65, A68, A69, A72, A73, A76, A77, A80, A81, B14, B15, B19, B20, B23, B24, B27, B28,	PCIe Data Signal pins	Switched data lanes depend on the variation of the module, data signal which are switched are individually switched by a High-Speed RF Switch. Each signal is AC coupled on the receiver side of the switch with a 1uF capacitor (see signal names for information on what signals are switch for each variation)
A13, A14	PCIe Reference Clock pins	Individually switched by a bilateral analog switch
A2, A3, B1, B2, B3	12V Power Pins	Connected together and switched by 16A power FET
A9, A10, B8	3.3V Power Pins	Connected together and switched by 16A power FET
B10	3.3Vaux Pin	Switched by 16A power FET
A1, B17, B31	Presence Pins	Individually Switched by analog switch
A5, A6, A7, A8, A11, B5, B6, B9, B11, B30	Sideband Signal Pins	Individually Switched by analog switch
A19, A32	Reserved Pins	Each signal is permanently connected from host to device

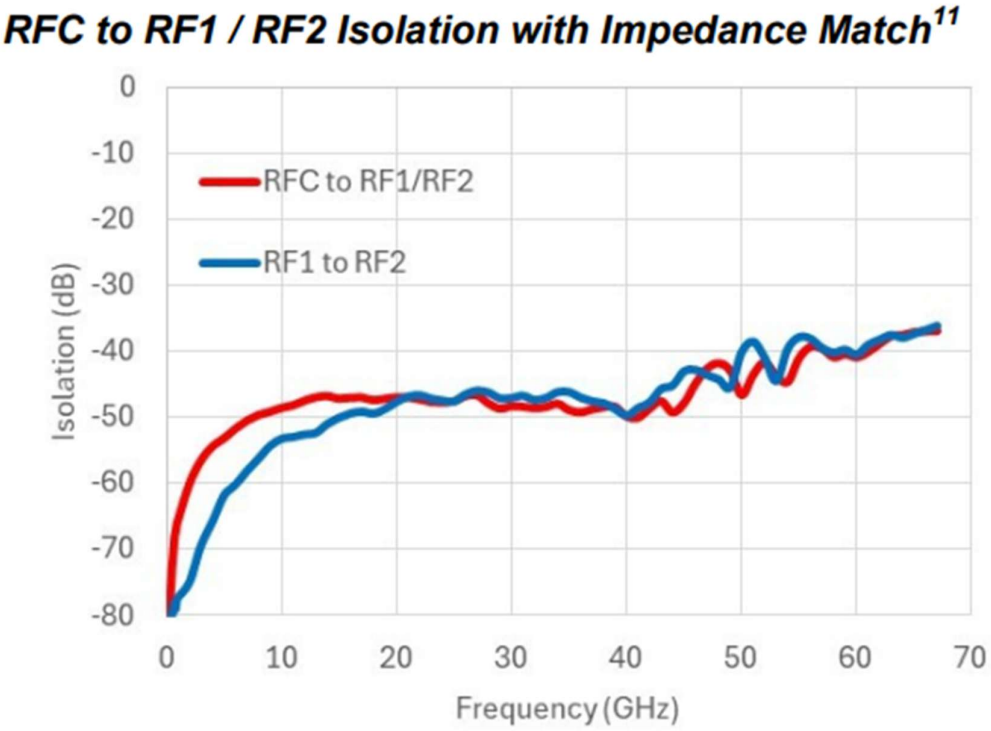
High Speed Switch Characteristics

Insertion Loss with Impedance Match[11]



RFC Return Loss with Impedance Match[12]





Power Injection

The module provides a power injection port to power the PCIe add in card with an external Programmable Power Module (QTL1999). When power injection mode is selected 12V and 3.3V are supplied from the injection port instead of the PCIe host.

To enable power injection:

- The Programmable Power Module must be configured to output 12V and 3.3V and be connected to the PCIe Card module using adaptor QTL2052.
- Jumper J5 on the PCIe Card Module must be fitted to put the PCIe card module into Inject Power mode instead of Host Power Mode.
    - You should only change the jumper when the system is fully powered down

Most PCIe systems expect the card power rails to rise at a specific time. To allow this, the PCIe module support synchronisation of the external power source: By default, when Jumper J5 is fitted, power will NOT be initially supplied to the PCIe card. When the module detects a valid voltage at its input (PC side); it will immediately switch through power to the card. This ensures that the PCIe card powers up as if it were directly attached to the PC.

This function can be turned on/off with the ***CONFig:INJection:SYNC [ON|OFF]*** command. Turning sync mode off is an advanced function and not normally required. If you turn sync mode off then the power module may try to back-power the host if the host power is off.

In sync mode, the PCIe module synchronises 3v3 and 12v rails separately. 3v3 is activated when the host rises above 3v and 12v is activated when the host rises above 11.09v

Power Injection Maximum Continuous Current

Voltage Rail	Voltage Range	Max Current
12V	0-14.4V	4.0A (voltages > 10V)
3.3V	0-4V	4.0A (voltages > 1.5V)

For full specification of Power Margining Module output capabilities see product technical manual; available from https://quarch.com/products/hd-programmable-power-module.

Triggering

Triggering connectors and cables come supplied on the modules. Both trigger connectors are MCX receptacles and are for connection to 3.3V equipment.

Trigger IN

Trigger IN allows you to control the hot-swap state of the module. This can be EDGE triggered (each rising edge swaps the state, causing a power-up or power-down action) or LEVEL triggered (The hot-swap state follows the input signal, every time it changes).

Alternatively, trigger in can be used to control the glitch engine. This can be EDGE triggered (each rising edge starts the currently selected glitch) or LEVEL triggered (each rising edge starts the currently selected glitch and the falling edge will end it, assuming it has not already completed).

Trigger OUT

Trigger OUT allows external equipment to follow the actions applied by the unit. This can be set to follow the hot-swap state, or to follow the glitch state. The trigger out signal will change whenever the action occurs, regardless of the means (trigger in or manual control) used to start the action.

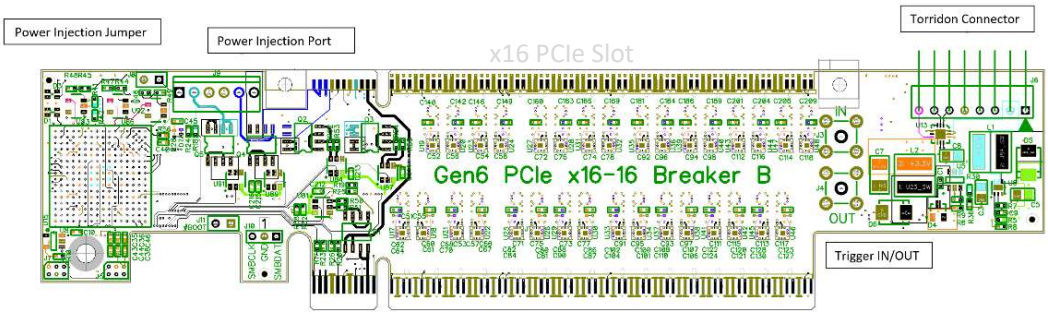
Mechanical Characteristics

Dimensions and Connector Locations

The module is half length and will displace a PCIe add in card upwards by 40.4mm. The horizontal and vertical dimensions of the card are shown below.

The left-hand connector with 'IN' marked below it is 'Trigger IN'.

The right-hand connector with 'OUT' marked below it is 'Trigger OUT'.



LEDs

The module has 4 LEDs:

STATUS Green: This LED is green when the module is able to communicate with the controller.

LINK Green: This LED shows the connection state of the attached PCIe card. This LED has 3 states:

OFF:	ALL PCIe pins are disconnected
GREEN:	ALL PCIe pins are connected
ORANGE:	Some, but not all, PCIe pins are connected

Control Interfaces

All Torridon Control Modules are designed to be used with a Torridon Array Controller (QTL1079) or a single Torridon Interface Card (QTL1144).

The control cable is an ultra-thin Flex cable.

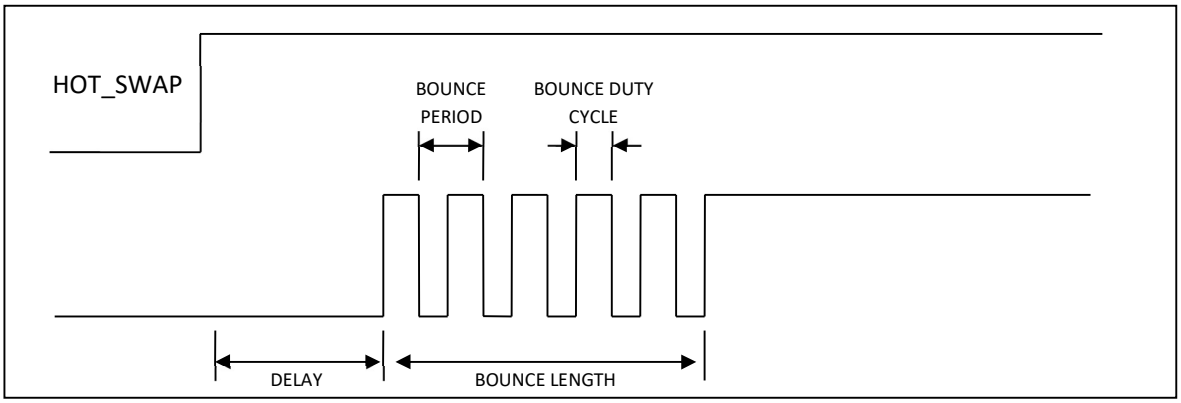
Control Interface	Form Factor	Torridon Module Ports	Control Methods Available	Interfaces
Torridon Array Controller	1U 19” Rack Mounted unit	24 at the front, 4 at the rear	Serial Scripting Script Generation through **TestMonkey GUI**	Serial via DB9 or RJ45 Ethernet
Torridon Interface Card	102mm x 26mm PCB	1 port	Serial Scripting Script Generation through **TestMonkey GUI** Real-time USB Control via **TestMonkey GUI**	Serial via DB9 or RJ45 USB

Basic Concepts

Each switch on the module is called a 'Signal' and can be programmed to follow one of 6 programmable delay and bounce profiles (called 'Sources'). This allows the user to sequence the signal connections in the cable in up to six programmable steps.

Each of the programmable delay and bounce profiles is called a control source, S1 to S6. For each control source the user sets up a delay, and bounce parameters. Three special sources (S0, S7 and S8) are also provided as described in the table below.

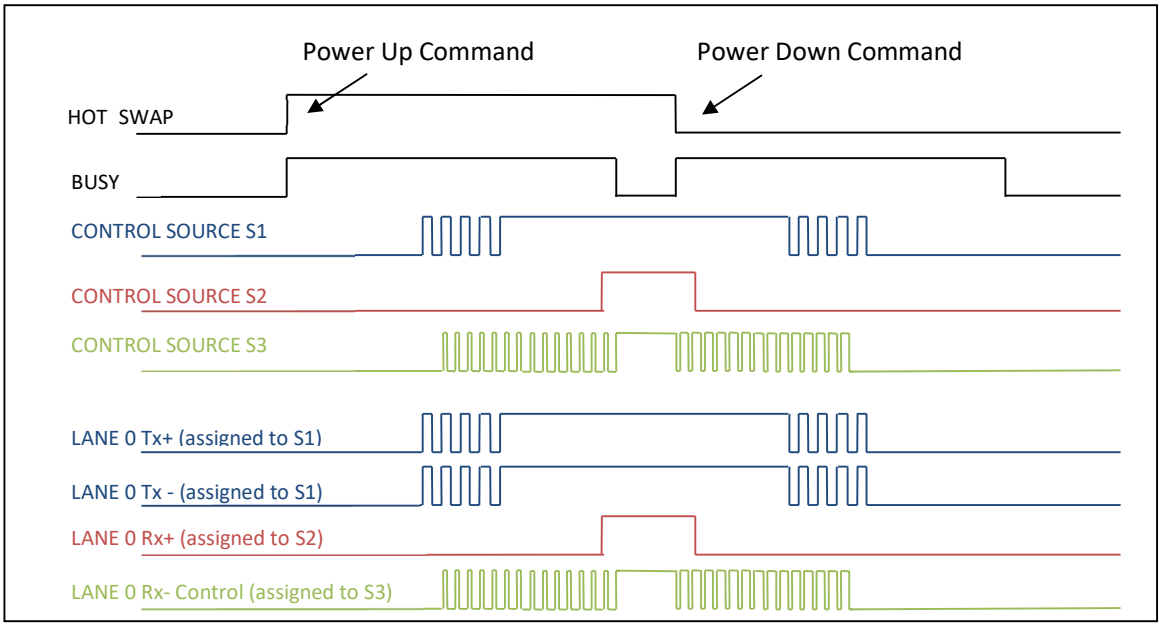
Control Source Parameters for a power up event (Basic Pin Bounce):



Once each delay period is set up, the user assigns each signal to follow the relevant control source, then uses the "run:power up" and "run:power down" commands to initiate the hot-swap.

The BUSY bit 1 in the control register is set during a power up, power down and short operation. This may be used to monitor for the completion of timed events.

Power up and Power down example:

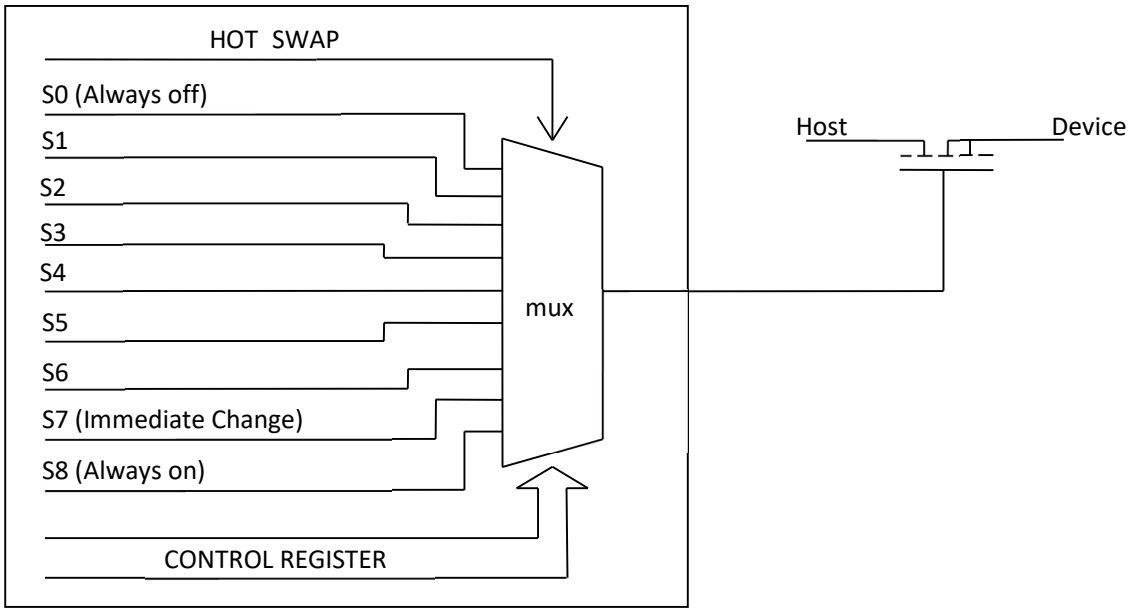


Signal Configuration

Each signal that is switched by the module is usually assigned to one of the 6 timed sources, S1 – S6. Each signal can also be assigned directly to 'always off' (source 0), 'immediate change' (source 7) or 'Always on' (source 8).

To assign a signal to a control source, write to its CONTROL_REGISTER:

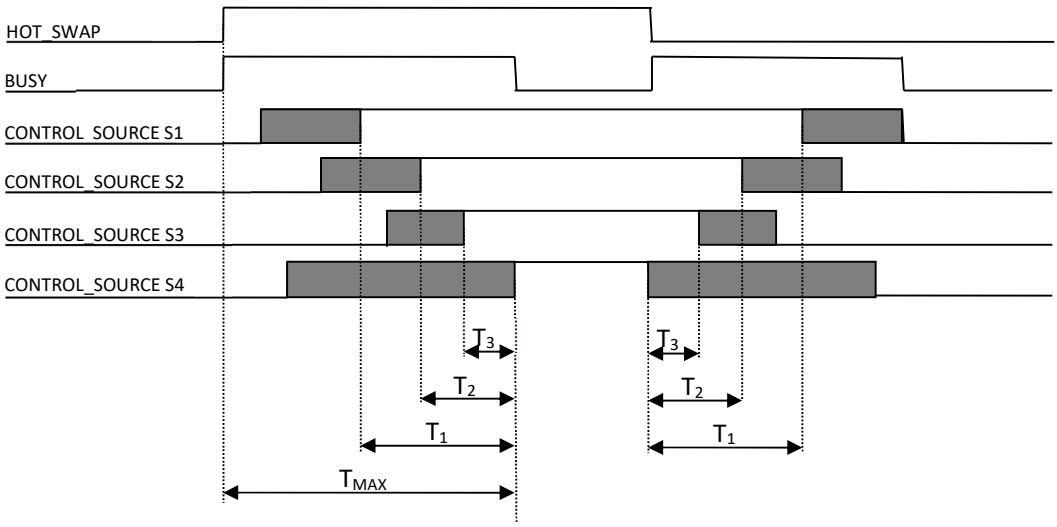
CONTROL_REGISTER Value	Description
0	Signal is always OFF
1	Signal assigned to control source 1
2	Signal assigned to control source 2
3	Signal assigned to control source 3
4	Signal assigned to control source 4
5	Signal assigned to control source 5
6	Signal assigned to control source 6
7	Signal changes with HOT_SWAP
8	Signal is always ON



This diagram shows the 9 possible source settings entering the control MUX for a switched signal. The value of the control register will determine which of the sources are used to control the signal. When enabled, the hot-swap line will cause the MUX to pass the control signal from that source through to the switch.

Power Up vs. Power Down Timing

Each control source is always configured with power-up parameters. The power-down profile is automatically generated by the module, and is the mirror image of the power up:



Power Down Times are automatically generated by the module but can be calculated as follows:

T_{MAX} = Largest Time Period
= the largest value from: (S1_DELAY + S1_BOUNCE_LENGTH),
(S2_DELAY + S2_BOUNCE_LENGTH),

up to

(S6_DELAY + S6_BOUNCE_LENGTH)

Power Down Delay T_X = T_{MAX} – (S_X_DELAY + S_X_BOUNCE_LENGTH)

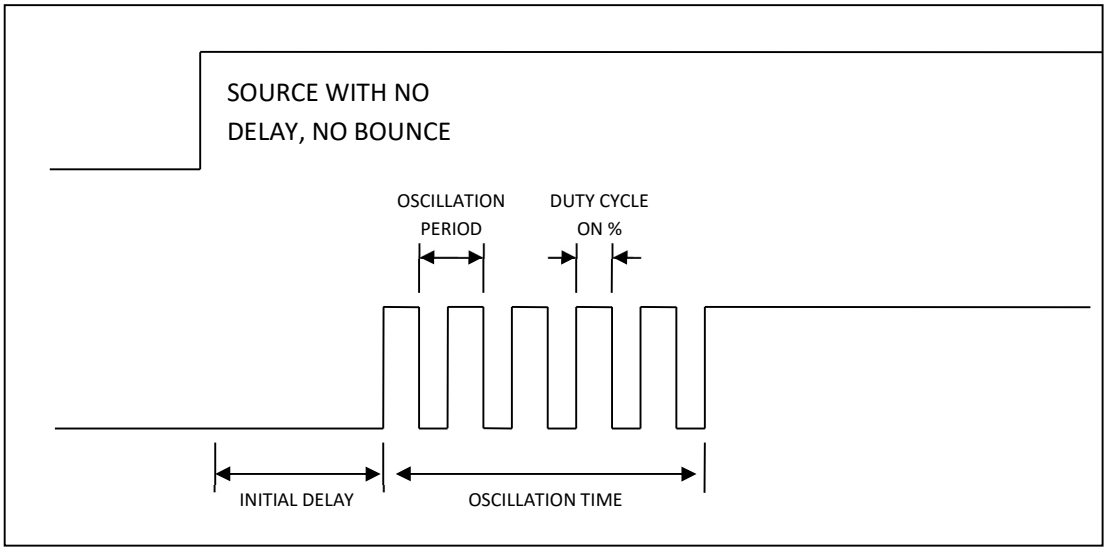
If you require a different power down sequence then you can alter any of the source timing values, pin bounce or signal assignments while the module is in the plugged state. When you initiate the 'pull' action, the new settings will be used.

Pin Bounce Modes

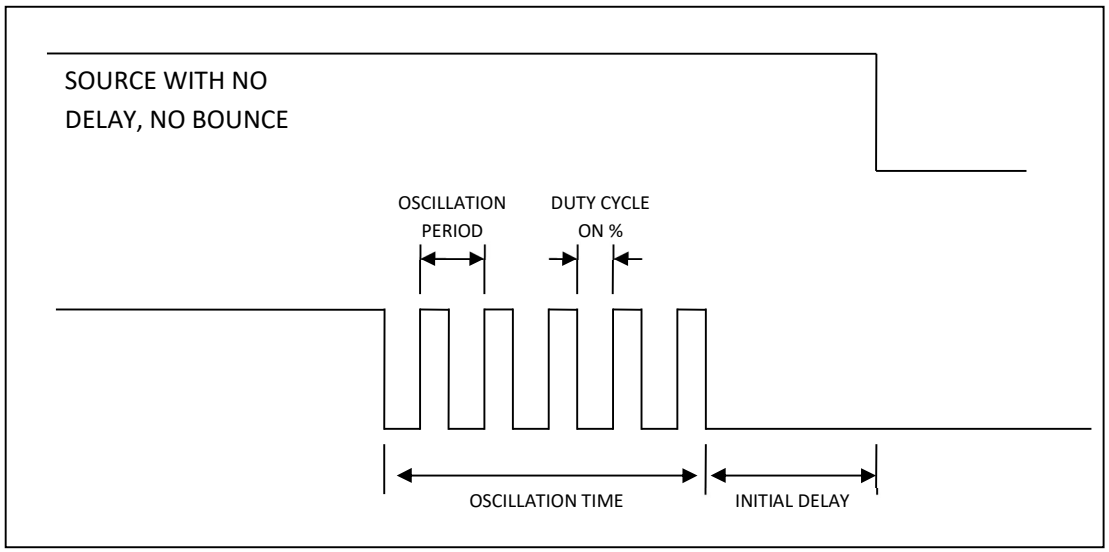
Pin Bounce can be set in two ways:

1. Basic Pin-Bounce (Constant Oscillation Frequency):
   - The Oscillation Time is set in one of two ranges:
     - 0 - 127 milliseconds in steps of 1mS
     - 0 – 1.27 seconds in steps of 10mS
   - The Oscillation Period is for the pattern is set in one of two ranges:
     - 0 - 1.27 milliseconds in steps of 10uS
     - 0 – 127 milliseconds in steps of 1mS
   - The Duty cycle (On %) is set as a percentage value in the range 0-100%
     - A value of 0% would leave the source off for the duration of the Oscillation Time
     - A value of 50% provides a symmetrical square wave as shown below and is the default
     - A value of 100% would turn the signal on for the duration of Oscillation Time

Basic bounce behavior on power up



Basic bounce behavior on power down



2. User Pin-Bounce (Custom Oscillation):

User Pin bounce allows the user to set a custom pattern of up to 112 bits which will be executed by the module instead of standard pin bounce. A custom pattern of alternating 1's and 0's would create a square wave just like the default basic bounce mode.

The executed pattern is determined by a number of factors:

- The Oscillation Time is set in one of two ranges:
  - 0 - 127 milliseconds in steps of 1mS
  - 0 – 1.27 seconds in steps of 10mS
- The Oscillation Period is for the pattern is set on one of the two ranges:
  - 0 - 1.27 milliseconds in steps of 10uS
  - 0 – 127 milliseconds in steps of 1mS
- The Pattern length may be set :
  - 1-112 bits
  - Choose to repeat pattern or sit on last bit until the end of Oscillation Time

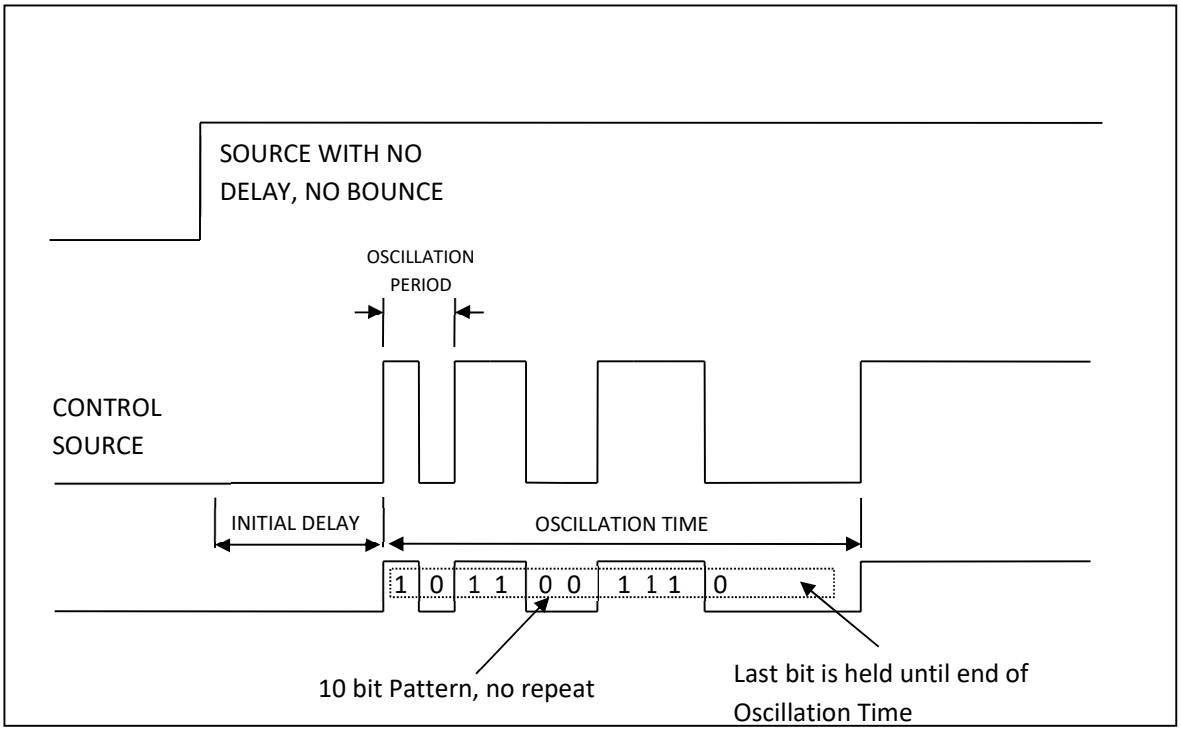
Custom Patterns can get confusing very quickly, the best way to make sure that the power down sequence is the opposite of power up, is to make sure that:

(Bit length * Number of Bits = Oscillation Time)

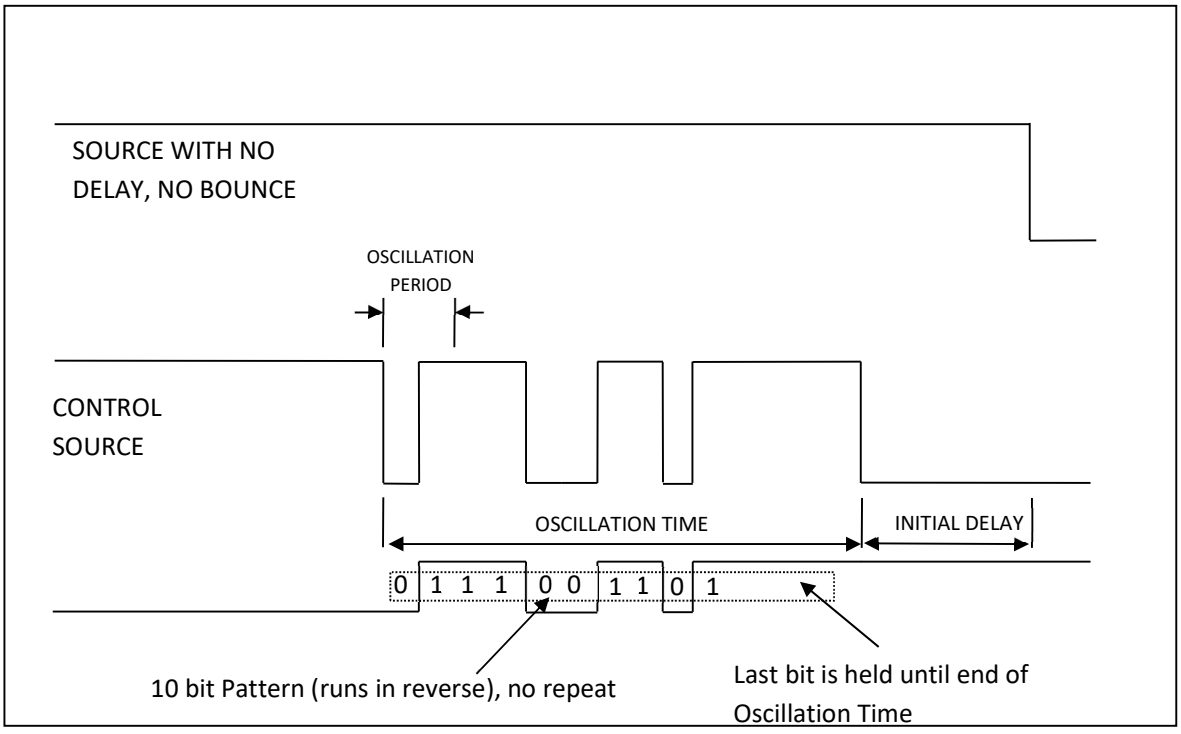
so that the pattern ends exactly at the end of Oscillation time. The SOURce:[x]:BOUNce:PATtern:SETup command does this automatically.

Custom patterns run in reverse on a power down, please see the following diagrams for examples of the same pattern being run on a power up and power down situation.

User bounce behavior on power up



User bounce behavior on power down



Glitch Control

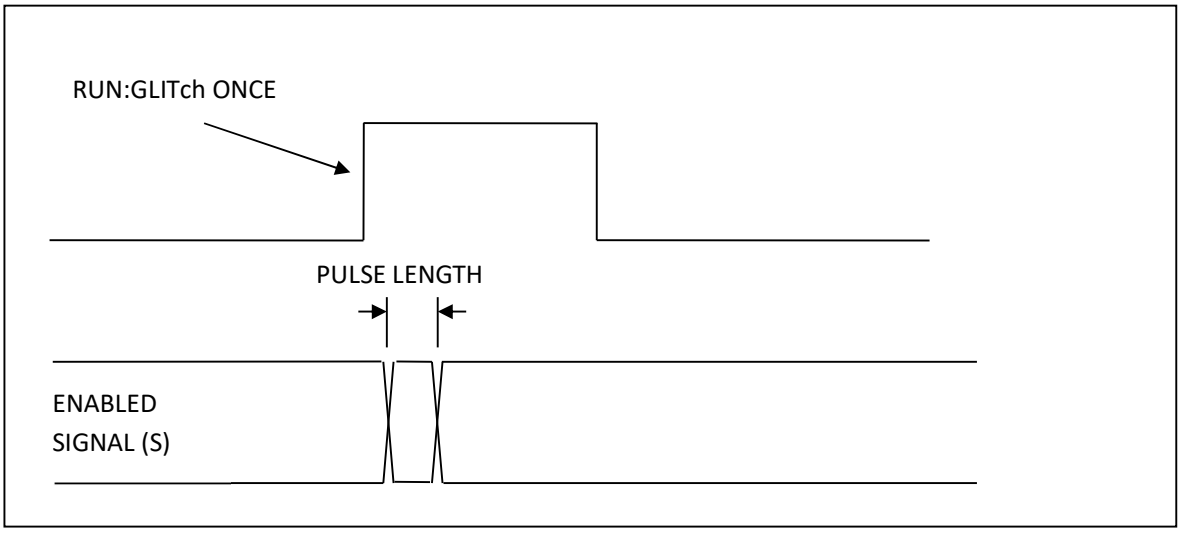
Any control signal may be glitched for a pre-determined length of time using the glitch generator logic.

Each Signal Control register contains a "GLITCH_ENABLE" bit which determines whether the glitch logic will affect that signal. The GLITCH ENABLE bit, defaults to off, so any glitches will have no effect unless explicitly set to do so.

Glitches will invert the current state of the switched signal. Therefore, if a switch is currently OFF, a glitch will turn it ON, and if the switch is ON, it will turn OFF.

Glitches may be applied in 3 modes:

Glitch Once:



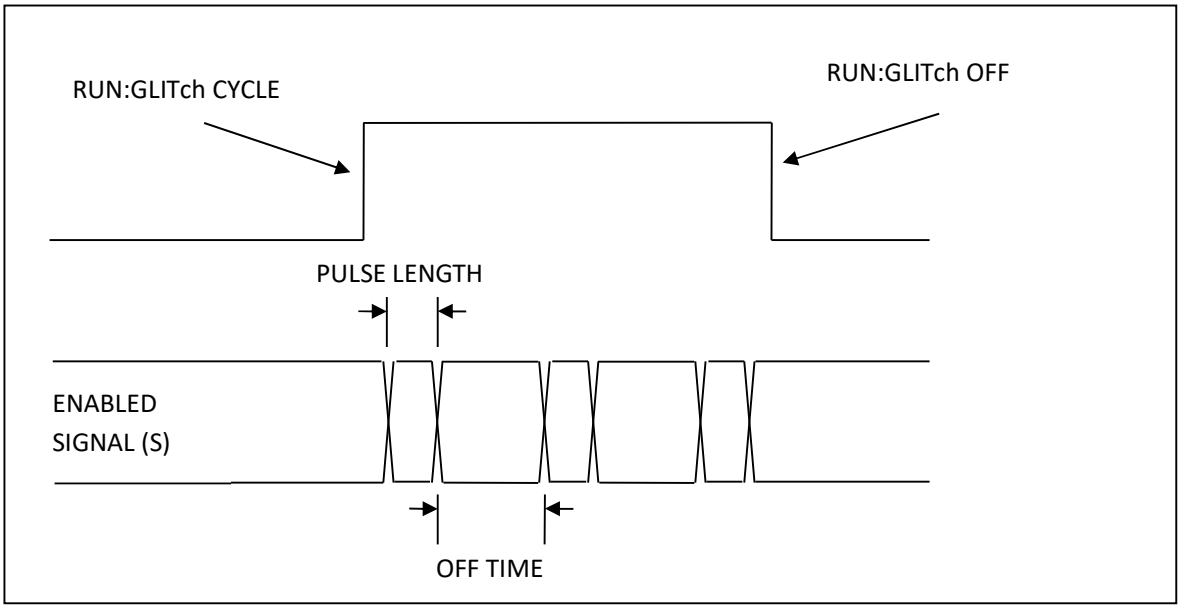
A single glitch is generated when the **RUN**:**GLITch ONCE** command is executed

The length of the glitch is determined by using the **GLITch:SETup** command or the **GLITch:MULTiplier and GLITch:LENgth** commands.

PULSE LENGTH = GLITch:MULTiplier x GLITch:LENgth

Repeated use of the **RUN:GLITch: ONCE** command will generate multiple glitches, it is not necessary to use the **RUN:GLITch OFF** command after a single glitch.

Glitch Cycle:



A sequence of glitches is generated when the **RUN:GLITch CYCLE** command is executed, and continues until **RUN:GLITch OFF** is executed.

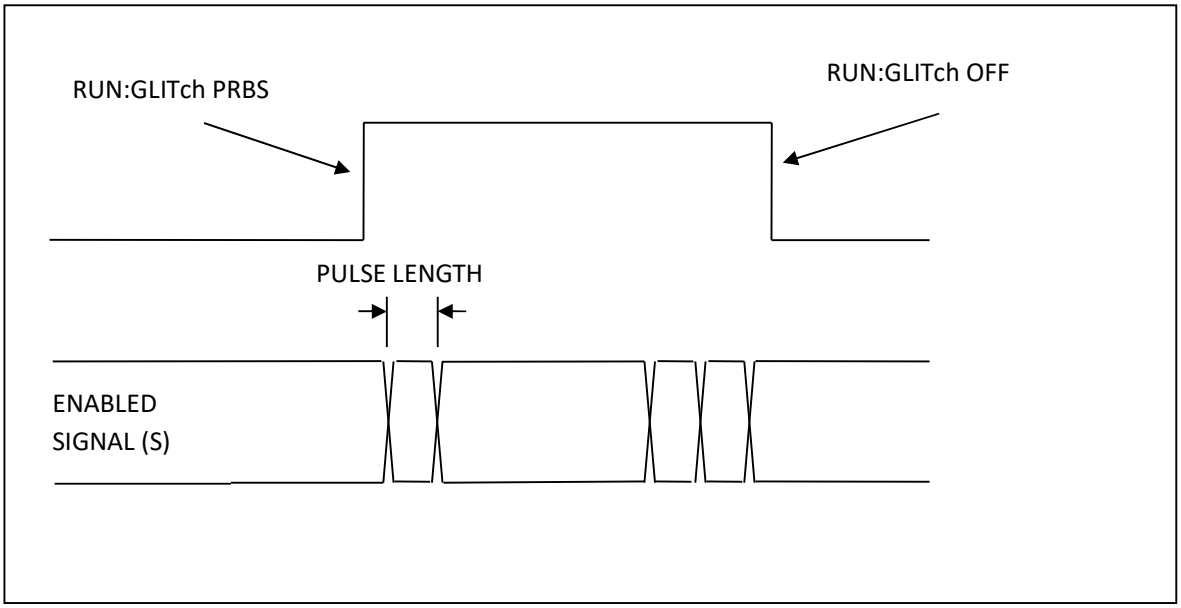
The length of the glitch is determined by using the **GLITch:SETup** command or the **GLITch:MULTiplier** and **GLITch:LENgth** commands:

PULSE LENGTH = GLITch:MULTiplier x GLITch:LENgth

The length of time between each glitch pulse is set in the same way as the glitch length, The length of the gap is determined by using the **GLITch:CYCle:SETup** command or the **GLITch:CYCle:MULTiplier** and **GLITch:CYCle:LENgth** commands:

OFF TIME = GLITch:CYCle:MULTiplier x GLITch:CYCle:LENgth

Glitch PRBS:



A pseudo random sequence of glitches is generated when the **RUN:GLITch PRBS** command is executed, and continues until **RUN:GLITch OFF** is executed.

The length of the glitch is determined by using the **GLITch:SETup** command or the **GLITch:MULTiplier and GLITch:LENGTH** commands:

PULSE LENGTH = GLITch:MULTiplier x GLITch:LENgth

The number of glitches in a set length of time is determined by the **GLITch:PRBS** command. A value of 2 will result in glitches at a ratio of 1:2 (the line will be in a glitched state 50% of the time), whilst a value of 256 will produce glitches in a ratio of 1:256.

Signal Driving

The module has the ability to drive the following sideband signals in certain configurations:

- PERST#
- CLKREQ#
- WAKE#
- PWRBRK#
- SMDAT
- SMCLK

For these signals, the user can specify a behaviour using the **SIGnal:[SIG_NAME]:DRIve:[OPEN|CLOSED] [NONE|HIGH|LOW]** command. The **OPEN** parameter is used to specify the action that the module should take when the switch is open (following a **RUN:POWer DOWN** command or when the signal is assigned to source 0), and the **CLOSED** parameter is used to specify the action to take when the switch should be closed (following a **RUN:POWer UP** command or when the signal is assigned to Source 8). The default behaviour for both **OPEN** and **CLOSED** states is **NONE,** which tells the module not to drive the signal lines at all (just open and close the switch as usual).

The behaviour of the module when signal driving is enabled (set to **HIGH** or **LOW**) is different depending on the signal being driven to avoid hardware conflicts:

Signal Name	Signal Type	Host Side Behaviour		Device Side Behaviour	
		High	Low	High	Low
PERST#	Push/Pull output from Host	Not Driven	Not Driven	Driven High	Driven Low
PWRBRK#	Open Drain output from host	Not Driven	Not Driven	Not Driven	Driven Low
CLKREQ#	Bidirectional Open Drain	Not Driven	Driven Low	Not Driven	Not Driven
WAKE#	Open Drain output from device	Not Driven	Driven Low	Not Driven	Not Driven
SMCLK	Bidirectional Open Drain	Not Driven	Driven Low	Not Driven	Driven Low
SMDAT	Bidirectional Open Drain	Not Driven	Driven Low	Not Driven	Driven Low

Examples:

PERST#

To issue a fundamental reset to the device under test:

PERST# is an active low signal so to assert it we need to drive it low. The module is already powered up so we need to change the **CLOSED** behaviour from **NONE** to **LOW**, and then back again to clear the reset.

>SIGnal:PERST:DRIVE CLOSED LOW

(reset is asserted)

>SIGnal:PERST:DRIVE CLOSED NONE

WAKE#

To wake up the host system:

PERST# is an active low signal so to assert it we need to drive it low. The module is already powered up so we need to change the **CLOSED** behaviour from **NONE** to **LOW**, and then back again to clear the reset.

>SIGnal:WAKE:DRIVE CLOSED LOW

(WAKE is asserted)

>SIGnal:WAKE:DRIVE CLOSED NONE

(WAKE is released)

CLKREQ#

To assert CLKREQ# on both device and host:

CLKREQ# is an active low signal so to assert it we need to drive it low. The module is already powered up so we need to change the **CLOSED** behaviour from **NONE** to **LOW**, and then back again to clear the reset.

>SIGnal:CLKREQ:DRIVE CLOSED LOW

(CLKREQ is asserted)

>SIGnal:CLKREQ:DRIVE CLOSED NONE

(CLKREQ is released)

Advanced Usage:

If we want to assert PERST for a set period of time, we can set the **OPEN** behaviour and use the glitch function on the PERST signal to control the "open" time.

>SIGnal:PERST:GLITch:ENAble ON

> GLITch:SETup 500us 2

>SIGnal:PERST:DRIVE OPEN LOW

>RUN:GLITch ONCE

Signal Monitoring

The 'signal monitoring' feature allows specific signals on a module (normally sideband signals) to be tracked.
The state of a monitored signal can be requested from the module at any time via a command
On 'triggering' modules, the state of a signal can be output in real time to one of the triggering ports. As there are two trigger ports, two signals can be monitored at a time. This is ideal for diverting SM_BUS to an analyser.
For a list of signals on the module that support triggering, see the annex at the end of the manual.

Requesting signal state

To get the state of a monitored signal:

SIGnal:[SIGNAL-NAME]:STATus:[HOST?|DEVice?]

Returns the current state of the monitored signal as **HIGH** or **LOW**. The signal state can (if supported) be monitored independently on both the host and device side of the module.

Live monitoring

This feature is supported on 'Triggering' modules only. Both the trigger IN and OUT ports can be used to monitor a signal.

> WARNING: As the trigger IN port can be ordered to OUTPUT a status, there is a risk of two devices driving against each other and causing damage. Before using the live monitoring feature, you must ensure that you do not have any equipment attached that may try to drive the trigger IN port.

To begin live monitoring, first enable the trigger ports you want to use. This is done via additional options to the existing trigger setup commands:

Trigger OUT port:

```
# Set the trigger mode to sideband monitor
TRIGger:OUT:MODE:SIDEband
```

Trigger IN port (requires double verification)

```
# Set the trigger mode to sideband monitor
TRIGger:IN:MODE:SIDEband
# Also set the trigger IN source to sideband out
TRIGger:IN:SOURCE:SIDEband
```

The commands to control monitoring are:

```
# Select a signal for live monitoring
TRIGger:MONitor[IN|OUT]:[SIGNAL-NAME]:[HOST|DEVice]
```

Sets a trigger port to activate live monitoring for a given signal. The host/device parameter selects the side of the module to monitor on.

```
TRIGger:MONitor[IN?|OUT?]
```

Returns the selection for live monitoring on the given trigger port. The response will be in the form **PERST:HOST** or similar (**SIGNAL_NAME:SIDE**)

Note that the triggering mode must also be set before the live monitoring will start.

Voltage Measurements

The modules are capable of measuring various voltages both for self test and system monitoring. The following measurement points are available:

Measurement Command	Description	Resolution / Accuracy
MEASure:VOLTage:SELF 1v2?	Returns the voltage of the modules internal 1.2v power rail	64mV / 5%
MEASure:VOLTage:SELF 3v3?	Returns the voltage of the modules internal 3.3v power rail – This powers the modules internal circuitry, and the active circuitry on the IN connector	64mV / 5%
MEASure:VOLTage:SELF 12v?	Returns the voltage of the modules internal 12v power rail	64mV/ 5%
MEASure:VOLTage 3v3_host?	Returns the host PC voltage on the 3v3 rail	64mV/ 5%
MEASure:VOLTage 3v3_device?	Returns the voltage being supplied to the device on the 3v3 rail	64mV/ 5%
MEASure:VOLTage 3v3_aux_host?	Returns the host PC voltage on the 3v3 aux rail	64mV/ 5%
MEASure:VOLTage 3v3_aux_device?	Returns the voltage being supplied to the device on the 3v3 aux rail	64mV/ 5%
MEASure:VOLTage 12v_host?	Returns the host PC voltage on the 12v rail	64mV/ 5%
MEASure:VOLTage 12v_device?	Returns the voltage being supplied to the device on the 12v rail	64mV/ 5%

Default Startup State

On power up or reset, the control modules enter a default state. On this module, all signals are connected at startup. The “run:power down” command will immediately disconnect the card without needing any initial setup.

The default hot-swap scenario will disconnect all pins based on the pin length, without any pin-bounce. All data and control pins are assigned to source 1 and will change immediately. Presence pins are assigned to source 2 and will change 25ms after the other pins.

Source Number	Initial Delay	Pin Bounce Mode	Bounce Length	Bounce Period	Bounce Duty Cycle
1	0mS	Standard	0mS	0uS	50%
2	25mS	Standard	0mS	0uS	50%
3	0mS	Standard	0mS	0uS	50%
4	0mS	Standard	0mS	0uS	50%
5	0mS	Standard	0mS	0uS	50%
6	0mS	Standard	0mS	0uS	50%

Signal	Assigned Source
TX0_PL	Source 1
TX0_MN	Source 1
RX0_PL	Source 1
RX0_MN	Source 1
…	Source 1
TX15_PL	Source 1
TX15_MN	Source 1
RX15_PL	Source 1
RX15_MN	Source 1
REFCLK_PL	Source 1
REFCLK_MN	Source 1
12V_POWER	Source 1
3V3_POWER	Source 1
3V3_AUX	Source 1
PERST	Source 1
WAKE	Source 1
SMCLK	Source 1

SMDAT	Source 1
PWRBRK	Source 1
PRESENT1	Source 2
PRESENT2_B17	Source 2
PRESENT2_B31	Source 2
TRST	Source 1
TCK	Source 1
TDO	Source 1
TDI	Source 1
TMS	Source 1

** Data lane names depend on variations*
x16-0 has no PCIe data lane switching.
x16-1 only switches lane zero, all other lanes always connected.
x4-4 only connects and switches first 4 lanes (Lane 5 upwards not connected).
x16-16 has switches on all data lanes.

Hot-Swap State:
The module is in the 'plugged' state, waiting for a **RUN:POWer DOWN** command to disconnect it.

Controlling the Module

The module can be controlled either by:

- Serial ASCII terminal (such as HyperTerminal)
  This is normally used with scripted commands to automate a series of tests. The commands are normally generated by a script or user code (PERL, TCL, C, C# or similar).
- Telnet Terminal (Only when connected to an Array Controller or on a module with a PoE/LAN option). This mode uses exactly the same commands as the serial ASCII terminal
- USB
  Quarch's TestMonkey application can control a single module via USB, this allows simple graphical control of the module.

Serial Command Set

When connected via a serial terminal, the module has a simple command line interface

SCPI Style Commands

These commands are based on the SCPI style control system that is used by many manufacturers of test instruments. The entire SCPI specification has NOT been implemented but the command structure will be very familiar to anyone who has used it before.

- SCPI commands are NOT case sensitive
- SCPI commands are in a hierarchy separated by ':' (LEVel1:LEVel2:LEVel3)
- Most words have a short form (e.g. 'register' shortens to 'reg'). This will be documented as REGister, where the short form is shown in capitals.
- Some commands take parameters. These are separated by spaces after the main part of the command (e.g. "meas:volt:self 3v3?" Obtains the 3v3 self test measurement)
- Query commands that return a value all have a '?' on the end
- Commands with a preceding '*' are basic control commands, found on all devices
- Commands that do not return a particular value will return "OK" or "FAIL". Unless disabled, the fail response will also append a text description for the failure if it can be determined.

 # [comments]

Any line beginning with a # character is ignored as a comment. This allows commenting of scripts for use with the module.

***RST**

Triggers a reset, the module will behave as if it had just been powered on

***CLR**

Clear the terminal window and displays the normal start screen. Also runs the internal self test. The same action can be performed by pressing return on a blank line.

***IDN?**

Displays a standard set of information, identifying the device. An example return is shown below

Family: Torridon System		[The parent family of the device]
Name: Ethernet Cable Pull Module		[The name of the device]
Part#: QTL1271-01		[The part number of the hardware]
Processor:	QTL1159-01,3.50	[Part# and version of firmware]
Bootloader:	QTL1170-01,1.00	[Part# and version of bootloader]
FPGA 1:1.0		[Version of FPGA core]

***TST?**

Runs a set of standard tests to confirm the device is operating correctly, these tests are also performed at start up. Returns 'OK' or 'FAIL' followed by a list of errors that occurred, each on a new line.

CONFig:MODE BOOT

Configures the card for boot loader mode (to update the firmware), requires an update utility on the PC.

CONFig:MESSages [SHORt|USER]
CONFig:MESSages?

Gets or sets the mode for messages that are returned to the user's terminal

Short: Only a "FAIL" or "OK" will be returned
User: Full error messages are returned to the user on failure

CONFig:TERMinal USER

Sets the terminal response mode to the default 'User' setting. This is intended for use with HyperTerminal or similar and manually typed commands

CONFig:TERMinal SCRIPT

Sets the terminal response mode for easier parsing. Especially useful from a UNIX/LINUX based system. Characters sent from the PC are not echoed by the device and a <CR><LF> is sent after the cursor to force a flush of the USART buffer.

CONFig:TERMinal?

Returns the current terminal mode

CONFig:WIDth [1|2|4|8|16]

Sets the lane width of the module. This deactivates the unused lanes and prevents them from being connected, overriding the normal signal settings.

CONFig:WIDth?

Returns the current lane width setting

DEPRECATED COMMANDS – Provided for backwards compatibility, we strongly suggest you use the 'Signal' and 'Source' commands instead.

REGister:READ [0xAA]

Returns the value of the register with address [0xAA]. [0xAA] should be in hex format and preceded by the suffix "0x". e.g. "0x6D". The value is returned in the same form as the address.

REGister:DUMP [0xA1] [0xA2]

Returns the value of each register in a range, starting at the first register address, up to the second. [0xA1] and [0xA2] should be in hex format and preceded by the suffix "0x". Each data value will be returned on a new line.

REGister:WRITe [0xAA] [0xDD]

Writes the byte [0xDD] to register [0xAA], both [0xDD] and [0xAA] should be in hex format and preceded by the suffix "0x". The command returns "OK" or "FAIL".

CONFig:INJection:SYNC [ON|OFF]
CONFig:INJection:SYNC?

Sets or returns the power injection sync state. When on AND Jumper J5 is connected, the PCIe card will not have power supplied unless the host power is detected

CONFig:INJection:MODE?

Returns the state of the J5 jumper. When ON, the power injection jumper (J5) is connected.

SOURce:[1-6|ALL]:SETup [#1] [#2] [#3] [#4]

Sets up the source in a single command. All parameters are positive integer numbers:

#1 = Initial delay (mS)
[Limits: 0 to 127ms in steps of 1ms, 0 to 1270ms in steps of 10ms]
#2 = Bounce length (mS)
[Limits: 0 to 127ms in steps of 1ms, 0 to 1270ms in steps of 10ms]
#3 = Bounce Period (uS)
[Limits: 10 to 1270us in steps of 10us, 1000 to 127000us in steps of 1000us]
#4 = Duty Cycle (%)
[Limits: 0 to 100% in steps of 1%]

SOURce:[1-6|ALL]:DELAY [#ms]
SOURce:[1-6|ALL]:DELAY?

Sets the initial delay of a source in mS. The delay is entered as a integer number with no units. E.g. “Source:1:delay 300”.

#1 = Initial delay (mS)
[Limits: 0 to 127ms in steps of 1ms, 0 to 1270ms in steps of 10ms]

SOURce:[1-6|ALL]:BOUNce:SETup [#1] [#2] [#3]

Sets up the bounce parameters in a single command. All parameters are positive integer numbers:

#1 = Bounce length (mS)
[Limits: 0 to 127ms in steps of 1ms, 0 to 1270ms in steps of 10ms]
#2 = Bounce Period (uS)
[Limits: 10 to 1270us in steps of 10us, 1000 to 127000us in steps of 1000us]
#3 = Duty Cycle (%)
[Limits: 0 to 100% in steps of 1%]

SOURce:[1-6|ALL]:BOUNce:LENgth [#ms]
SOURce:[1-6|ALL]:BOUNce:LENgth?

Sets the length of the pin bounce in mS. The delay is entered as a decimal number with no units. E.g. “Sour:2:boun:len 50”.

#1 = Bounce length (mS)
[Limits: 0 to 127ms in steps of 1ms, 0 to 1270ms in steps of 10ms]

SOURce:[1-6|ALL]:BOUNce:PERiod [#us]
SOURce:[1-6|ALL]:BOUNce:PERiod?

Sets the bounce period of the pin bounce in uS. The value is entered as a decimal number with no units. E.g. “Sour:6:boun:period 300”.

#1 = Bounce Period (uS)
[Limits: 10 to 1270us in steps of 10us, 1000 to 127000us in steps of 1000us]

SOURce:[1-6|ALL]:BOUNce:DUTY [#%]
SOURce:[1-6|ALL]:BOUNce:DUTY?

Sets the duty cycle of the pin bounce as a %. The value is entered as a decimal number with no units. E.g. “source:3:bounce:duty 50”.

#1 = Duty Cycle (%)
[Limits: 0 to 100% in steps of 1%]

SOURce:[1-6|ALL]:BOUNce:MODE [SIMPLE|USER]
SOURce:[1-6|ALL]:BOUNce:MODE?

Sets the bounce pattern to SIMPLE (Duty cycle driven oscillation) or USER (User defined custom pattern).

SOURce:[1-6|ALL]:BOUNce:PATtern:WRITe [0xAAAA] [0xDDDD]

Writes a word of the custom bounce pattern to the give address within the pattern
0xAAAA is the address (for example 0x0002)
0xDDDD is the pattern data (for example 0x13F2)

SOURce:[1-6|ALL]:BOUNce:PATtern:READ [0xAAAA]

Reads a word of the custom bounce pattern
0xAAAA is the address (for example 0x0002)

SOURce:[1-6|ALL]:BOUNce:PATtern:DUMP [0xAAAA] [0xAAAA]

Reads a range of words from the custom bounce pattern
0xAAAA is the start and end address range (for example 0x0002)

SOURce:[1-6|ALL]:BOUNce:CLEAR

Removes any pin bounce from the source and sets all bounce settings to default values. See “Default Startup State” for details for the default settings.

SOURce:[1-6|ALL]:STATE [ON|OFF]
SOURce:[1-6|ALL]:STATE?

Sets or returns the enable state of the source. Any signals assigned to a disabled (off) source will immediately be disconnected and vice versa. If a source state is changed, all signals assigned to it will change at exactly the same time (if a change is required).

SOURce:[1-6]:BOUNce:PATtern:LENgth [#bits]
SOURce:[1-6]:BOUNce:PATtern:LENgth ?

Sets or returns the number of bits of the custom bounce pattern that are to be used. This defaults to the maximum (112) and can be reduced to create more accurate patterns.

SOURce:[1-6]:BOUNce:PATtern:REPeat [ON|OFF]
SOURce:[1-6]:BOUNce:PATtern:REPeat ?

Sets the custom pattern repeat flag. This is used when the current custom bounce pattern is shorter that the specified bounce length. When the flag is set (default) the pattern will wrap. When this flag is off, the last bit of the pattern will be repeated.

SOURce:[1-6]:BOUNce:PATtern:SETup [#us] [#binarypattern]

Sets a basic custom pattern from a single command. This command will alter the bounce period, bounce length, pattern length and the custom pattern.

[#uS] – Integer value of uS to specify the period. The length of each bit in the pattern will be half of this value. 20uS is the minimum value (10uS per bit)

[#binarypattern] – String parameter containing 1s and 0s, for example “001” is a 2 bit pattern that is low for 2 bits then high for 1. The given pattern will always be padded up to the nearest millisecond. This is because the total glitch length has a 1mS resolution.

SIGnal:[SIG_NAME|ALL]:SETup [#num]
SIGnal:[SIG_NAME|ALL]:SOURce [#num]

Assigns a given signal to a numbered timing source (0-8). SIGNAL_NAME is one of the signals/groups as found in the 'Signal Names' appendix at the end of this manual

SIGnal:[SIG_NAME|ALL]:GLITch:ENAble [ON|OFF]
SIGnal:[SIG_NAME|ALL]:GLITch:ENAble?

Enables a signal for glitching. If this in on, the signal will be glitched whenever the glitch logic is in use. Multiple signals may be set to glitch at the same time.

SIGnal:[SIG_NAME]:DRIve:[OPEn|CLOsed] [NONE|HIGH|LOW]
SIGnal:[SIG_NAME]:GLITch:[OPEn|CLOsed]?

Sets the drive state for signals that can be driven (forced to a high or low state). See the section on signal driving for full details

GLITch:SETup [MULTIPLIER_STEP] [#count]

Sets up the length of the glitch in a single command.

#1 = Multiplier factor for glitch length (mS)
[50ns|500sn|5us|50us|500us|5ms|50ms|500ms]
#2 = Length of the glitch (number of times the multiplication factor will be run)
[Limits: 0 to 255 in steps of 1]

This gives a maximum glitch of 127.5 Seconds.

GLITch:MULTiplier [MULTIPILER_STEP]
GLITch:MULTiplier?

Sets the multiplier value for the glitch time to one of the specified durations.
This factor is multiplied with the **GLITch:LENgth** value to give the actual glitch time.

#1 = Multiplier factor for glitch length (mS)
[50ns|500sn|5us|50us|500us|5ms|50ms|500ms]

GLITch:LENgth [#count]
GLITch:LENgth?

This value is multiplied by **GLITch:MULTiplier** to give the glitch duration.

#1 = Length of the glitch (number of times the multiplication factor will be run)
[Limits: 0 to 255 in steps of 1]

GLITch:CYCle:SETup [MULTIPLIER_STEP] [#count]

Sets up the length of the glitch cycle in a single command.

#1 = Multiplier factor for glitch cycle length (mS)
[50ns|500sn|5us|50us|500us|5ms|50ms|500ms]
#2 = Length of the glitch cycle (number of times the multiplication factor will be run)
[Limits: 0 to 255 in steps of 1]

This gives a maximum glitch cycle time of 127.5 Seconds.

GLITch:CYCle:MULTiplier [MULTIPILER_STEP]
GLITch:CYCle:MULTiplier?

Sets the multiplier value for the glitch cycle time to one of the specified durations.
This factor is multiplied with the **GLITch:CYCle:LENgth** value to give the actual time between cycled glitches.

#1 = Multiplier factor for glitch length (mS)
[50ns|500sn|5us|50us|500us|5ms|50ms|500ms]

GLITch:CYCle:LENgth [#count]
GLITch:CYCle:LENgth?

This value is multiplied by **GLITch:CYCle:MULTiplier** to give the actual time between cycled glitches.

#1 = Length of the glitch (number of times the multiplication factor will be run)
[Limits: 0 to 255 in steps of 1]

GLITch:PRBS [2|4|8|16|32|64|128|256|512|1024|2048|4096|8192|16384|32768|65536]

Sets the PRBS rate for Pseudo Random repeat glitching, this is a ratio, 2 means 1:2 (approximately 50% of the time the signal will be glitched), 256 means 1:256.

#1 = PRBS Ratio
[2|4|8|16|32|64|128|256|512|1024|2048|4096|8192|16384|32768|65536]

RUN:POWer [UP|DOWN]

Initiates a plug or pull operation (legacy name used to preserve compatibility between Torridon modules). This is done by changing the HOT_SWAP bit, register 0x00 bit 0. This is the master control for all switches on the card. The same action can be performed by writing this bit directly.

The command will fail if you order a power up when the module is already in the connected state and vice-versa as the action cannot be performed.

The "OK" response will be returned as soon as the hot-swap event has begun. If your timing sequence is very long you may have to poll the BUSY bit in register 0 to check when it has completed.

RUN:POWer?

Returns the current plugged/pulled state of the module.

RUN:GLITch ONCE

Triggers a single glitch with length **GLITch:MULTiplier** x **GLITch:LENgth.**

RUN:GLITch CYCLE

Triggers a sequence of repeated glitches that run until the **RUN:GLITch STOP** command is executed. All signals with **GLITch:ENAble** set to ON are glitched for **GLITch:MULTiplier** x **GLITch:LENgth** and then released for a duration of **GLITch:MULTiplier** x **GLITch:LENgth** x **GLITch:CYCLE**. This is repeated until the **RUN:GLITch STOP** command is run.

RUN:GLITch PRBS

Triggers a PRBS glitch sequence which runs until the **RUN:GLITch STOP** command.

RUN:GLITch STOP

Stops any running glitch sequence.

RUN:GLITch?

Returns the state of the current glitch sequence running on the module

Triggering Commands

These commands are found only on the triggering version of the module

TRIGger:IN:MODE [OFF|POWER|GLITCH]
TRIGger:IN:MODE?

Sets/Return the trigger in mode. This choses the action to be performed when a trigger in is received.

POWER : Trigger in will alter the hot-plug state
GLITCH : Trigger in will start a glitch

TRIGger:IN:TYPE [EDGE|LEVEL]
TRIGger:IN:TYPE?

Sets/Returns the trigger in signal type. This describes how the trigger in signal should be interpreted. See the triggering details for information on how each trigger mode is affected

TRIGger:OUT:MODE [OFF|POWER|GLITCH]
TRIGger:OUT:MODE?

Sets/Returns the trigger out mode. This choses the action that will cause a trigger out to occur.

POWER : Trigger out will occur on hot-swap
GLITCH : Trigger out will occur on glitch

RUN:INTerrupt?

Returns a list of active interrupt flags and also clears those flags. Flags are:

COMPLETE : An action (such as hot-swap or glitch completed fully)
TRIGGERED : An external trigger occurred

GLITch:MODE [ONCE|CYCLE|PRBS]
GLITch:MODE?

Sets/Returns the current glitch mode. This allows you to choose the glitch action that will be performed if the trigger:in:mode is set to 'GLITCH'.

Control Register Map

Access to the FPGA registers should not be required for the majority of operations and customers are strongly encouraged to use the high level commands in order to maintain compatibility with future firmware versions. If you require details of the register map, please contact:

support@quarch.com

Appendix 1 - Signal Names

The following signal names are used to specify a single signal or a group of signals. These may be used in commands that take a parameter “SIGNAL_NAME”. Note that some commands, such as those returning a value, only accept a parameter that resolves to a single signal. In this case you cannot use the group names

Signals

TX0_PL (Data transmitted from the ‘input’ port on Lane 0 (+ve side of differential pair)
TX0_MN
RX0_PL (Data received at the ‘input’ port on Lane 0 (+ve side of differential pair)
RX0_MN
…
TX15_PL (Data lanes names only appear on modules if it’s a switchable signal see *)
TX15_MN
RX15_PL
RX15_MN

REFCLK_PL
REFCLK_MN

12V_POWER
3V3_POWER
3V3_AUX

PERST
WAKE

SMCLK
SMDAT

PWRBRK

PRESENT1
PRESENT2_B17
PRESENT2_B31

TRST
TCK
TDO
TDI
TMS

** Data lane names depend on variations*
x16-0 has no PCIe data lane switching.
x16-1 only switches lane zero, all other lanes always connected.
x4-4 only connects and switches first 4 lanes (Lane 5 upwards not connected).
x16-16 has switches on all data lanes.

Signal Groups

ALL	(Allows change of all signals at the same time)
LANE0	(Affect all signals relating to a specific PCIe lane)
…	
LANE15	(Lane groups differ on modules depending number of switches data signals.)
POWER	(All power supply pins)
PRESENT	(All present/mated pins)
JTAG	(All pins relating to the JTAG bus)
DATA	(All PCIe data pins)

Appendix 2 – Signals Supporting 'Monitoring'

Signal name	Side that can be monitored
SMCLK	Host and Drive
SMDAT	Host and Drive
PERST	Host and Drive
WAKE	Host and Drive
PWRBRK	Host and Drive