

Quarch Technology Ltd

Torridon SFF PCIe Drive Control Module

Technical Manual

For use with:

QTL1743 - Torridon PCIe SFF HS Drive Control Module

Using Quarch firmware version 4.0 and above

Friday, 04 April 2014



Change History

1.0	20 November 2013	Initial Release
1.1	4 March 2014	Corrected RESET signal name in appendix

Contents

Introduction	4
Technical Specifications	5
Switching Characteristics:	5
Mechanical Characteristics:	6
Control Interfaces	7
Basic Concepts	8
Signal Configuration	10
Power Up vs. Power Down Timing	11
Pin Bounce Modes	12
Glitch Control	14
Voltage Measurements	17
Default Startup State	18
Controlling the Module	19
Serial Command Set	19
Control Register Map	28
Appendix 1 - Signal Names	29

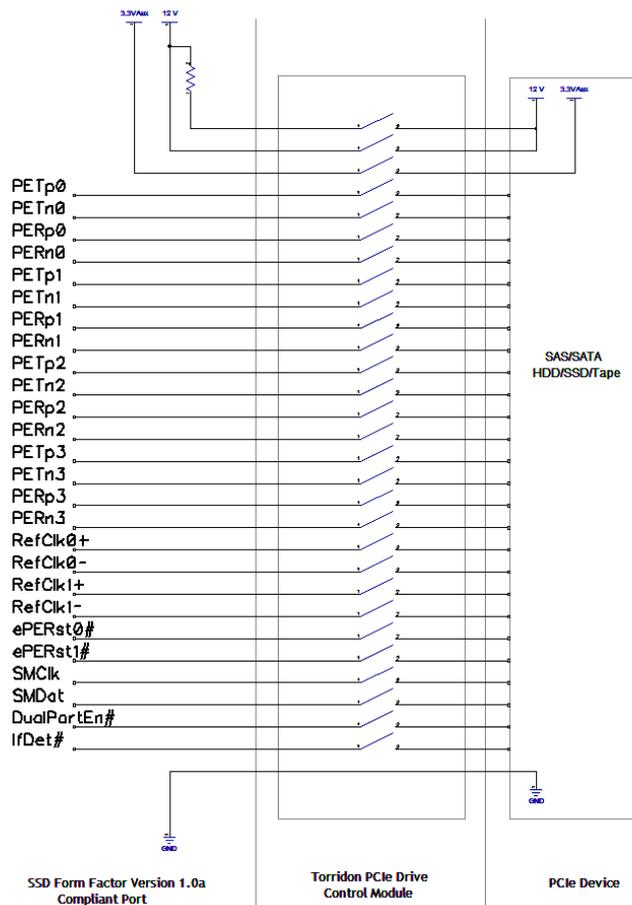
Introduction

The **Torridon PCIe SFF HS Drive Control Module** allows remote switching of power, drive presence, PCIe data pins, and sideband signals to a PCIe SFF Disk Drive whilst plugged into a host system.

Any standard drive compatible with the SFF-8639 can be controlled. Each set of pins can be individually switched, allowing complete control over the power up sequence of a drive. The switches can be sequenced at precise timings to simulate a hot-swap event, including pin bounce. Individual pins can be broken or glitched at any time to simulate a fault in the system.

The Control module switches the high speed PCIe data lines at speeds up to Gen 3 (8GT/s). This greatly increases the number of faults that can be injected into the system and produces a more comprehensive hot-swap.

Quarch modules may be customized to support other proprietary signals on request.



Technical Specifications

Switching Characteristics:

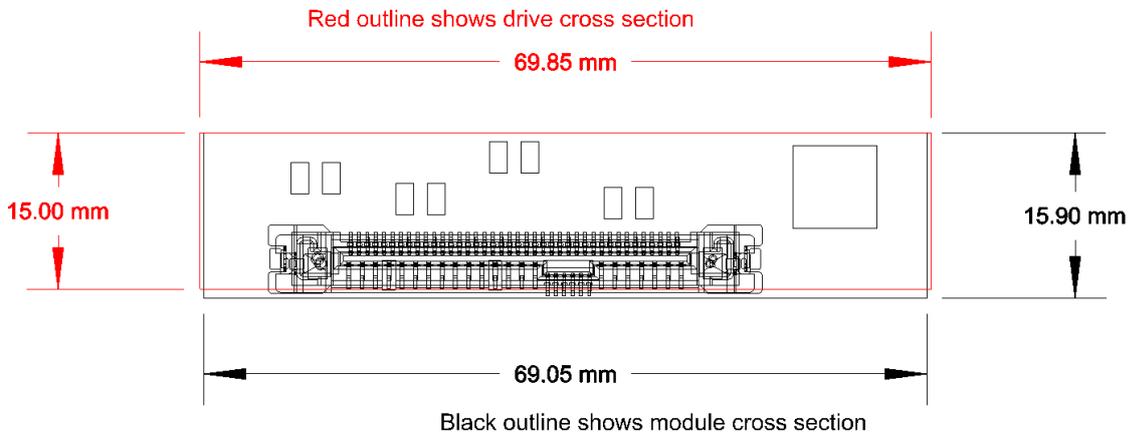
SFF8639 Connector Pin	Description	Switching Action
S1, S4, S7, S8, S11, S14, S16, S19, S22, S25, S28, P5, P6, P12, E9, E12, E15, E19, E22	Ground Pins	All connected to digital ground on the Module
S17, S18, S20, S21, S23, S24, S26, S27, E1, E2, E7, E8, E10, E11, E13, E14, E17, E18, E20, E21	PCIe Data Signals and Reference Clocks	Each signal is individually switched by a high speed RF switch
P14, P15	12V Power	Connected together and switched by 6.9A power FET
P13	12V Precharge	Switched by 6.9A power FET
E3	3V3 Aux Power	Switched by 0.7A FET
E4, E5	PCIe A & B Resets	Digitally Switched, Tri-State output
P4, E25	IFDet, DualPortEn	Digitally Switched, Open-Collector output
P10	PRSNT#	Always Connected (never driven by PCIe drive)
P11	Activity LED	Always Connected
E23, E24	SM Bus Clock and Data	Individually switched by bilateral analog switch
Signals below not used for PCIe devices		
P7, P8, P9	5V Power	Always Connected
E6, E16, S15, P1, P2, P3	Reserved	Always Connected
S2, S3, S5, S6, S9, S10, S12, S13	SAS/SATA/SATAe Data lanes	Never Connected

Mechanical Characteristics:

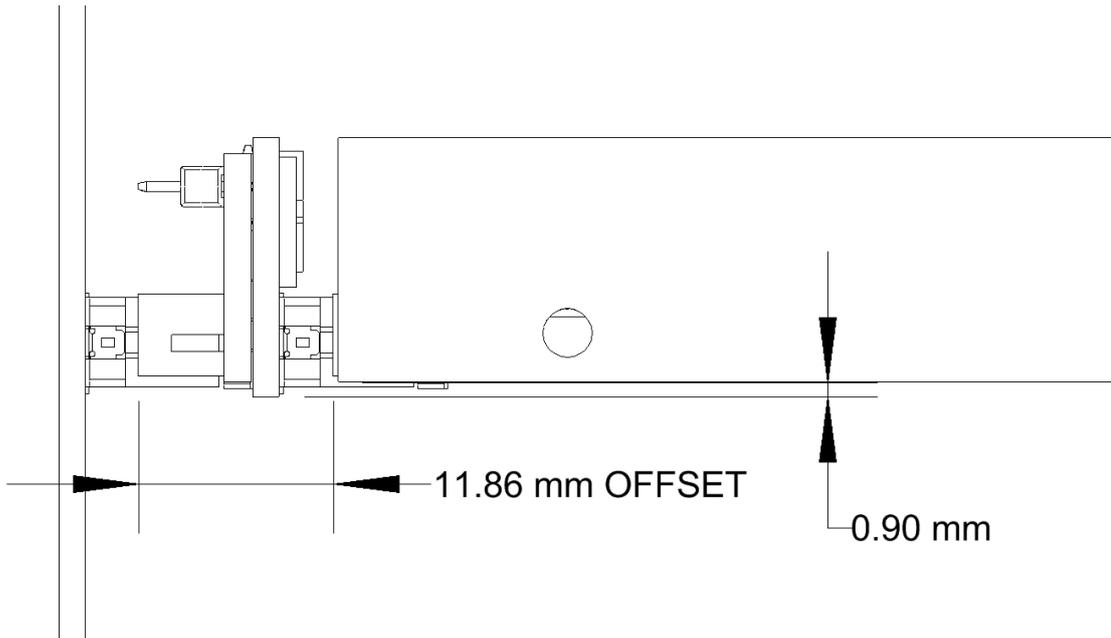
The module is designed to fit into a 2.5" SFF drive bay. The card does stick out 0.9mm below the allowed cross section of the drive, with the addition of the thickness of a drive carrier and general clearances within a chassis design this should not be a problem

A 3D Model of the module can be downloaded from:

<https://quarch.sharefile.com/d/sca73aa5cbc6497d8>



- The module will offset the drive by 11.86mm when installed.



Control Interfaces

All Torridon Control Modules are designed to be used with a Torridon Array Controller (QTL1461,QTL1079) or a single Torridon Interface Module (QTL1260).

The control cable is an ultra-thin Flex cable.

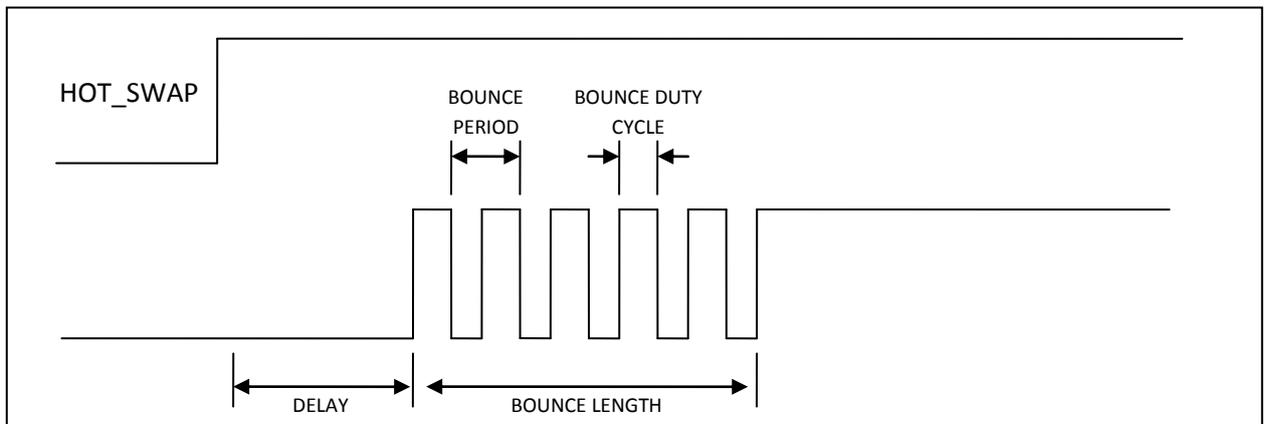
Control Interface	Form Factor	Torridon Module Ports	Control Methods Available	Interfaces
28 Port Torridon Array Controller	1U 19" Rack Mounted unit	24 at the front, 4 at the rear	Terminal Scripting TestMonkey 2 GUI	Serial via DB9 or RJ45 Ethernet
4 Port Array Controller	160x165x53mm Enclosure 1U Enclosure also available	4 ports on front	Terminal Scripting TestMonkey 2 GUI	Serial via RJ45 Ethernet USB
Torridon Interface Card (Legacy)	102mm x 26mm PCB	1 port	Terminal Scripting TestMonkey 2 GUI	Serial via DB9 or RJ45 USB
Torridon Interface Module	60mm x 45mm x 30mm Box	1 port	Terminal Scripting TestMonkey 2 GUI	Serial via RJ-45 Serial via USB/Serial convertor 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.

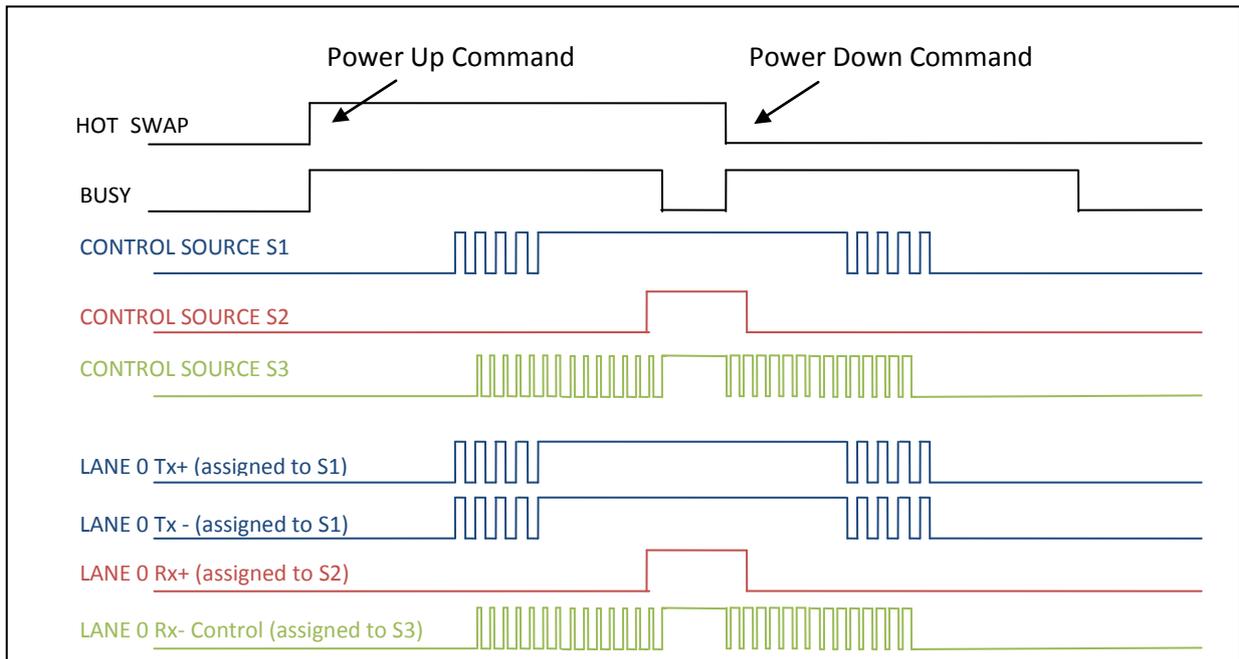
Control Source Parameters for a power up event (Basic Pin Bounce):



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:

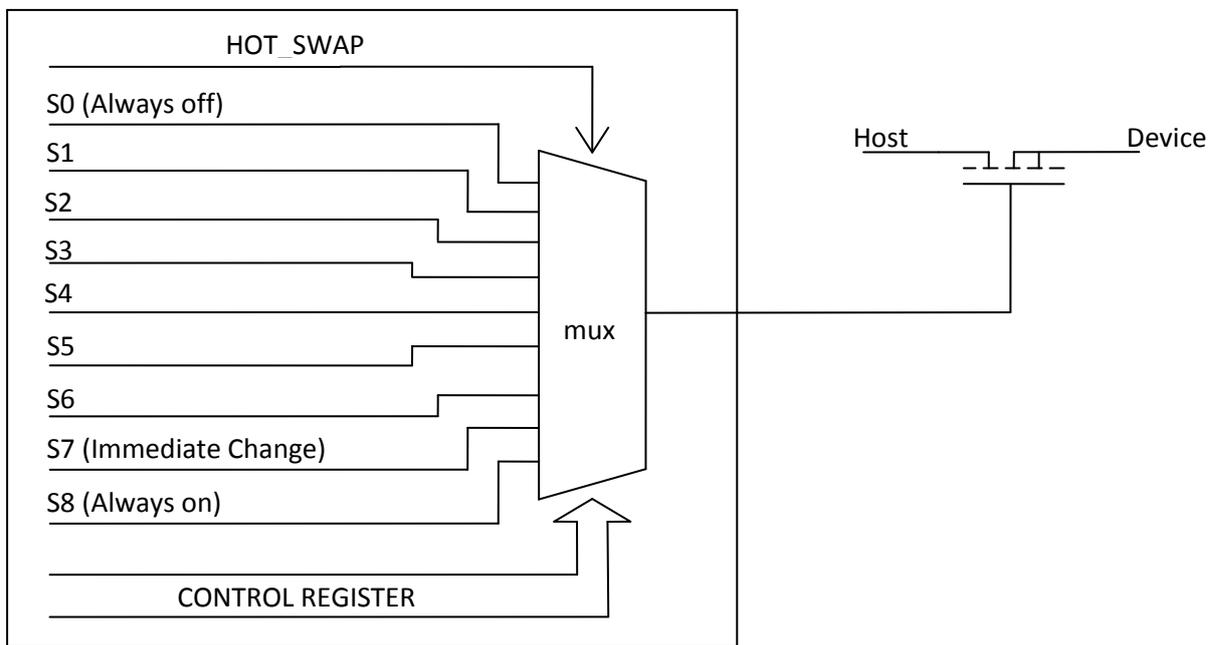


Signal Configuration

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:

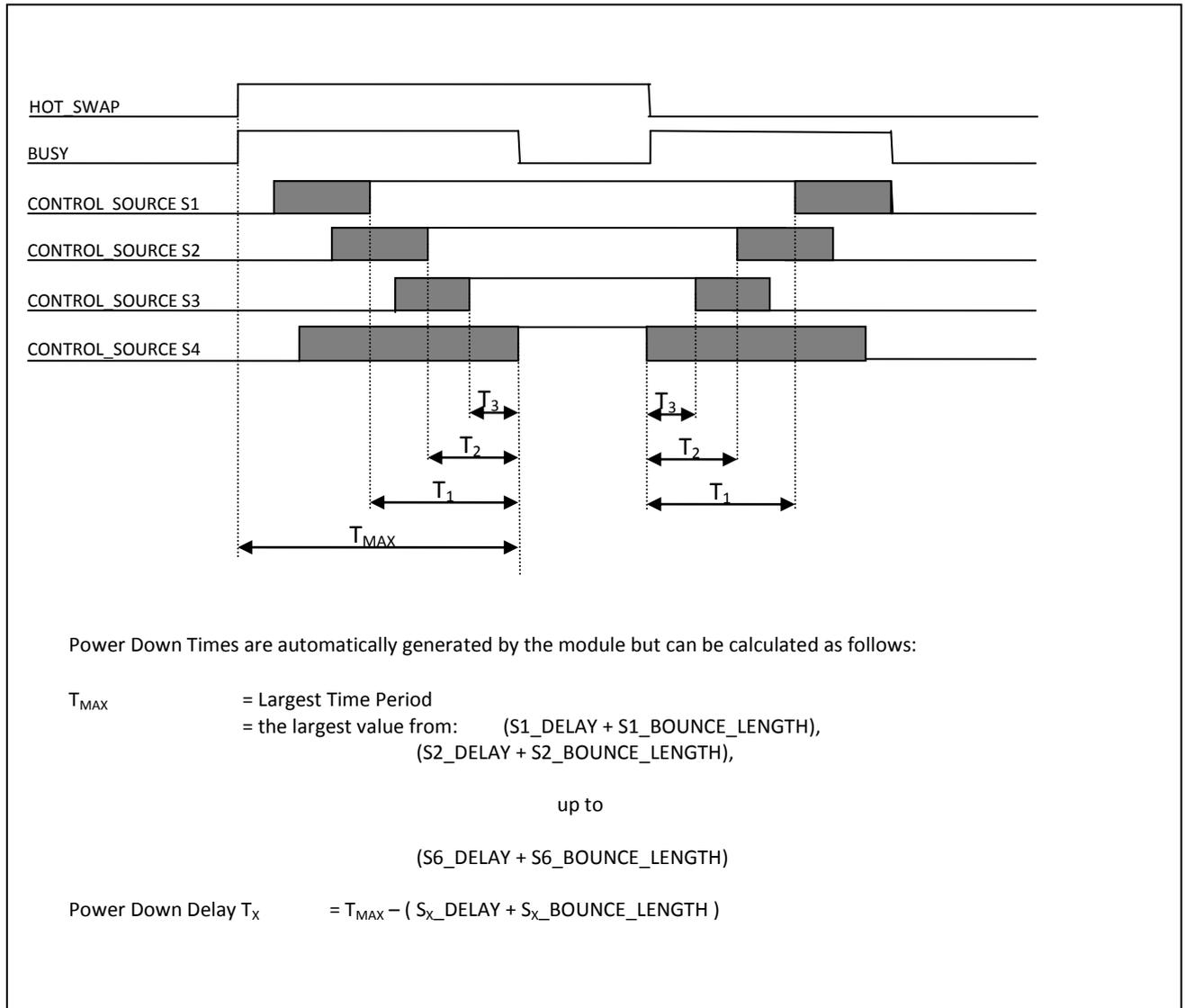
CONTROL_REGISTER Value	Description
0	Signal is always OFF
1	Signal assigned to control source 1
2	Signal assigned to control source 2
3	Signal assigned to control source 3
4	Signal assigned to control source 4
5	Signal assigned to control source 5
6	Signal assigned to control source 6
7	Signal changes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



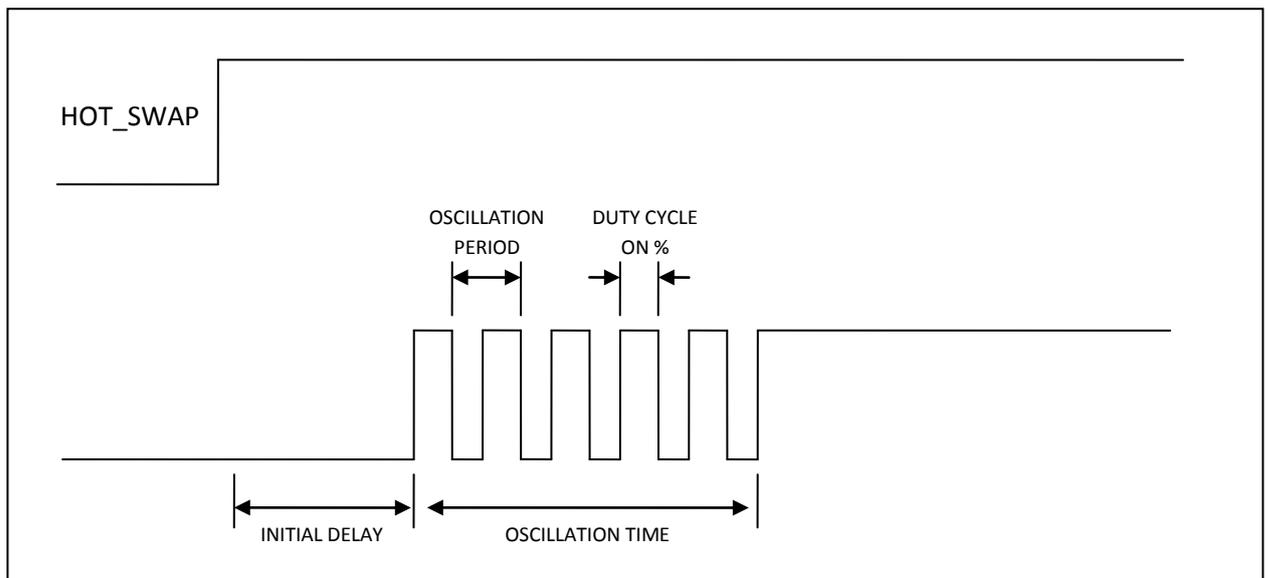
If you require a different power down sequence then you can alter any of the source timing values, pin bounce or signal assignments while the module is in the plugged state. When you initiate the 'pull' action, the new settings will be used.

Pin Bounce Modes

Pin Bounce can be set in two ways:

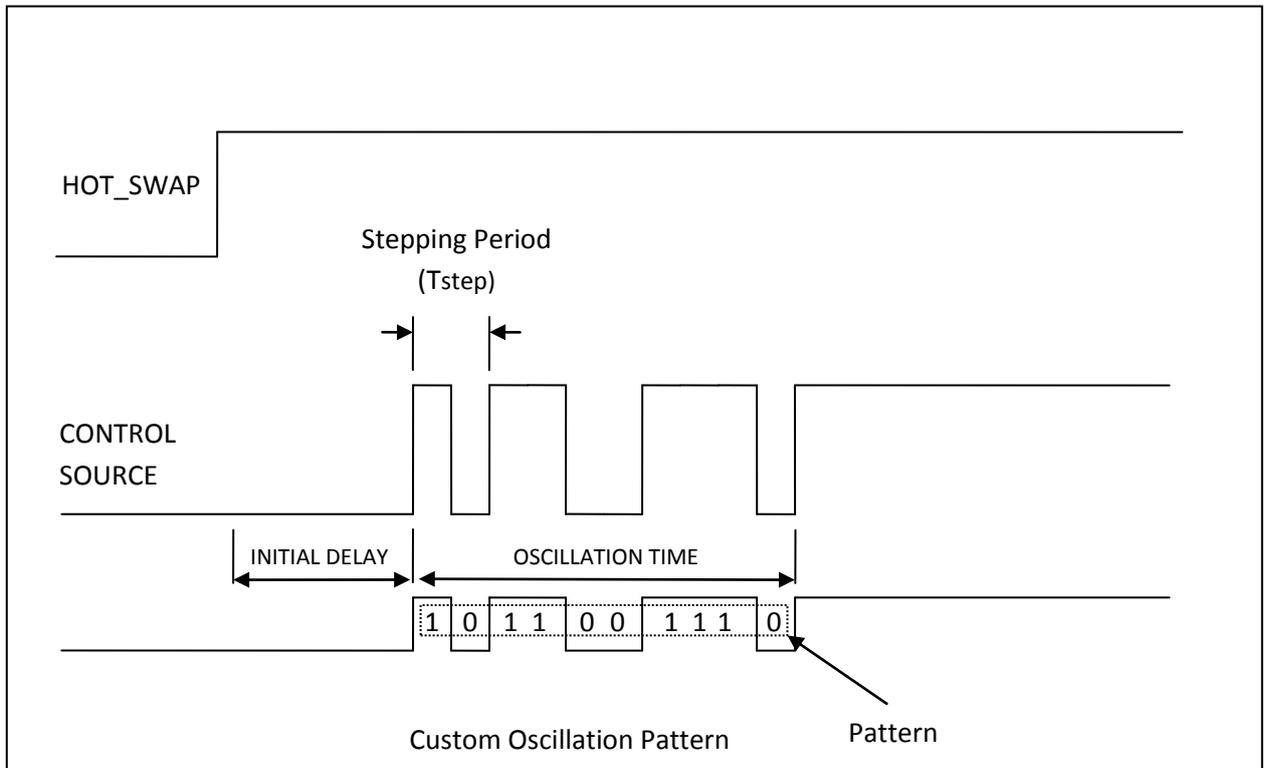
1. Basic Pin-Bounce (Constant Oscillation Frequency):

- The oscillation pattern length (Time) set in one of the two ranges:
 - 0 - 127 milliseconds in steps of 1mS
 - 0 - 1.27 seconds in steps of 10mS
- The bounce period is for the pattern (T_{osc}) is set on one of the 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 1-99%



2. User Pin-Bounce (Custom Oscillation):

- The oscillation pattern length (Time) set in one of the two ranges:
 - 0 - 127 milliseconds in steps of 1mS
 - 0 – 1.27 seconds in steps of 10mS
- The stepping period (T_{step}) 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 Custom pattern is described in 100 bits, where 2 bits are stepped through in each T_{step} period. The 100 bit pattern will loop if the oscillation pattern time is longer than the available pattern.



Glitch Control

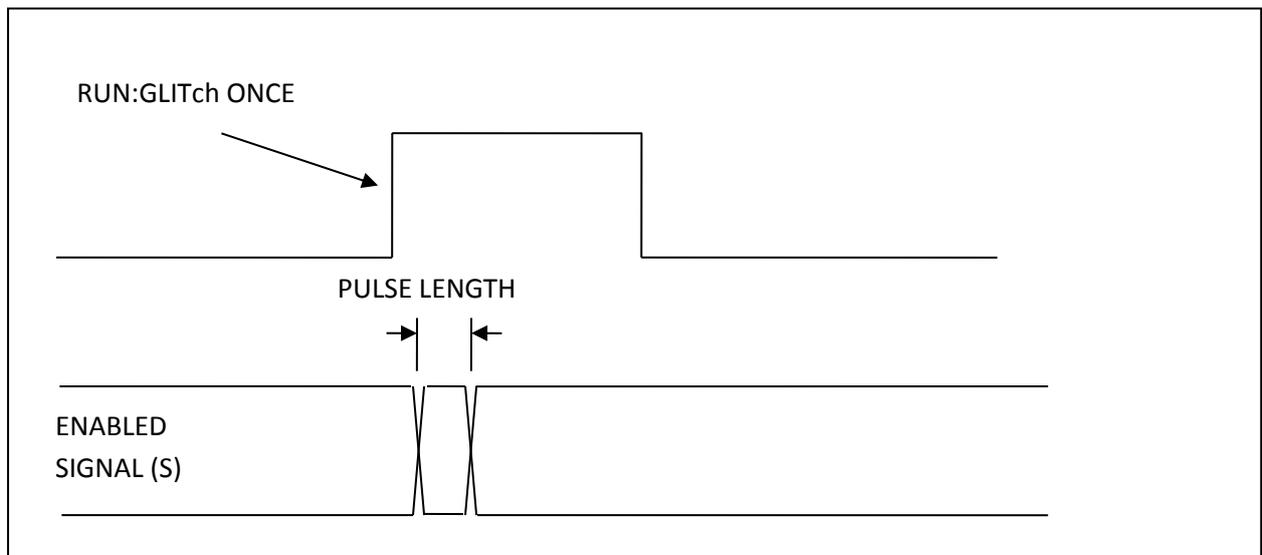
Any control signal may be glitched for a pre-determined length of time using the glitch generator logic.

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:



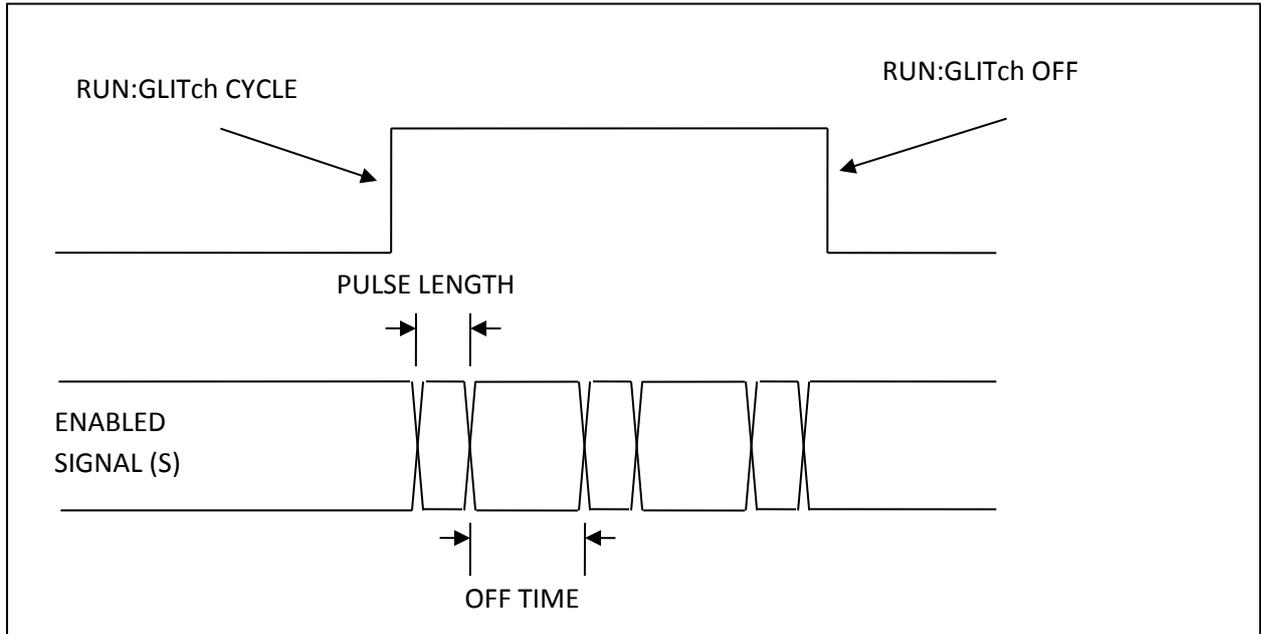
A single glitch is generated when the **RUN:GLITCh ONCE** command is executed

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

$$\text{PULSE LENGTH} = \text{GLITCh:MULTiplier} \times \text{GLITCh:LENgth}$$

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.

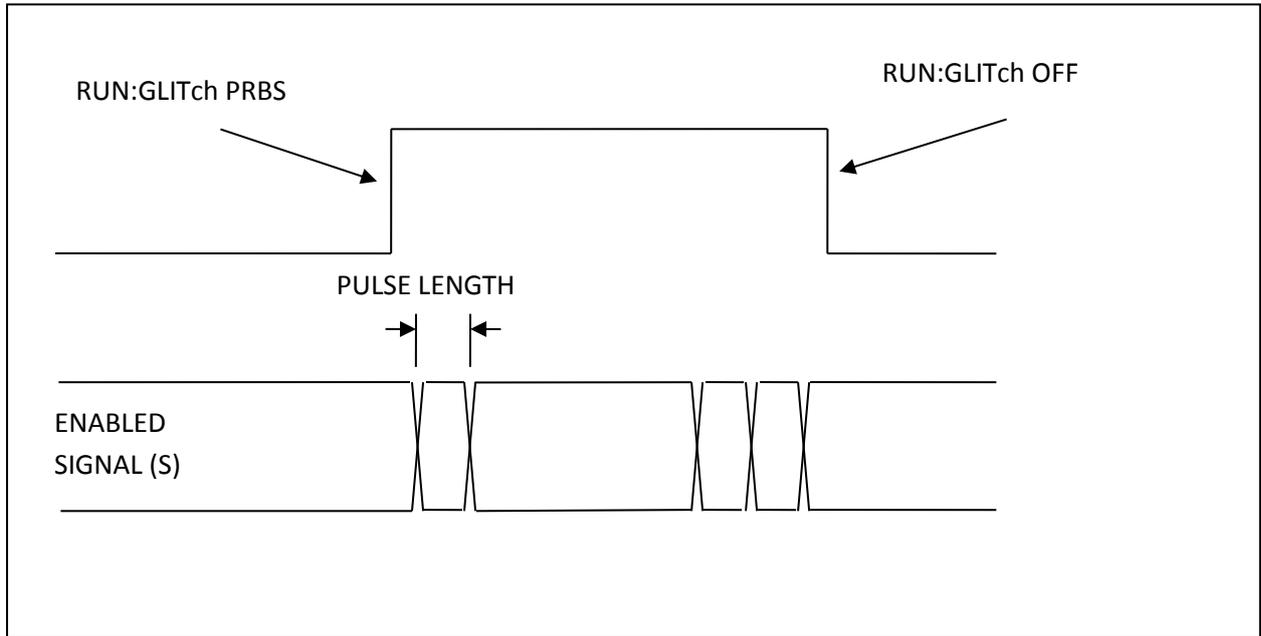
The length of the glitch is determined by using the **GLITCh:SETup** command or the **GLITCh:MULTiplier** and **GLITCh:LENgth** commands:

$$\text{PULSE LENGTH} = \text{GLITCh:MULTiplier} \times \text{GLITCh:LENgth}$$

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:

$$\text{OFF TIME} = \text{GLITCh:CYCle:MULTiplier} \times \text{GLITCh:CYCle:LENgth}$$

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:

$$\text{PULSE LENGTH} = \text{GLITCh:MULTIplier} \times \text{GLITCh:LENGTh}$$

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.

Voltage Measurements

The modules are capable of measuring various voltages both for self test and to assist in the testing of a customer's system. The following measurement points are available:

Measurement Command	Description	Resolution / Accuracy
MEASure:VOLTage:SELF 3v3?	Returns the voltage of the modules internal 3.3v rail	12mV / 3%
MEASure:VOLTage:SELF 5v?	Returns the voltage of the modules internal 5v power rail	15mV / 3%
MEASure:VOLTage 12vin?	Returns the voltage of the 12V power pins on the backplane (unswitched) side of the Module	46mV / 3%
MEASure:VOLTage 12vout?	Returns the voltage of the 12v power pins on the drive (switched) side of the module	46mV / 3%
MEASure:VOLTage 12vin_chg?	Returns the voltage of the 12V charge pin on the backplane (unswitched) side of the Module	46mV / 3%
MEASure:VOLTage 12vout_chg?	Returns the voltage of the 12v charge pin on the drive (switched) side of the module	46mV / 3%

Default Startup State

On power up or reset, the control modules enter a default state. To make the module as easy to use as possible, the default state is a ‘standard’ hot-swap scenario with preset source and signal settings such that the “run:power up” command will immediately power up the drive without needing any initial setup.

The default hot-swap scenario will connect pre-charge then power then pins, each step with a 25mS delay. All sources are enabled.

Source Number	Source Enabled	Initial Delay
1	YES	0mS
2	YES	25mS
3	YES	50mS
4	YES	0mS
5	YES	0mS
6	YES	0mS

Signal	Assigned Source
IF_DETECT	Source 1
12V_CHARGE	Source 2
All other signals	Source 3

Hot-Swap State:

Drive is in the ‘plugged’ state, waiting for a “**RUN:POWER DOWN**” command to remove 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). 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

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:DEFault:STATE

Resets the state of the module. This will set all source/signal/glitch etc logic to its default power-on values. Terminal setting will not be affected. This command allows the module to be brought back to a known state without resetting it.

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”.

MEASure:VOLTage [12vin? | 12vout? | 12vin_chg? | 12vout_chg?]

Returns the voltage on the specified rail in mV. Vin refers to the upstream or host side of the card, and Vout refers to the switched, drive side. Values are returned in the form “3300mV”.

MEAS:VOLTage:SELF [3v3?,5v?]

Returns the self test voltages. These are measurements of voltage rails required for correct operation of the module. The values are returned in the form “5000mV”

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).

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

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

Sets a given signal to a numbered timing source (0-8). SIGNAL_NAME is one of the items in the 'Signal Names' Appendix at the end of this manual.

SIGnal:[SIG_NAME]:SOURce?

Returns the source number that the signal is assigned to.

SIGnal:[SIG_NAME | ALL]:GLITch:ENABle [ON | OFF]

SIGnal:[SIG_NAME | ALL]:GLITch:ENABle?

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

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

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 [MULTIPLIER_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 [MULTIPLIER_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|500ns|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]

Control Register Map

This map is provided for backwards compatibility with old modules only. While you can use the 'Read' and 'Write' commands, we **STRONGLY** recommend you use the SCPI based 'Signal' and 'source' commands. Please contact Quarch for the full register map.

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

12V_POWER
12V_CHARGE
3V3_AUX
SM_CLK
SM_DAT
EPE_RST_0
EPE_RST_1
REF_CLK_0_PL
REF_CLK_0_MN
REF_CLK_1_PL
REF_CLK_1_MN
PETP_0

PETN_0
PERT_0
PERN_0
PETP_1
PETN_1
PERT_1
PERN_1
PETP_2
PETN_2
PERT_2
PERN_2
PETP_3
PETN_3
PERT_3
PERN_3

(Transmit Plus on lane 0) is the output data from the backplane to the drive, on the +ve side of the differential pair.

Signal Groups

ALL	(Allows change of all signals at the same time)
PORT_A	(All signals for port A: 8x Data signals plus REF_CLK and EPE_RST signals)
PORT_B	
DATA_A	(All data signals for port A, PCIe data lanes only)
DATA_B	
CLK_A	(Ref clock signals for port A)
CLK_B	
LANE0	(All 4 PCIe data signals that make up Lane 0)
LANE1	
LANE2	
LANE3	
DRIVE_POWER	(All power and pre-charge signals)
SM_BUS	(All SM BUS signals)