Quarch Technology Ltd

Torridon USB 3.0 Physical Layer Switch

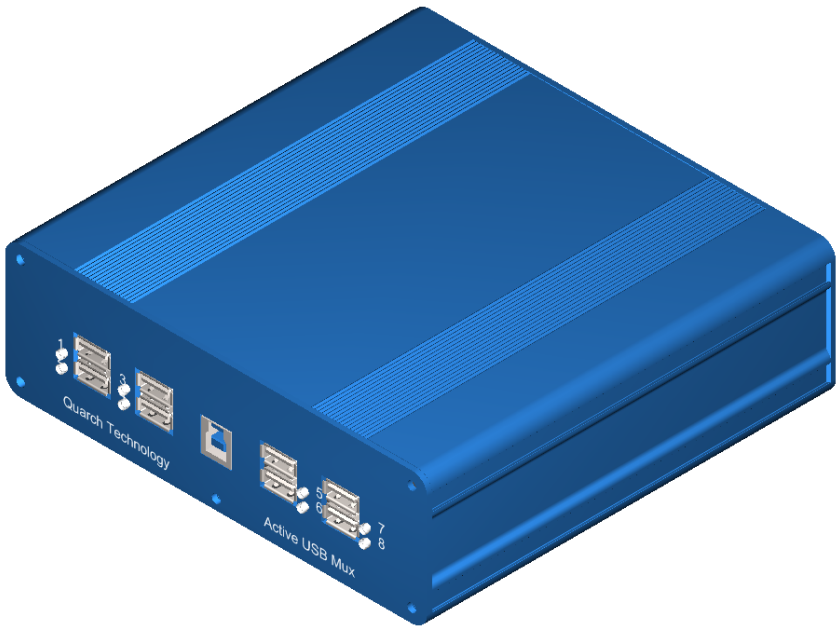
Technical Manual

For use with:

QTL1443 - Torridon USB 3.0 Physical Layer Switch

Using -04A or later hardware
Using Quarch firmware version 4.100 and above

Tuesday, 05 November 2013



Change History

1.0	1st June 2011	Initial Release
1.1	17th June 2011	Added missing commands
1.2	17th September 2013	Altered for new hardware design, using passive switches. Added new commands relating to added cable-pull functions
1.3	5th November 2013	Minor changes to some description sections Mux commands moved to main command list

Contents

Introduction

The **Torridon USB 3.0 Physical Layer Switch** has 1 USB 3.0 Host port and 8 USB 3.0 Device ports, all ports can transmit USB 3.0 data at speeds up to 5Gb/s.

The user can set up a bi-directional connection between host A and any of the Device ports as if they were cabled together. The switch uses a USB 3.0 re-driver IC to boost signal amplitude but it should be transparent to host and device.

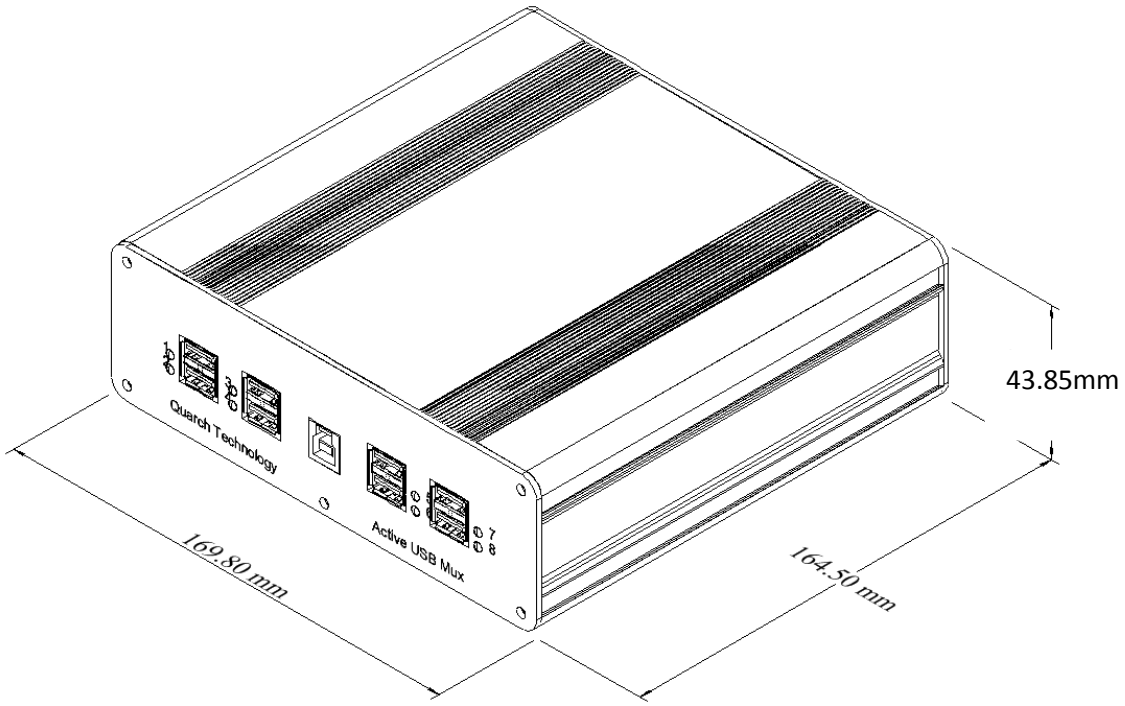
This is different from earlier revisions of the product (hardware revisions -01A through -03A), which used an active cross-point switch.

The -04A hardware also now includes the functionality of the QTL1309 USB Cable-Pull module. This circuitry provides full control of the sequencing of each signal and adds extra test options such as fault injection, pin bounce and glitching.

Technical Specifications

Mechanical Characteristics:

QTL1443 USB Physical Layer Switch Module



Basic Concepts

The module can switch one USB 3.0 B (Host) port to any one of 8 USB 3.0 A (Device) ports. The host system will see the USB device as if it is attached normally.

While the signals are passively switched in the -04A version of the product, the addition of the extra cables and connectors would bring the total attenuation of the connection path outside of the USB specification. A compliant USB re-driver IC is used to boost the signal levels back to acceptable levels.

In this manual the word CONNECTION is used to imply a bidirectional link between two ports, i.e host port A (the only host on this model) takes its data from port 5, and port 5 takes its data from port A. Bidirectional CONNECTIONS can be made with the single command:

MUX:CONnect [Host PORT] [Device PORT]

This command will connect the two ports specified (i.e transmitter A to receiver 5 and transmitter 5 to receiver A) and it will remove any other prior connections associated with those two ports. On the USB Multiplexer one of the two specified ports must be host port **A**.

Examples:

We may have a Host cabled to port A and a device cabled to port 1, we can connect ports A and 1 with the command:

MUX:CONnect A 1

Now we want to connect the host to a second USB device, on port 3:

MUX:CONnect A 3

When the command above is run, the device on port 1 will be disconnected, then the device on port 3 will be connected. By default, there is a delay of 5 seconds between the disconnect and re-connect, this avoids confusing the host. The delay can be set in seconds from 0 to 255 with the command:

CONF:MUX:DELAY 10

The Mux always presents one of the 8 device ports to the host port. If you require to disconnect the current connected device from the host without selecting another device, you can use the hot-swap commands to power down the attached device. This will completely disconnect it from the host.

POWER DOWN

To reconnect the device, simple run “POWER UP” or run another “MUX:CON” command.

Basic Concepts – Cable Pull Additions

The cable-pull functionality sits between the host side of the mux and the host connector on the front of the module.

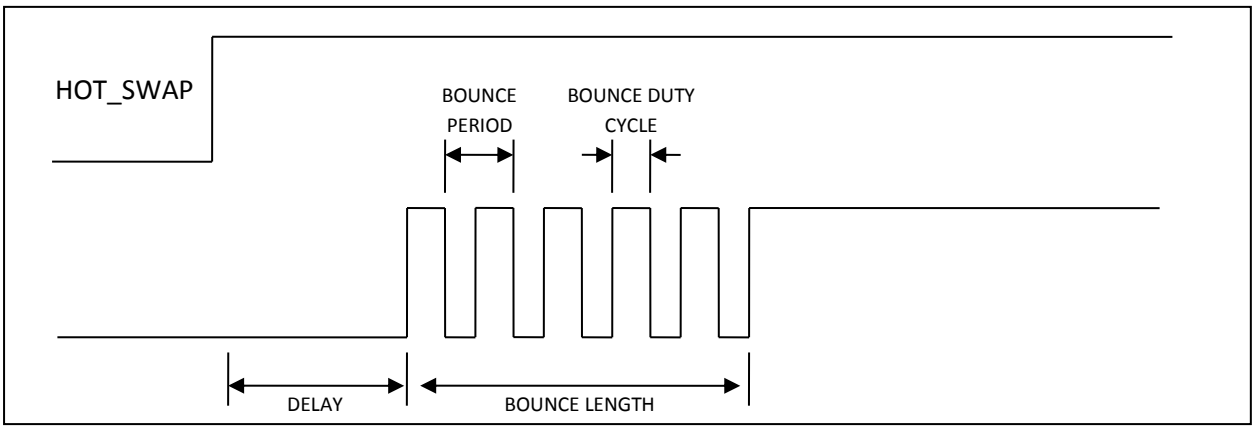
This allows us to perform fully sequenced hot-swap events, pin bounce and data glitching.

There is no need to understand this section if you simply want to use the unit to swap between USB devices. It will be useful though, if you want to do more comprehensive testing of hot-swap scenarios.

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.

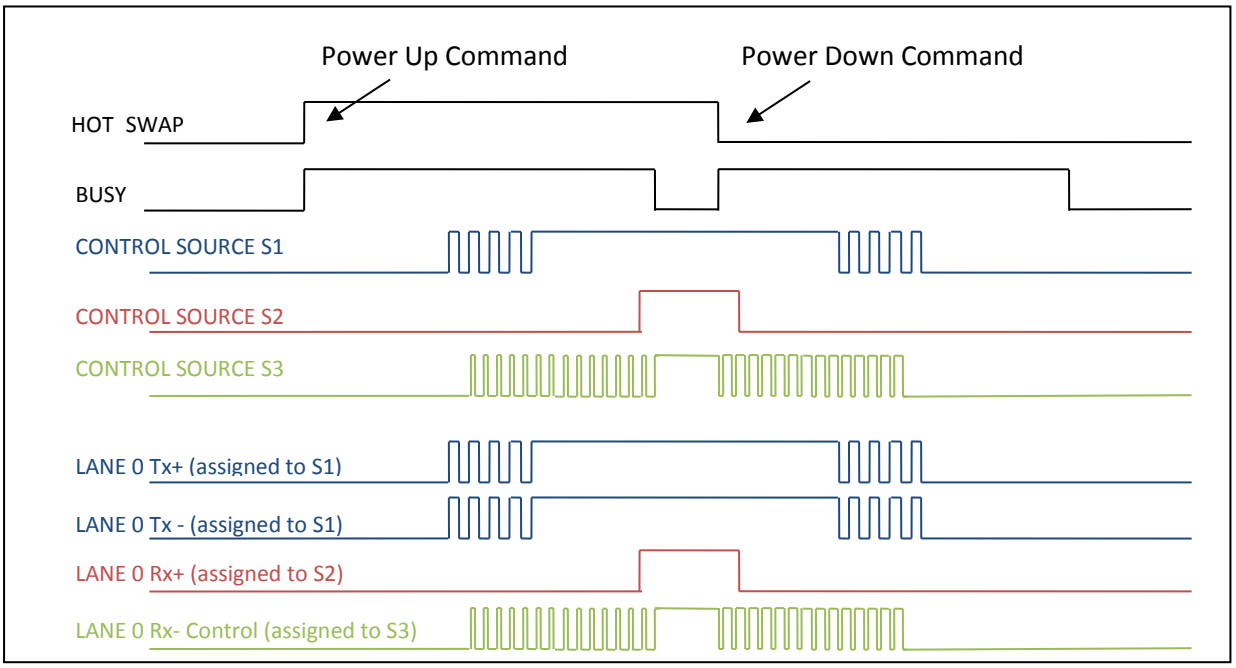
Control Source Parameters for a power up event (Basic Pin Bounce):



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:

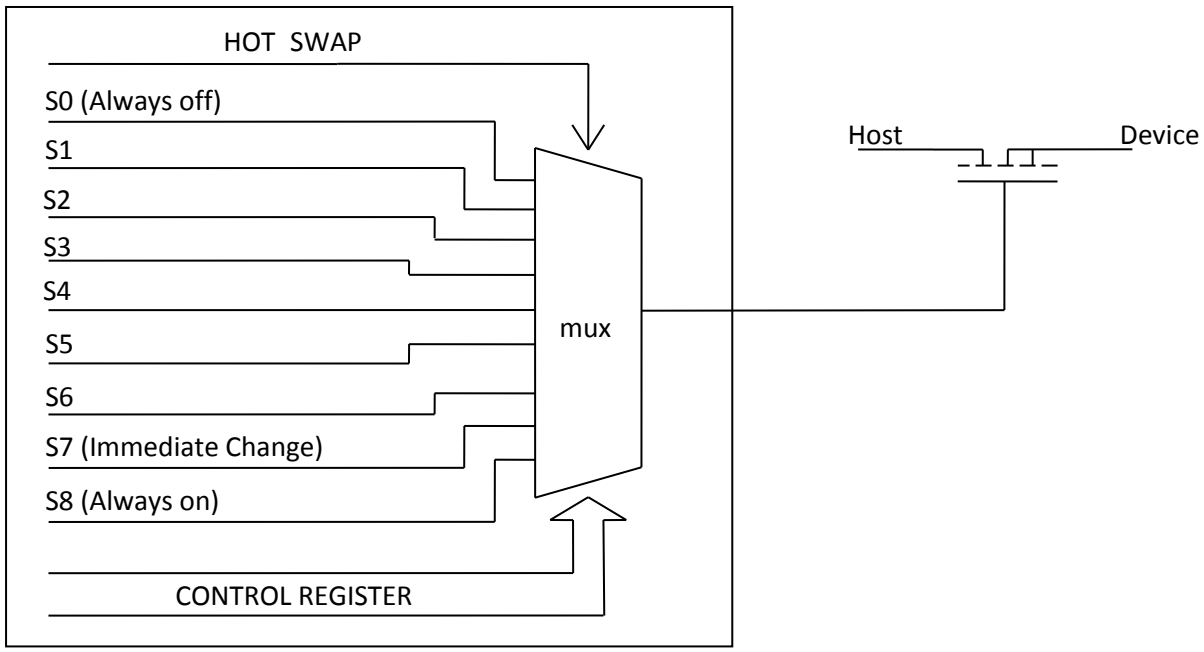


Signal Configuration

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:

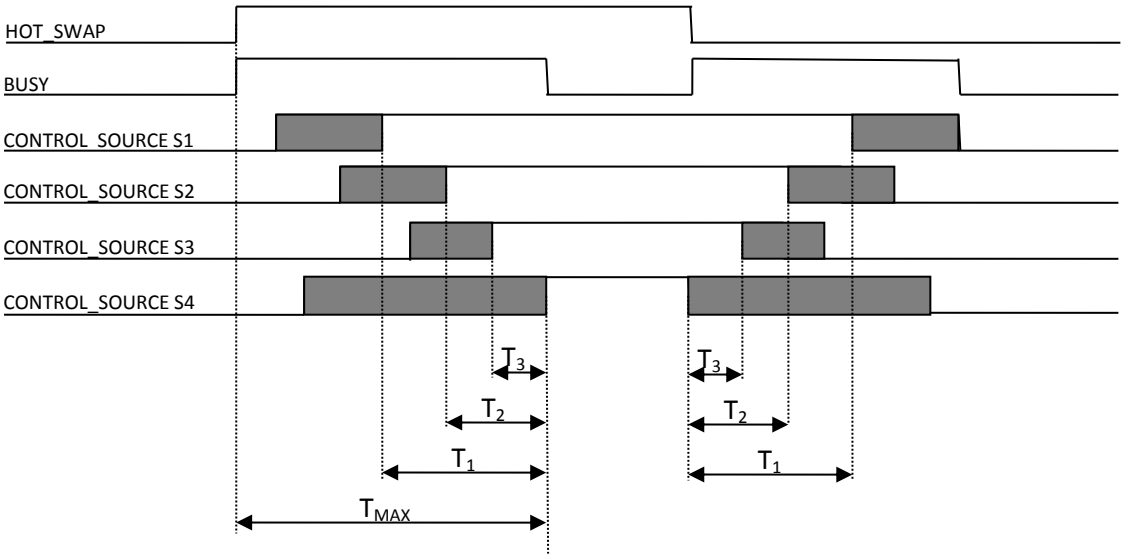
CONTROL_REGISTER Value	Description
0	Signal is always OFF
1	Signal assigned to control source 1
2	Signal assigned to control source 2
3	Signal assigned to control source 3
4	Signal assigned to control source 4
5	Signal assigned to control source 5
6	Signal assigned to control source 6
7	Signal changes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

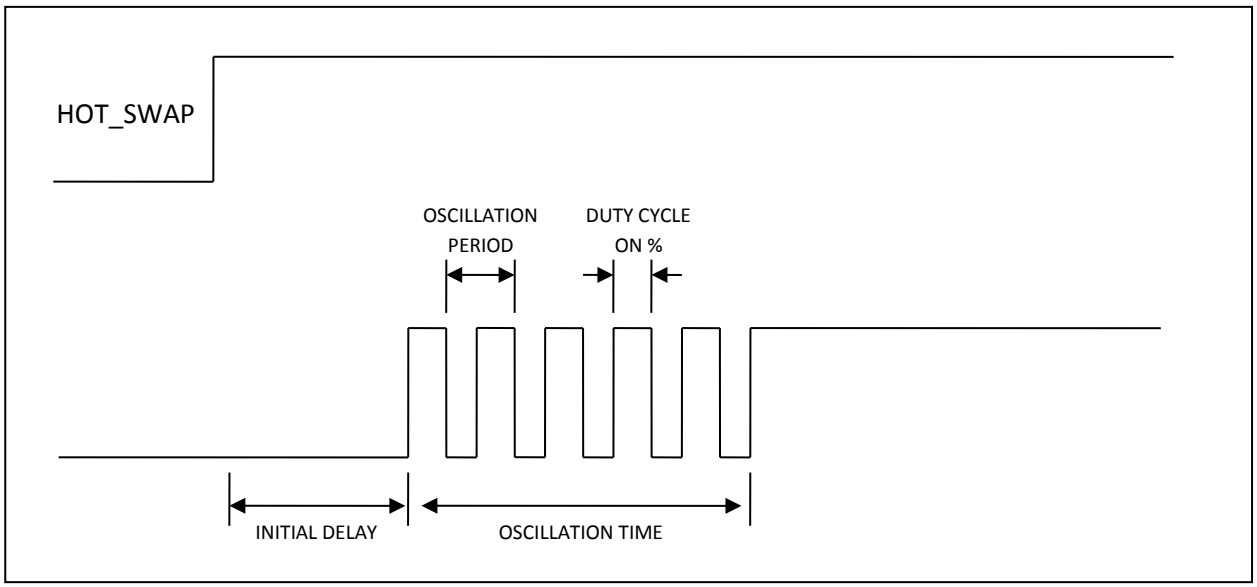
Power Down Delay T_X = T_{MAX} – (S_X_DELAY + S_X_BOUNCE_LENGTH)

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.

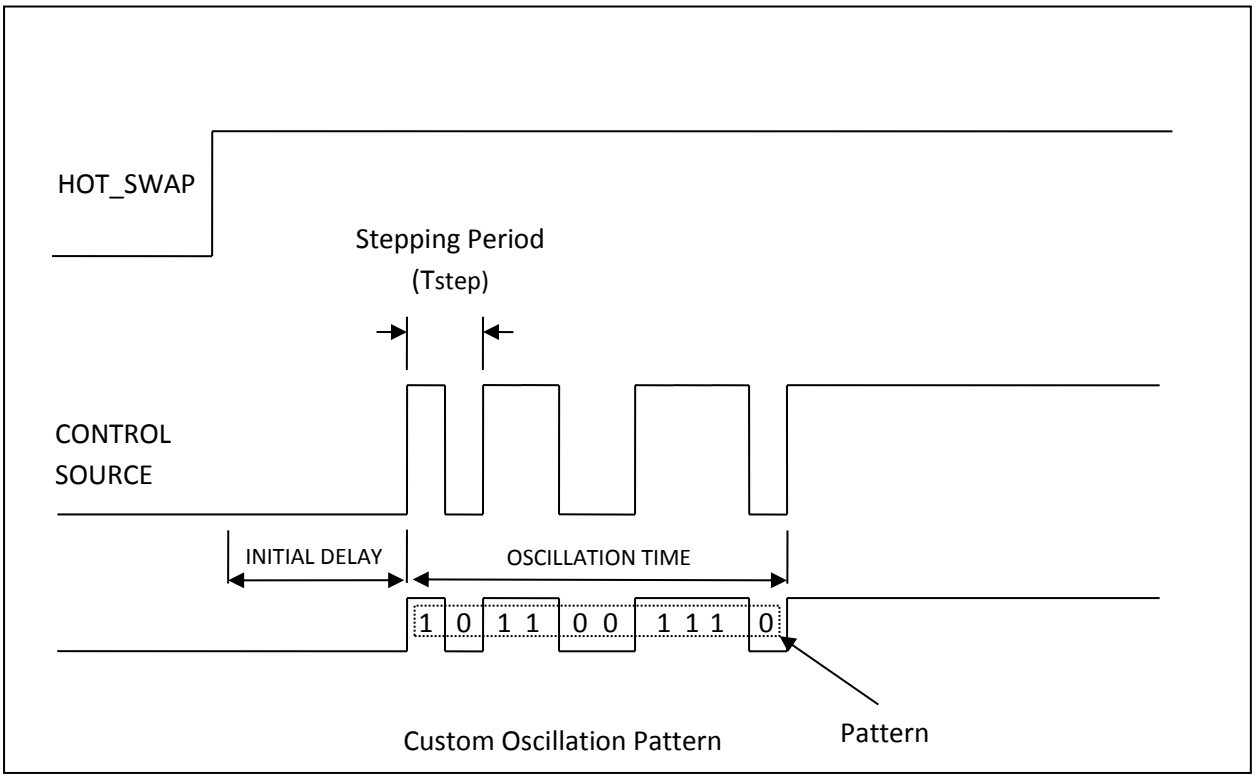
Pin Bounce Modes

Pin Bounce can be set in two ways:

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

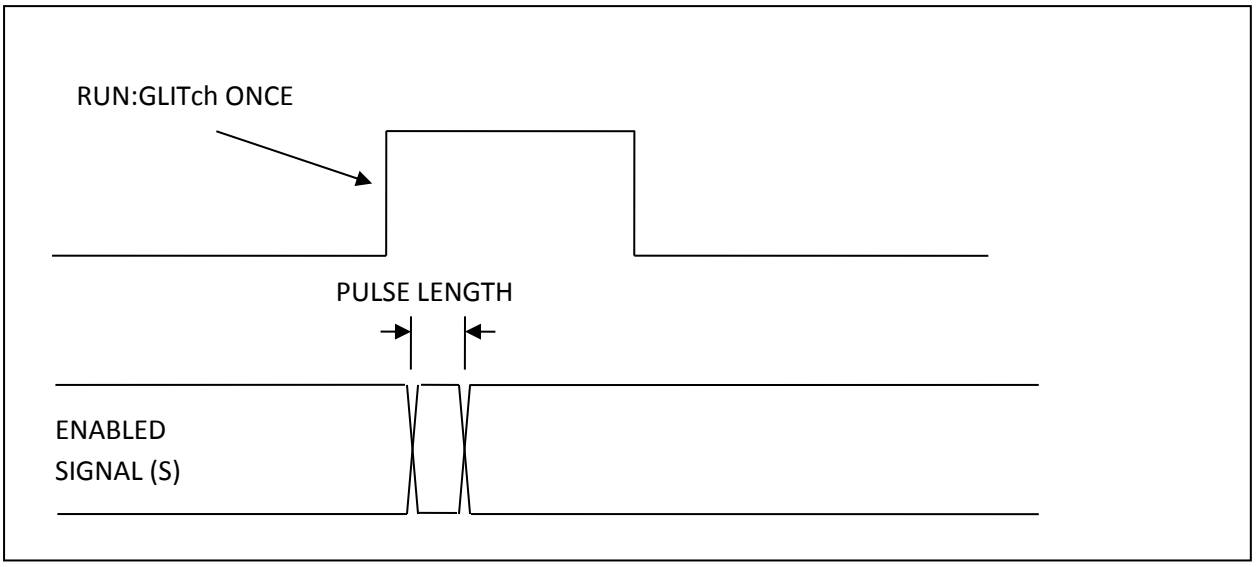
Any control signal may be glitched for a pre-determined length of time using the glitch generator logic.

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:



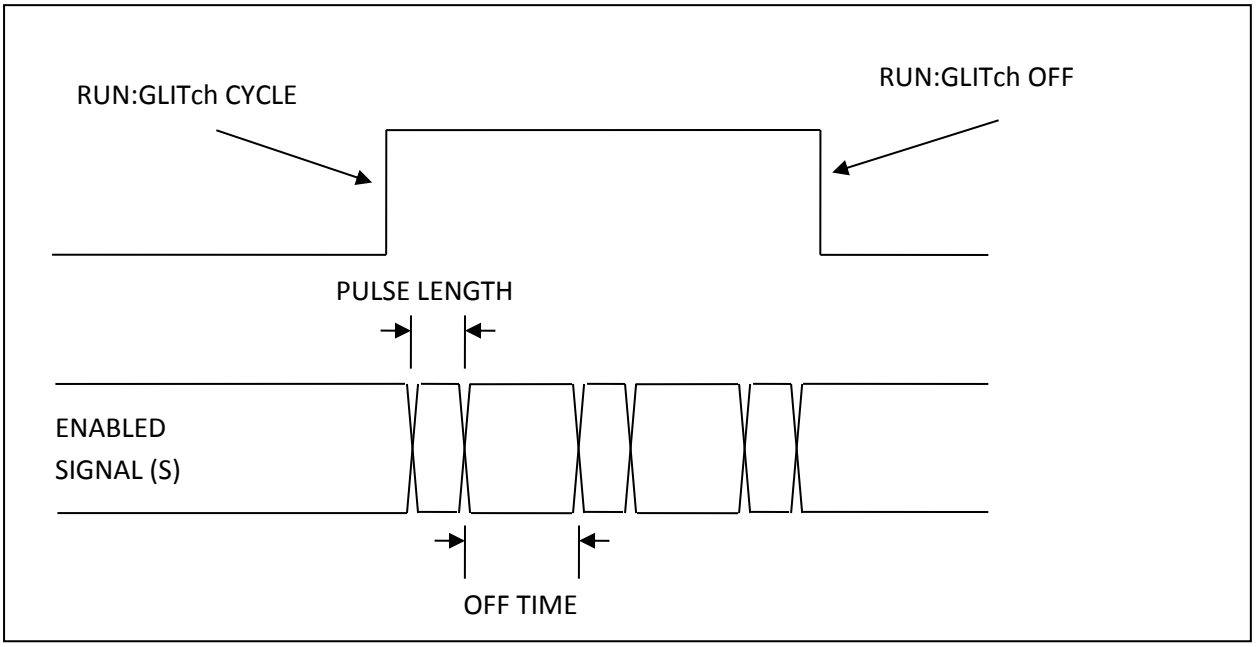
A single glitch is generated when the **RUN:GLITch ONCE** command is executed

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.

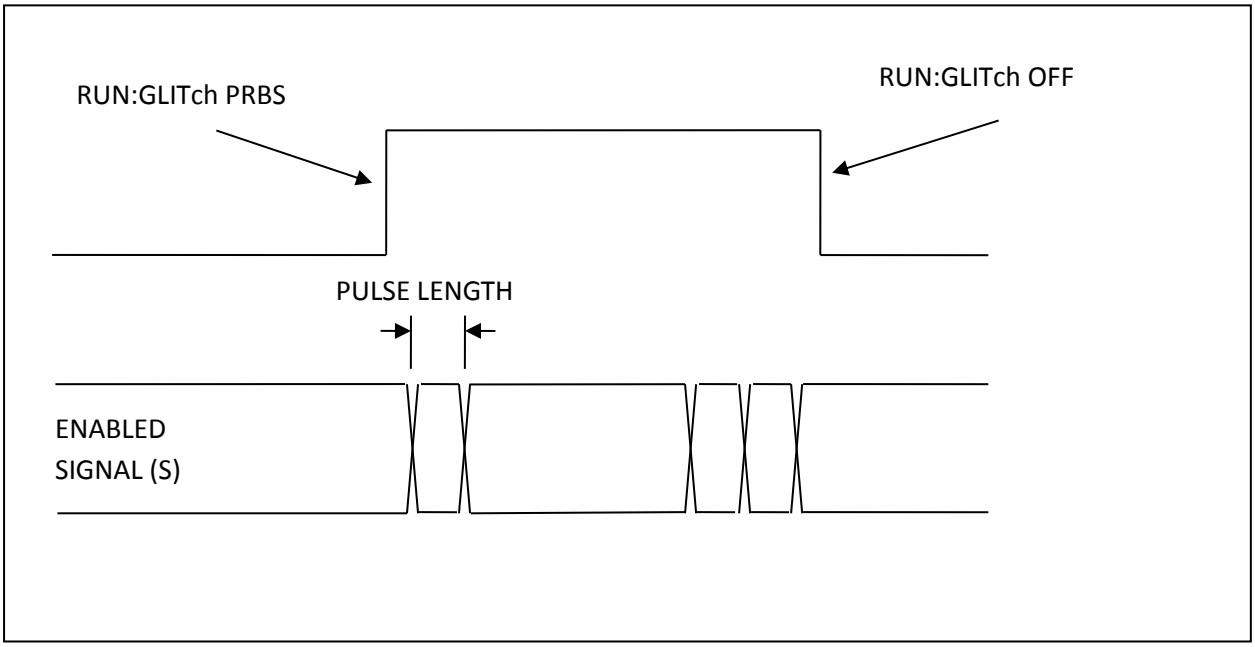
The length of the glitch is determined by using the **GLITch:SETup** command or the **GLITch:MULTiplier** and **GLITch:LENgth** commands:

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.

Front Panel LEDs

The unit uses LEDs to give a visual indication of the connections in place. Each device port has a RGB LED that changes color depending on the connection state

A solid LED means that there is a bidirectional link in place with the host port. A flashing LED means that this port is forwarding data from the host port, but not transmitting data back to it.

Blue – All signals connected – USB 2.0 and USB 3.0 available

Green – USB 2.0 connected, but USB 3.0 is disconnected

Yellow – USB 3.0 connected, but USB 2.0 is disconnected

OFF – Port is not connected, LED will also be off at any time that the VBUS signal is disconnected

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
Native Connections	-	-	Terminal Scripting TestMonkey 2 GUI	Ethernet Ethernet or USB
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

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%
MEASure:VOLTage:SELF 12v?	Returns the voltage of the modules internal 12v power rail	64mV / 5%

Default Startup State

On power up or reset, the control modules enter a default state. Each host port is connected to a device port, and hosts and devices may be plugged into these ports and will be instantly connected

Host Port	Default Device Port
A	1

The 'Cable-Pull' functions of the Mux module also start up in a default state. The defaults are designed to simulate a 'normal' hot-plug scenario that should always work correctly.

The default hot-swap scenario will disconnect the pins in sequence, based on the length of the pin in the USB connector.

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	50mS	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
VBUS	Source 1
USB 2.0 signals	Source 2
USB 3.0 signals	Source 3

Hot-Swap State:
The cable is in the 'plugged' state, so the host port will be connected to whichever of the Mux device ports is selected.

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.

CONF:MUX:DELAY [#1]

Sets the delay between disconnect and re-connect (in seconds) when using the 'MUX:CONnect' command.

MUX:CONnect [port 1] [port 2]

Creates a bi-directional link between the two specified ports, any other ports that were connected to either port are turned off

MUX:[PORT]:SOURce?

Returns the port number that [PORT] is receiving data from

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).

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

Control Register Map

Please contact Quarch if you require access to the register map. This is unlikely to be required for normal testing operations.

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

VBUS
USB2
USB3

Signal Groups

ALL (Affects all signals)