Quarch Technology Ltd

Torridon Quad QSFP Cable Pull Module

Technical Manual

For use with:

QTL1663 - Torridon Quad QSFP Cable Break Module

Using Quarch firmware version 4.0 and above

Tuesday, 30 July 2013

Change History

1.0	10th May 2013	Initial Release

Contents

Introduction

The **Torridon Quad QSFP Cable Pull Module** allows remote switching of the data, power and management interface signals on a QSFP or QSFP+ Transceiver for test automation or fault injection purposes.

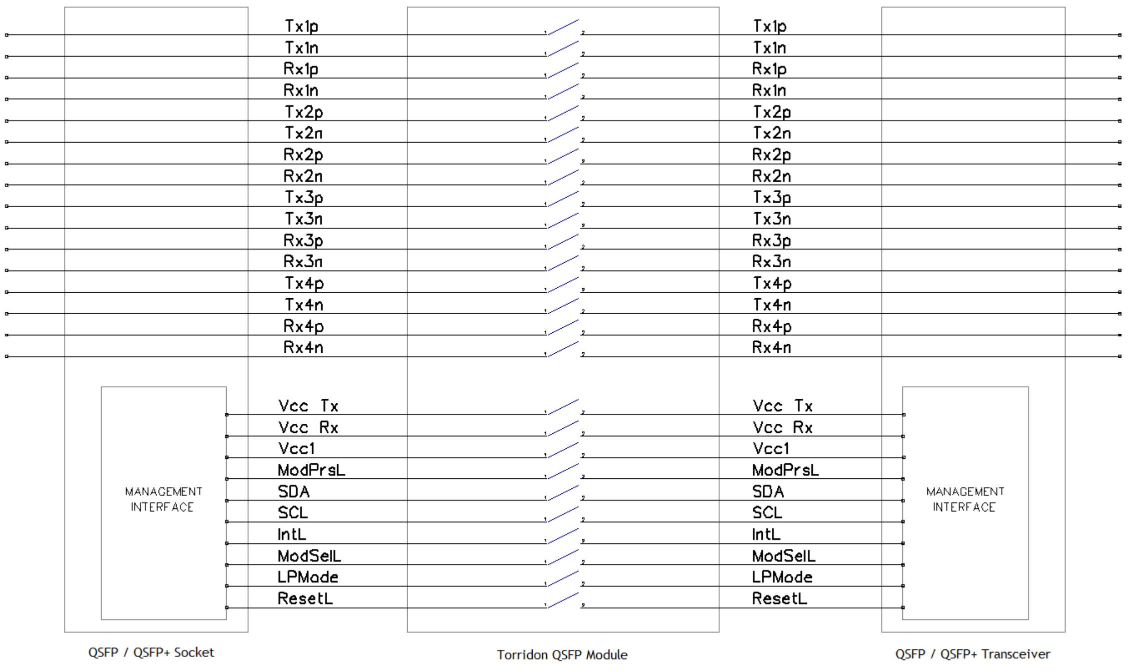
The module is functionally identical to the QTL1366 QSFP Cable Pull Module, but fits into a standard Quad QSFP cage, allowing control over all 4 cables (Where there would be insufficient space to fit 4 individual QTL1366 modules).

The data path is completely passive and therefore the module will work with any signal protocol. It has been qualified for data rates up to 6Gb/s, but is likely to perform at up to 12Gb/s.

Each pin is individually switched, allowing complete control over the mating sequence of the transceiver.

The module can switch and disrupt all of the power supplies to the transceiver; it can also disrupt and replace data being read back across i2c.

The switches can be sequenced at precise timings to simulate a hot-swap event, including pin bounce. Individual pins can also be broken or glitched at any time to simulate a fault in the system.



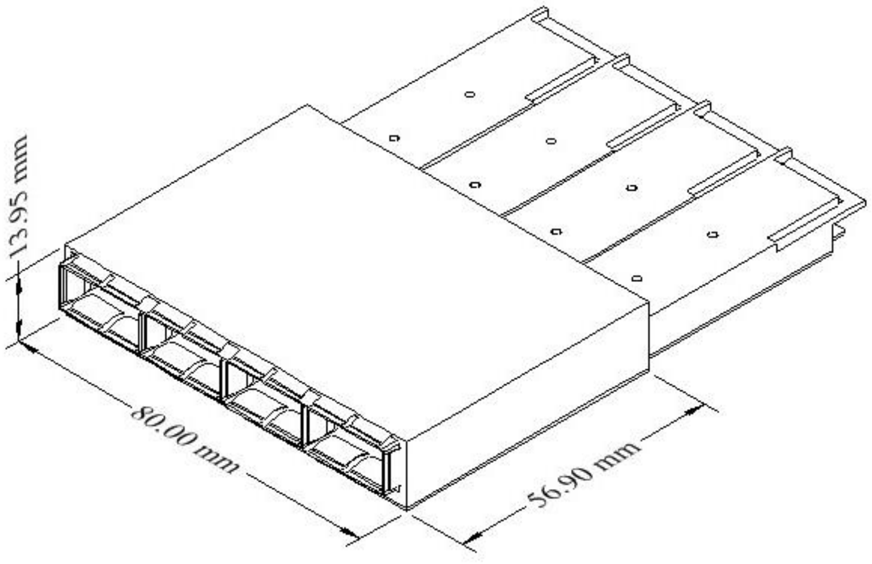
Technical Specifications

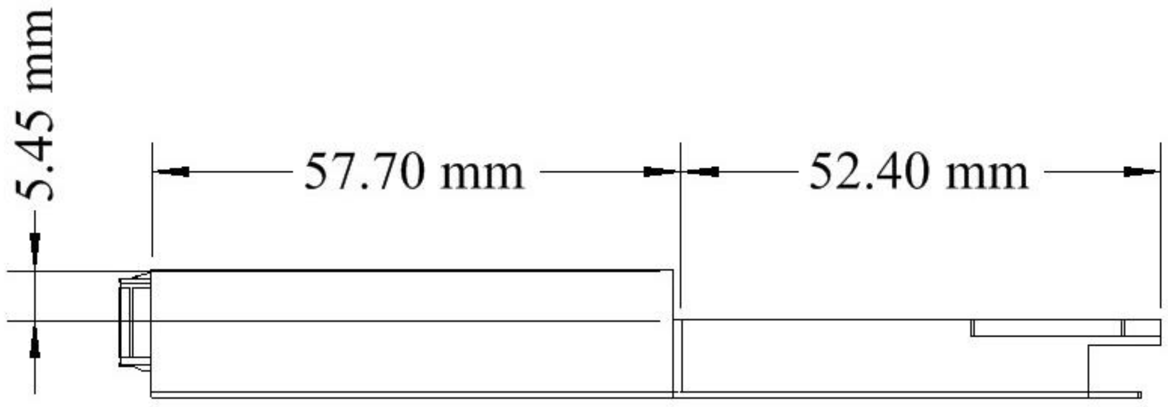
Switching Characteristics:

QSFP Connector Pin	Description	Switching Action
1, 4, 7, 13, 16, 19, 20, 23, 26, 32, 35, 38	Ground pins	All connected to Ground on the Module
2, 3, 5, 6, 14, 15, 17, 18, 21, 22, 24, 25, 33, 34, 36, 37	SAS Data Signal pins	Each signal is individually switched by an RF Switch
1, 29, 30	Transceiver Power	Each power supply is individually switched by a 2A power FET
11, 12	TWI Bus	Each signal is individually switched by a digital switch, data read from the transceiver can be altered by the module
8, 9, 27, 28, 29, 30, 31	Other Management Interface	Each signal is individually switched by an FPGA

Mechanical Characteristics:

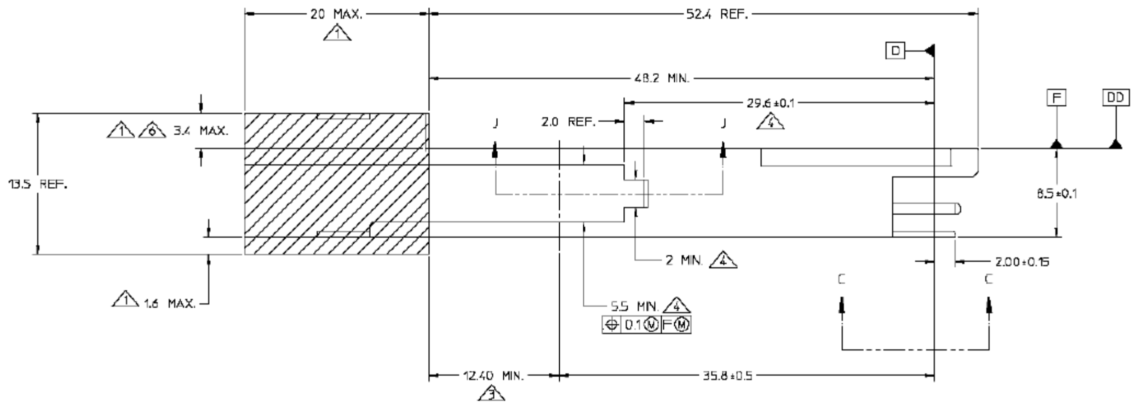
QTL1663 Quad QSFP Module Dimensions





QTL1663 Dimensional Non Conformities:

QSFP Dimensional Specification:



The height and length of the exterior part of a QSFP are limited to 3.4mm above the top surface of the QSFP and 20mm outside of the cage, the QTL1663 module breaks these specifications. The additional height of the module may cause mechanical interference if there is limited clearance around the QSFP port.

There is no precise standard for quad QSFP cages. All ganged QSFP cages currenlty found on the market have a 19mm spacing so compatibility should not be an issue. The following parts were checked:

Compatible 1x4 Cages:

• TE Connectivity (Tyco)	All 1x4 Cages
• Amphenol	All 1x4 Cages
• FCI	No quad cage found
• Molex	All 1x4 Cages

Incompatible Cages:

- None found

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).

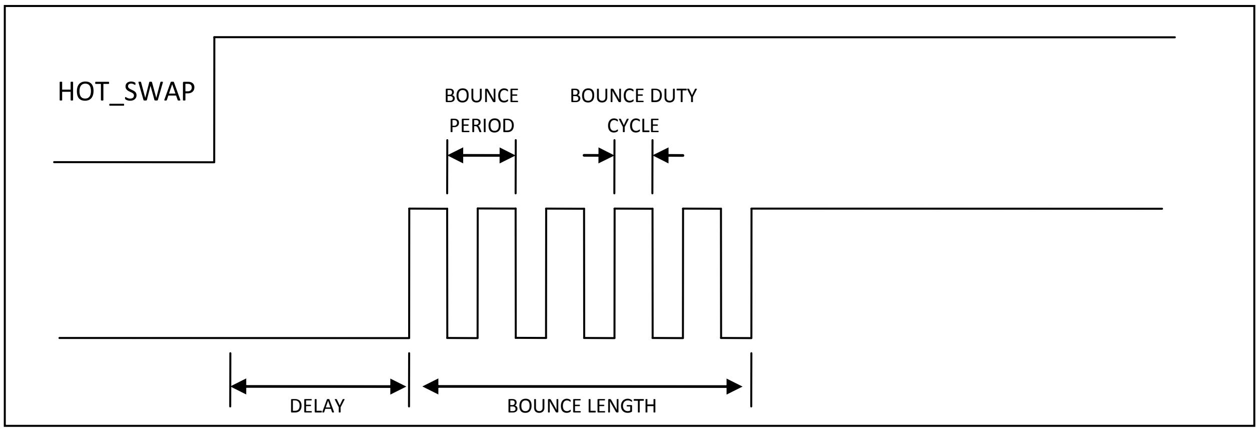
Control Interface	**Form Factor**	**Torridon Module Ports**	**Control Methods Available**	**Interfaces**
28 Port Torridon Array Controller	1U 19" Rack Mounted unit	24 at the front, 4 at the rear	Terminal Scripting TestMonkey 2 GUI	Serial via DB9 or RJ45 Ethernet
4 Port Array Controller	160x165x53mm Enclosure 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.

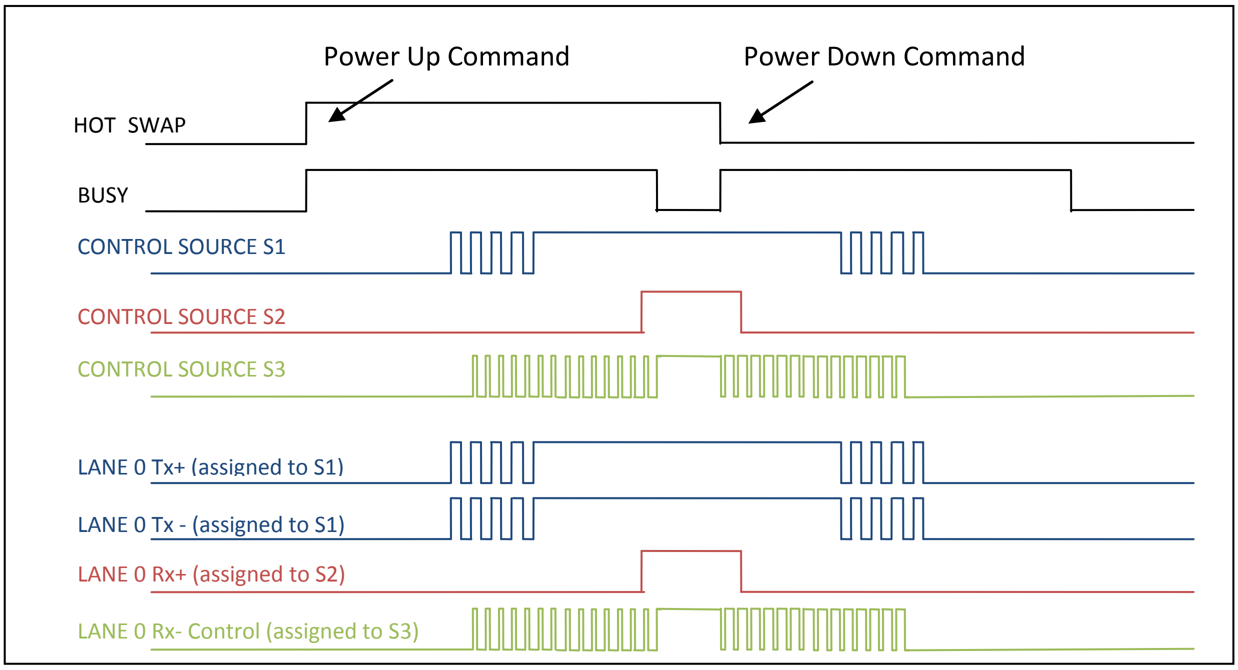
Control Source Parameters for a power up event (Basic Pin Bounce):



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:

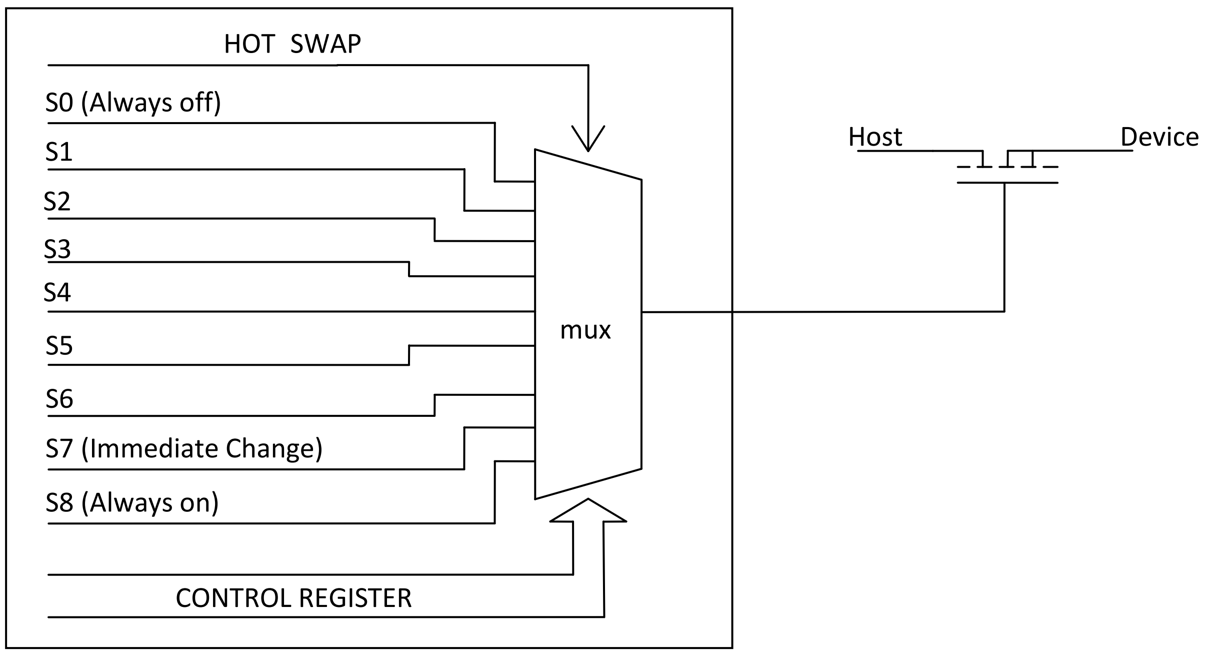

Power Up Command
Power Down Command
HOT SWAP
BUSY
CONTROL SOURCE S1
CONTROL SOURCE S2
CONTROL SOURCE S3
LANE 0 Tx+ (assigned to S1)
LANE 0 Tx - (assigned to S1)
LANE 0 Rx+ (assigned to S2)
LANE 0 Rx- Control (assigned to S3)

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:

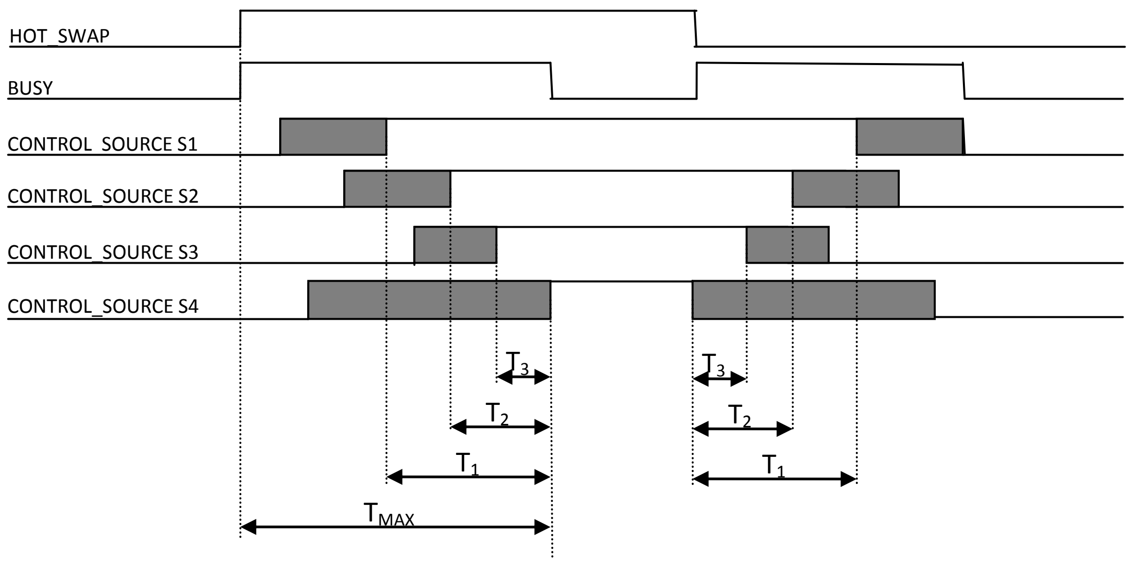
CONTROL_REGISTER Value	Description
0	Signal is always OFF
1	Signal assigned to control source 1
2	Signal assigned to control source 2
3	Signal assigned to control source 3
4	Signal assigned to control source 4
5	Signal assigned to control source 5
6	Signal assigned to control source 6
7	Signal changes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

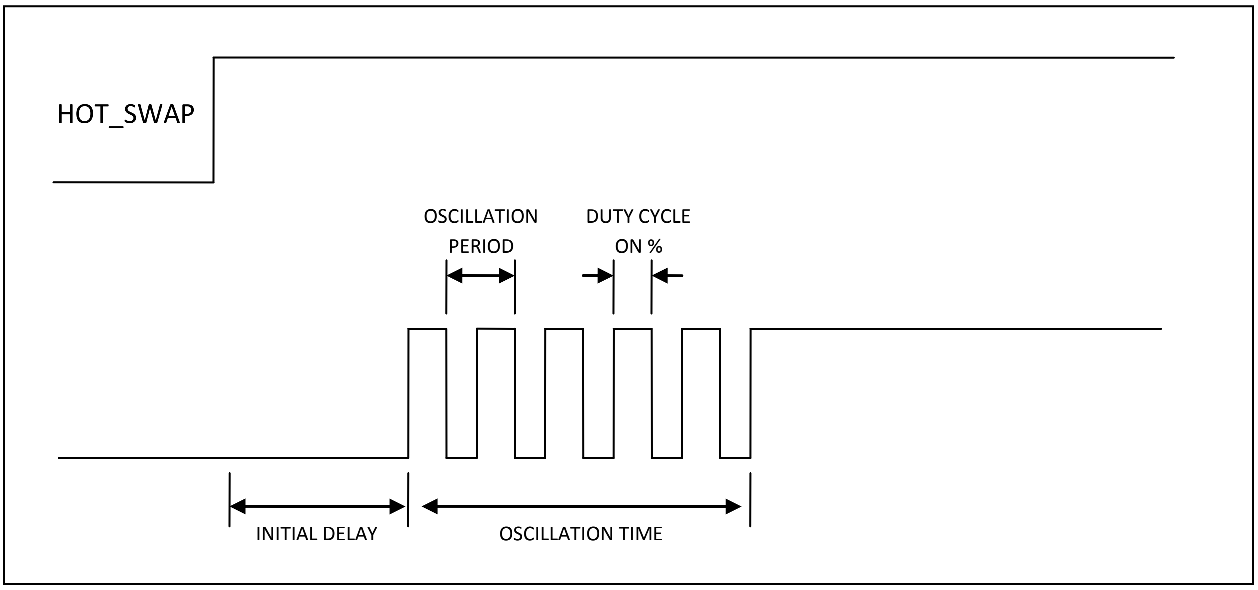
Power Down Delay T_X = T_{MAX} – (S_X_DELAY + S_X_BOUNCE_LENGTH)

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.

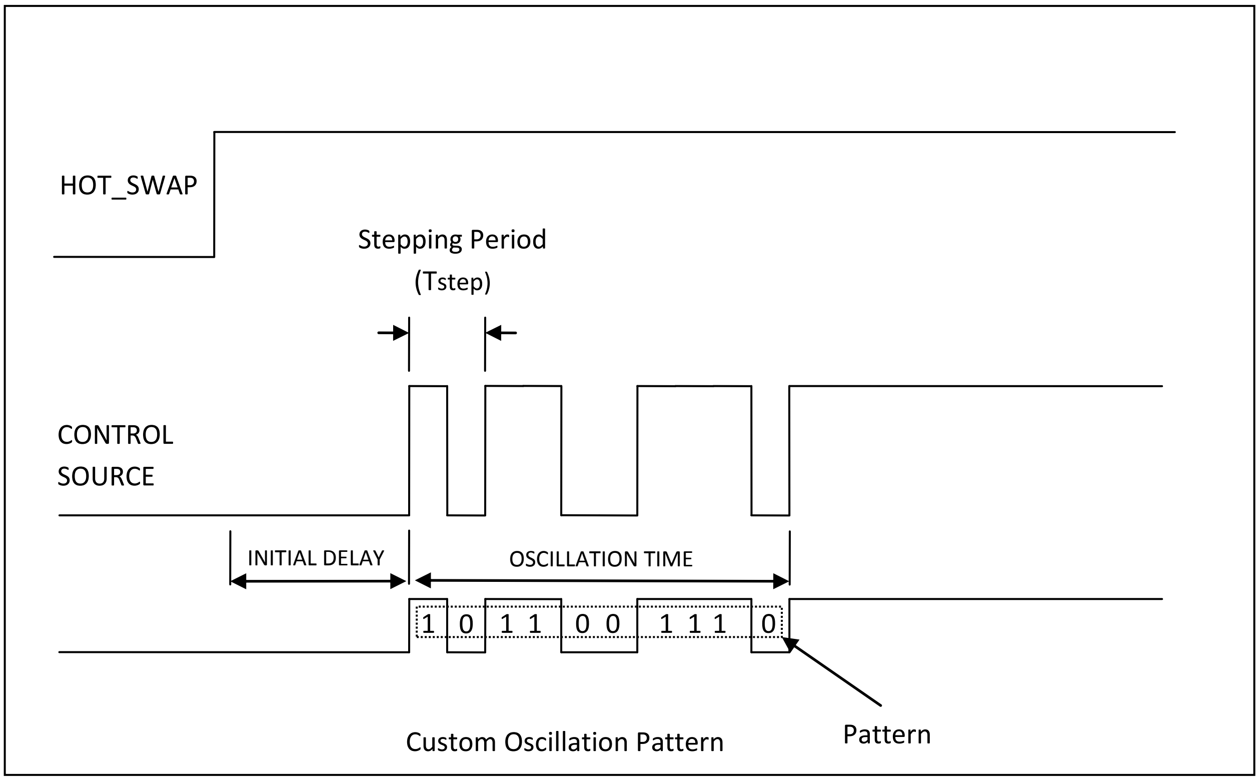
Pin Bounce Modes

Pin Bounce can be set in two ways:

1. Basic Pin-Bounce (Constant Oscillation Frequency):
   - ➢ The oscillation 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 (Tstep)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 Custom pattern is described in 100 bits, where 2 bits are stepped through in each Tstep period. The 100 bit pattern will loop if the oscillation pattern time is longer than the available pattern.



Glitch Control

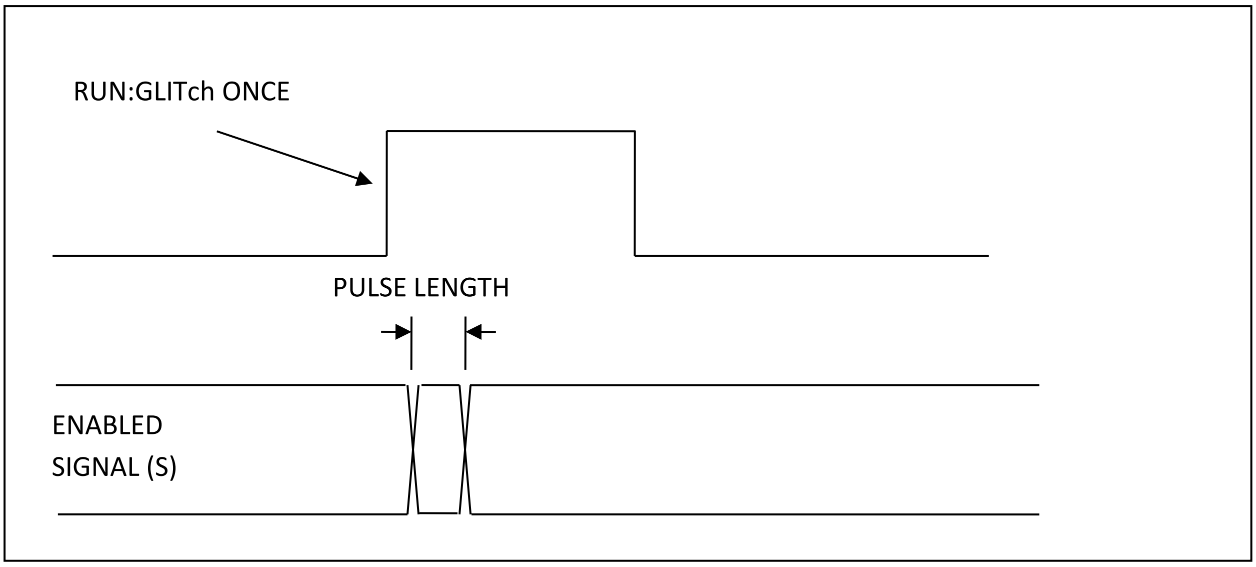
Any control signal may be glitched for a pre-determined length of time using the glitch generator logic.

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:



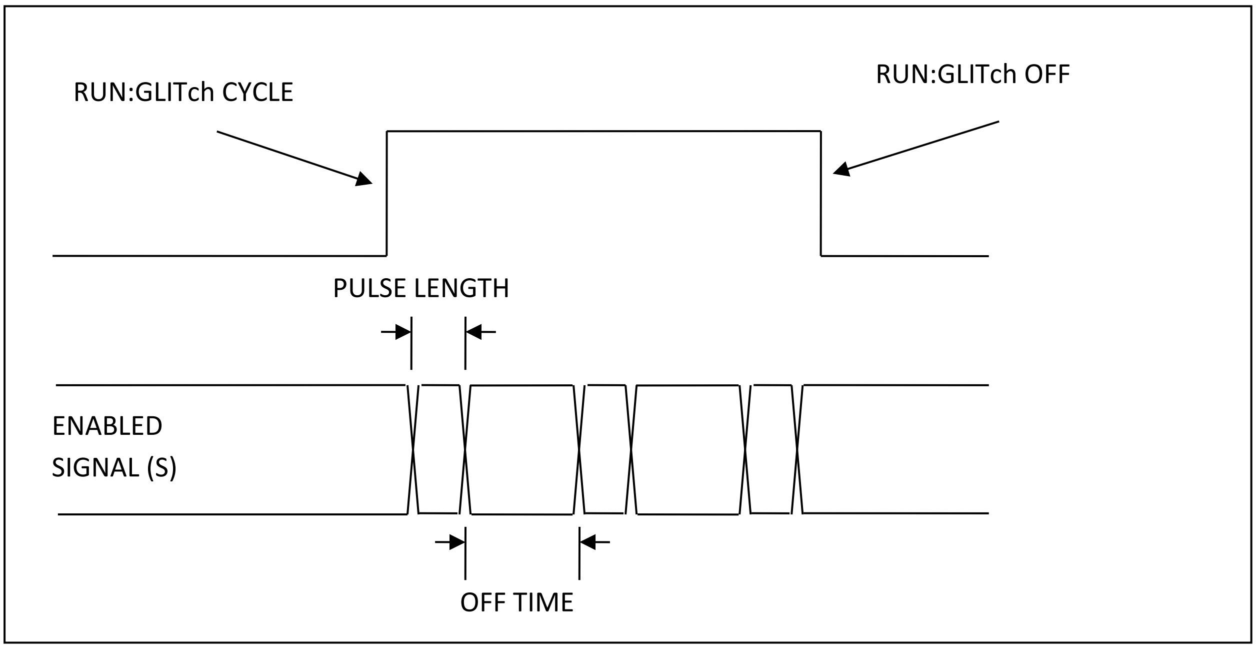
A single glitch is generated when the **RUN:GLITch ONCE** command is executed

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

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.

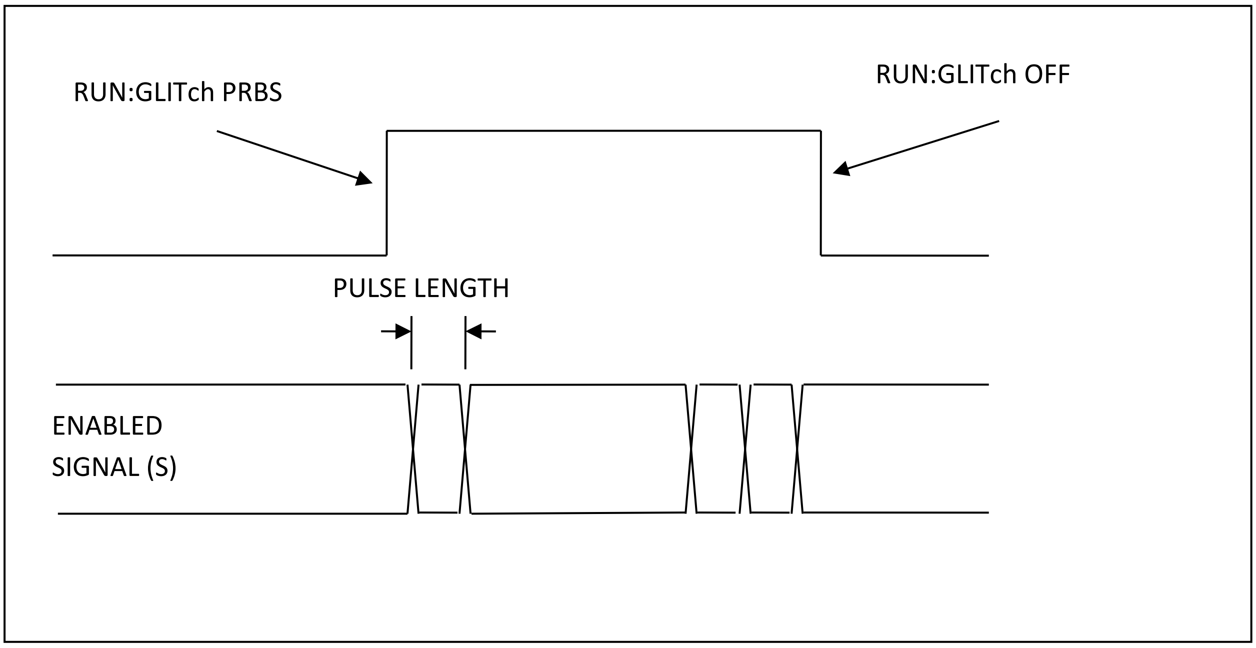
The length of the glitch is determined by using the **GLITch:SETup** command or the **GLITch:MULTiplier** and **GLITch:LENgth** commands:

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.

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 power rail	64mV / 5%
MEASure:VOLTage:SELF 5v?	Returns the voltage of the modules internal 5v power rail	64mV/ 5%

Management Interface – Serial Bus Override

The module can intercept and modify data read from the cable management interface. This is done by overriding the contents on individual registers. The actual cable is not affected. The QSFP module intercepts the EEPROM read command and returns the overridden data.

The user can select a register and set an override value. Any attempt to read this register via the management interface will now return the modified register value. Any number of registers can be modified.

EEPROM override follows the QSFP standard for the memory map. This has registers 0 – 127 and an additional paged area from 128 – 255. Valid page numbers are 0 – 3.

To override the contents of address 15 (Non paged area, so page is set to 0 as default) with value 0xFF:
CABle:[1-4]:OVERride 0 15 0xFF

To revert address 15 back to its normal value:
CABle:[1-4]:REVert 0 15

See the command listing for all commands relating to this function

Default Startup State

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

The default hot-swap scenario will disconnect all pins immediately, without delays or pin-bounce.

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
Px_VCC_TX	Source 1
Px_VCC_RX	Source 1
Px_VCC_1	Source 1
All other signals	Source 2

Hot-Swap State:
The cable 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). 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: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".

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]

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.

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

CABle:[1-4]:OVERridden?

Returns a list of modified (overridden) EEPROM registers

CABle: [1-4]:REVert ALL

Resets all overridden registers. All register reads will return the data from the cable EEPROM.

CABle: [1-4]:REVert [#page] [0xAA]

Resets a single overridden register

#page = EEPROM page (0 – 3). When accessing data below the paged area, set page to 0
0xAA = Address of the register to reset the override

CABle: [1-4]:OVERride [#page] [0xAA] [0xDD]

Sets on EEPROM override on the given register. This new value will now be returned if the EEPROM register is read by the host.

#page = EEPROM page (0 – 3). When accessing data below the paged area, set page to 0
0xAA = Address of the register to set the override
0xDD = Data value for the overridden register

Control Register Map

Access to the FPGA registers should not be required for the majority of operations and customers are encouraged to use the high level commands in order to maintain compatibility with future firmware versions. Please contact Quarch if you require the full 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

Signal/Group names are identical to the Single QSFP module, but with the addition of a ‘port prefix’: ***Px*** where x is a number from 1-4.

Signals

Px_TX1_PL	(Data transmitted from the ‘input’ port on Lane 0 (+ve side of differential pair)
Px_TX1_MN	
Px_RX1_PL	(Data received at the ‘input’ port on Lane 0 (+ve side of differential pair)
Px_RX1_MN	
Px_TX2_PL	
Px_TX2_MN	
Px_RX2_PL	
Px_RX2_MN	
Px_TX3_PL	
Px_TX3_MN	
Px_RX3_PL	
Px_RX3_MN	
Px_TX4_PL	
Px_TX4_MN	
Px_RX4_PL	
Px_RX4_MN	
Px_LPMODE	
Px_RESETL	
Px_INTL	
Px_VCC_TX	
Px_VCC_RX	
Px_VCC_1	
Px_MODPRSL	
Px_MODSELL	
Px_SDA	
Px_SCL	

Signal Groups

Px_ALL	(Allows change of all signals at the same time)
Px_LANE1	(Affect all signals relating to a specific SAS lane)
Px_LANE2	
Px_LANE3	
Px_LANE4	

Px_MANAGEMENT (Controls all signals related to the active cable management interface)
Px_POWER (Controls all signals related to QSFP power)

Cable Names

As there are now 4 active cable EEPROMs that can be overridden, the cable name must be used in each command. Cables are numbered 1-4 so to access EEPROM on cable 3, the command might be:
CABle:3:REVert ALL