

Quarch Technology Ltd

Torridon HS Drive Control Module

Technical Manual

For use with:

QTL1177 - Torridon HS Drive Control Module

Using Quarch firmware version 4.006 and above

Monday, 24 June 2013



Change History

1.0	15 November 2010	Initial Release
1.1	17 November 2010	Added source enable commands (Firmware v3.55 and above only)
1.2	25 November 2010	Updated to new format
1.3	05 January 2011	Added firmware version 4.0 changes New Pin-bounce and default commands
1.4	24 June 2013	Updated Register map and cli commands for FPGA 1.7 with enhanced glitching capabilities

Contents

Introduction	4
Technical Specifications	5
Switching Characteristics:	5
High Speed Switch Characteristics	6
Mechanical Characteristics:	8
Control Interfaces	9
Drive Presence Support	10
GND pin P4/P6 detection	10
P1/P2 continuity detection	10
Basic Concepts	11
Signal Configuration	13
Power Up vs. Power Down Timing	14
Pin Bounce Modes	15
Glitch Control	17
Voltage Measurements	20
Default Startup State	21
Controlling the Module	22
Serial Command Set	22
SCPI Style Commands	22
Control Register Map	31
Register Definitions	33
Control Registers	33
Source Registers	35
Signal Registers	39
Other Registers	40
Appendix 1 - Signal Names	41

Introduction

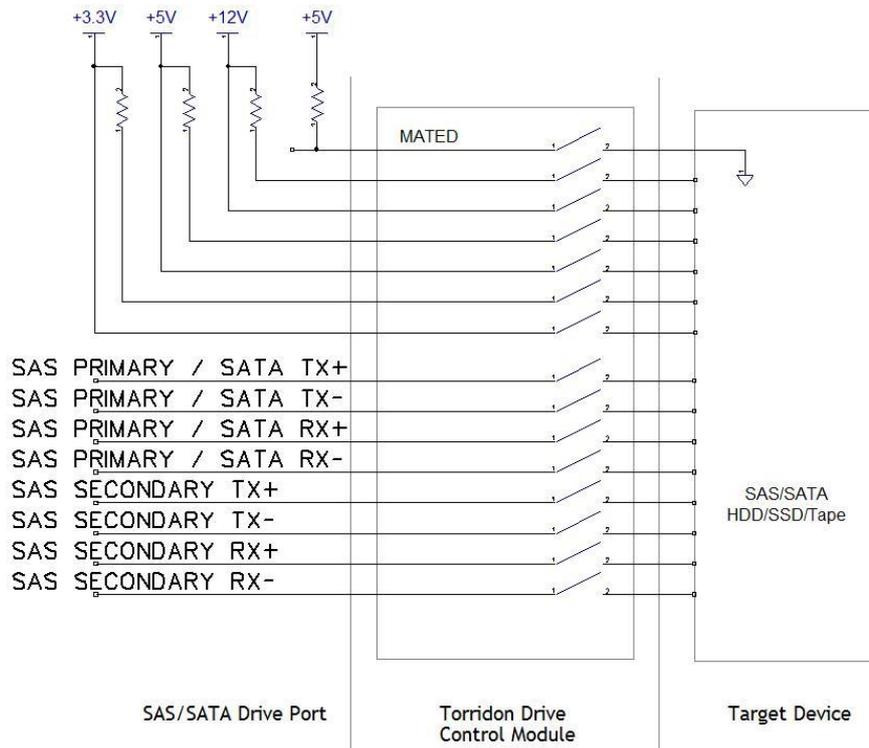
The **Torridon HS Drive Control Module** allows remote switching of power, drive presence and SAS data pins to a SAS or SATA Disk Drive whilst plugged into a host system.

Any SAS or SATA drive can be controlled, including: HDD, SSD, CD, Tape etc

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 HS control module switches the high speed SAS data lines. 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

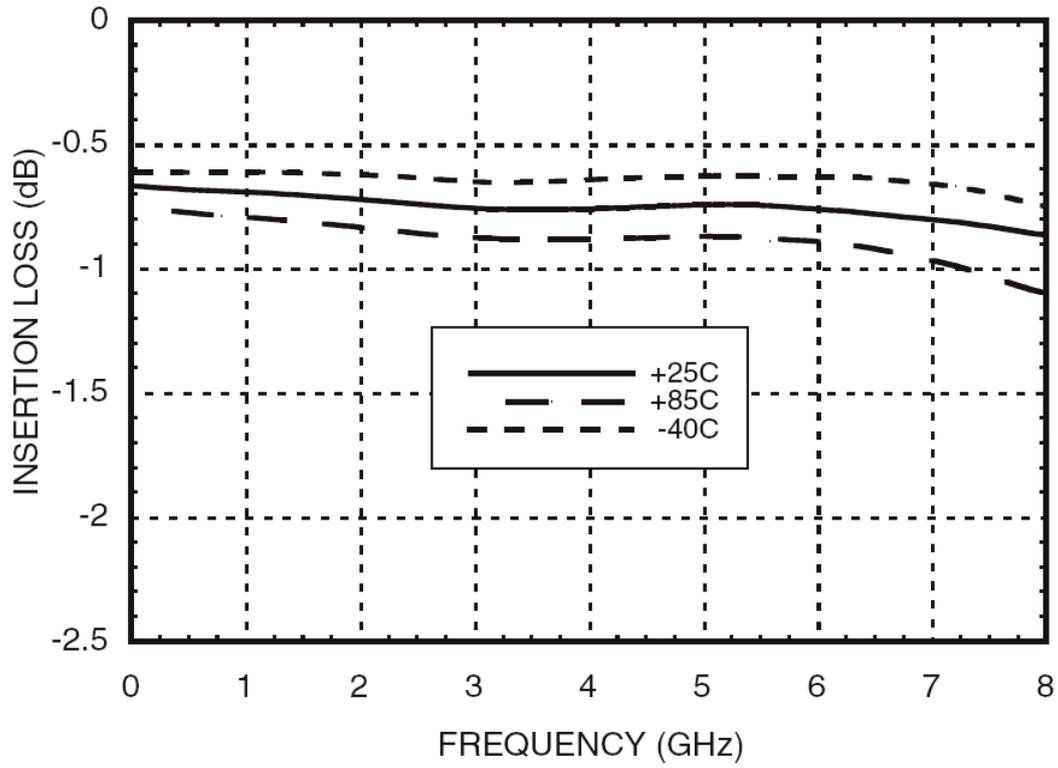
Switching Characteristics:

SAS Connector Pin	Description	Switching Action
S1,S4,S7,S8,S11,S14,P5, P10,P12	SAS Data and Power Ground Pins	All connected to digital Ground on the Module
S2,S3,S5,S6,S9,S10,S12,S13	SAS Data Signal pins	Each signal is individually switched by a High Speed RF Switch
P1,P2	3.3V Power Pins	Connected together and switched by 16A power FET.
P3	3.3V Pre-Charge Power Pin	Switched by 16A power FET
P4,P6	Ground / SPECIAL_1 (vendor specific*)	Switched by 0.5A FET
P7	5V Pre-Charge Power Pin	Switched by 16A power FET
P8,P9	5V Power Pins	Connected together and switched by 16A power FET.
P11	Ready LED	Individually connected from plug to receptacle
P13	12V Pre-Charge Power Pin	Switched by 16A power FET
P14,P15	12V Power Pins	Connected together and switched by 16A power FET.

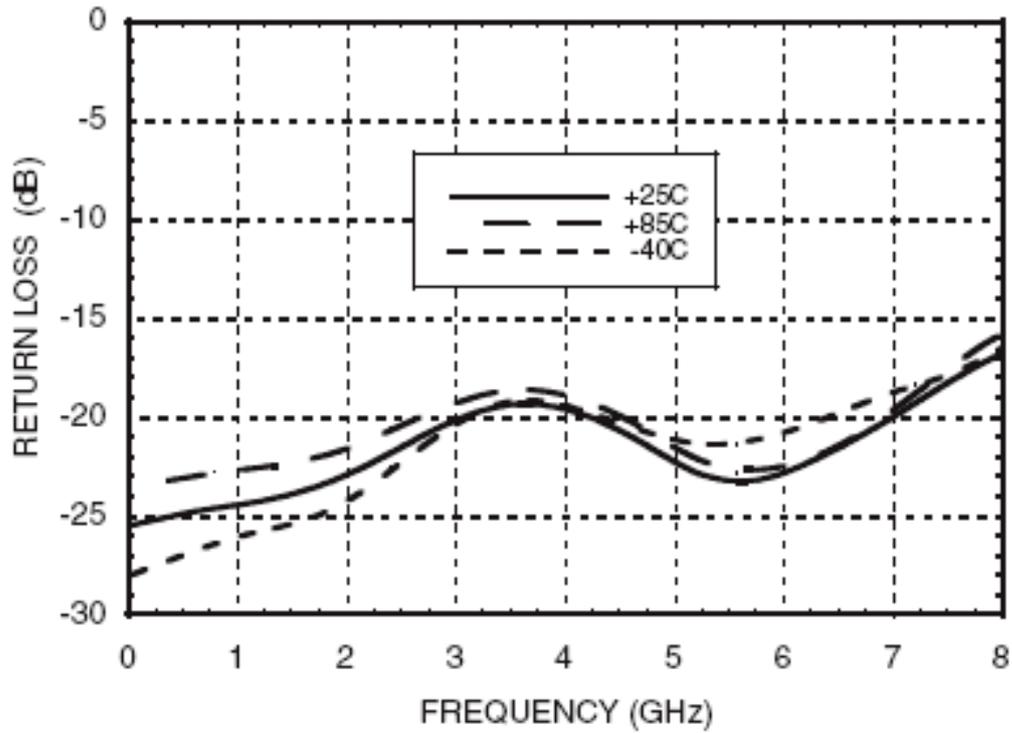
*Power Pins P4 and P6 are used as Mated signals by some vendors, these may also be independently switched to control drive presence. Subject to volume and lead time, the units may be customized to suit any proprietary mated circuitry.

High Speed Switch Characteristics

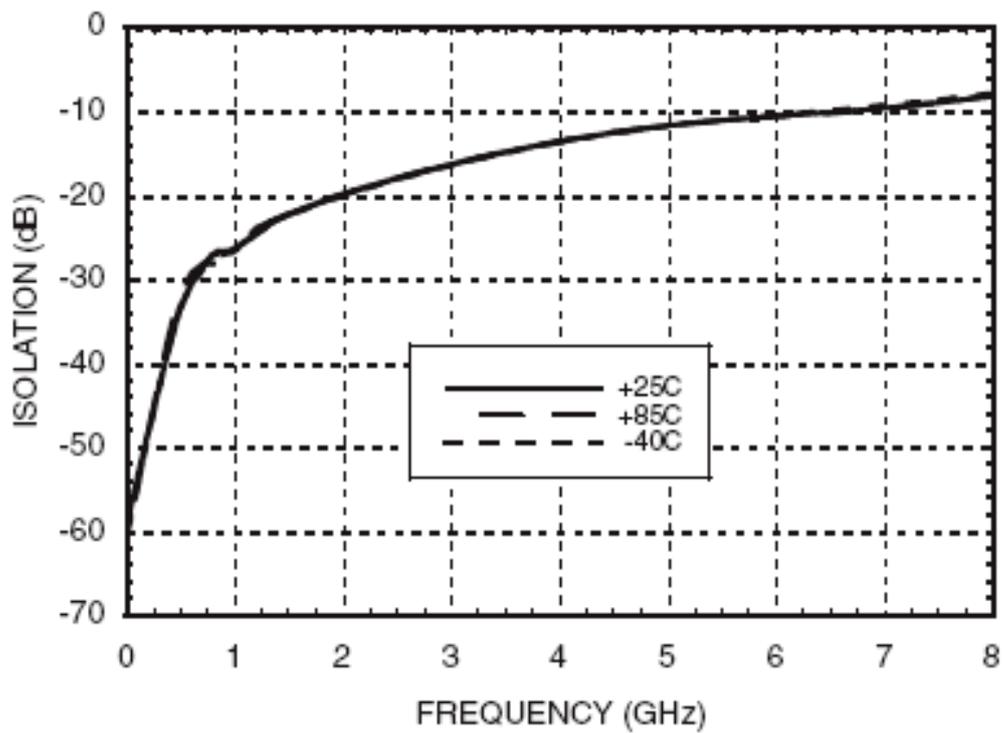
Insertion Loss



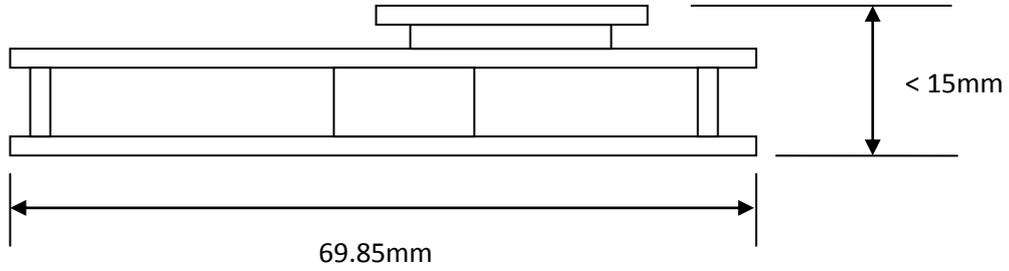
Return Loss



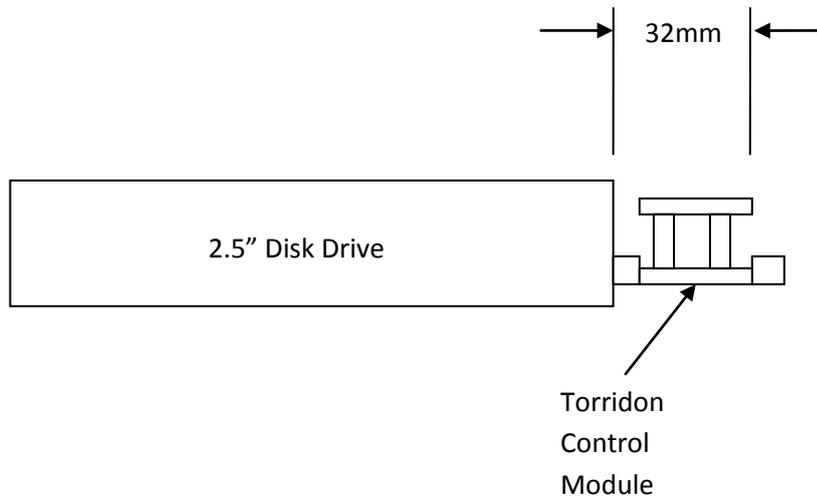
Isolation



Mechanical Characteristics:



- The Modules have the same cross section as a 2.5" disk drive, allowing them to fit into any 2.5" or 3.5" drive enclosure.



Control Interfaces

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

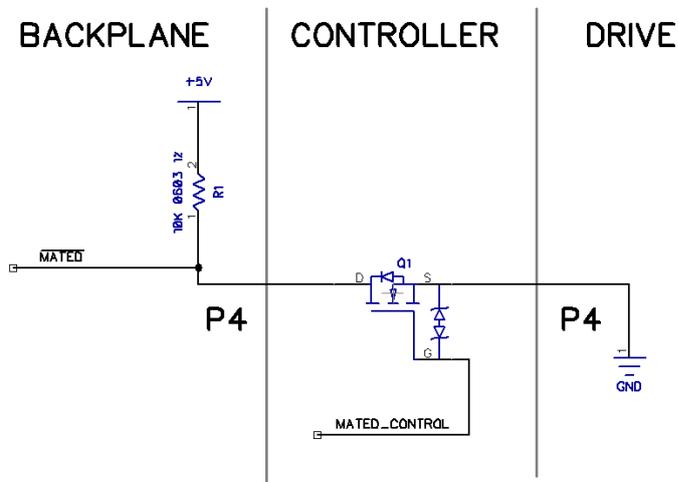
The control cable is an ultra-thin Flex cable.

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

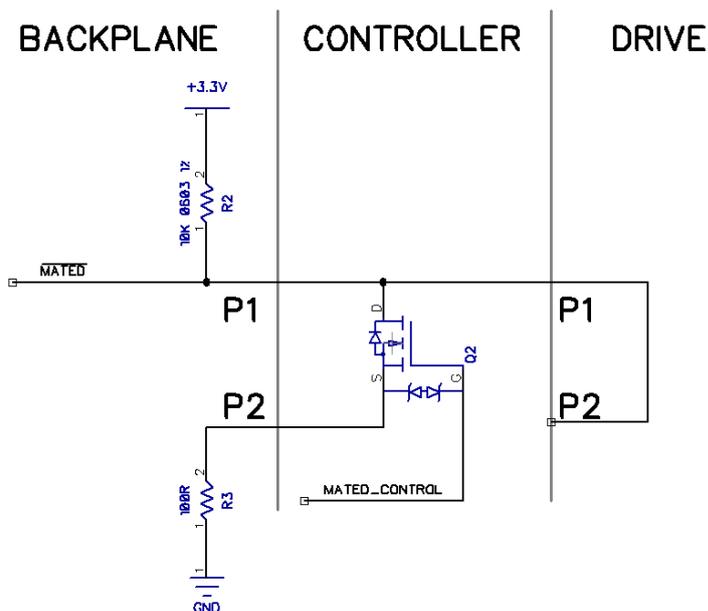
Drive Presence Support

The module supports two different proprietary drive presence circuits, all are switched by the same control signal SPECIAL_1. If the detection circuits are not implemented on the backplane this circuitry should have no adverse effect on drive operation.

GND pin P4/P6 detection



P1/P2 continuity detection

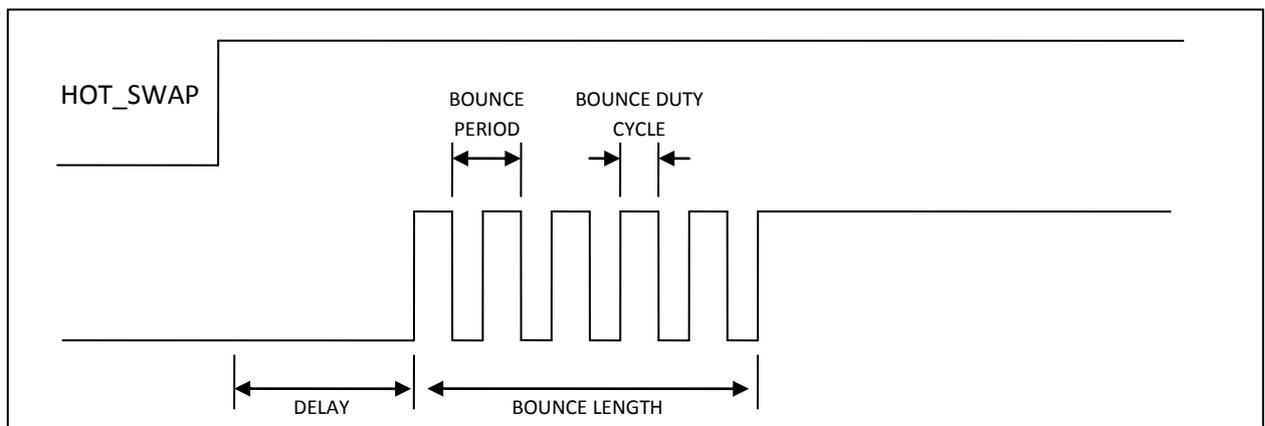


Basic Concepts

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.

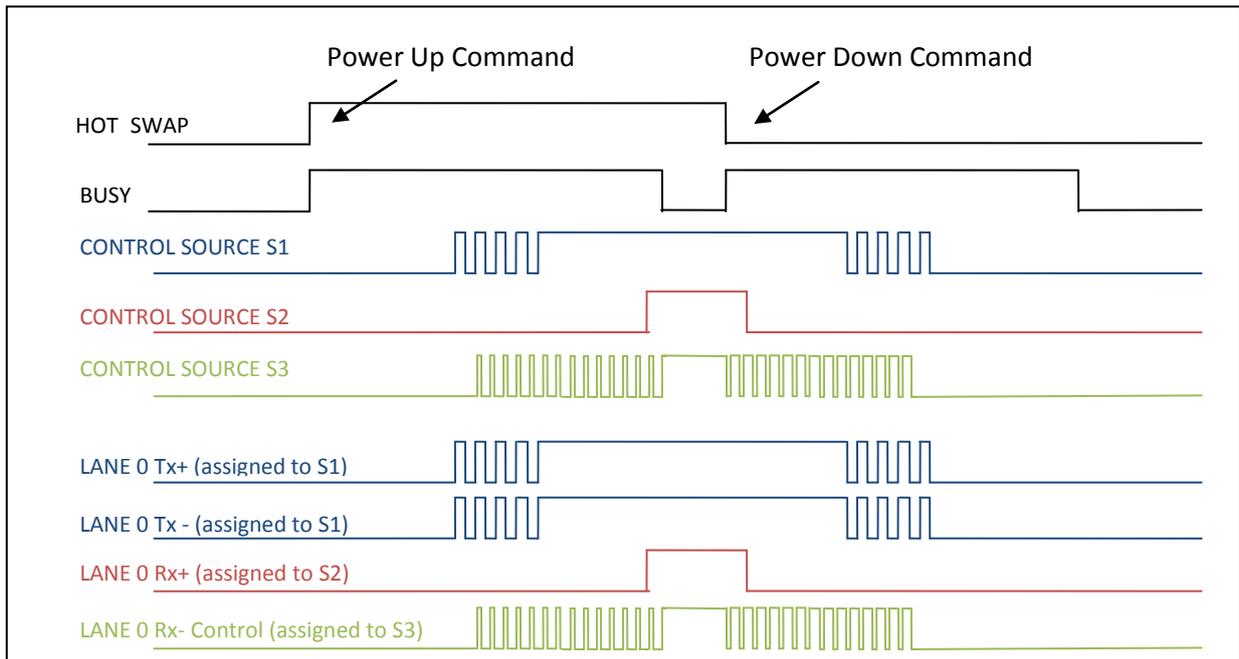
Control Source Parameters for a power up event (Basic Pin Bounce):



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:

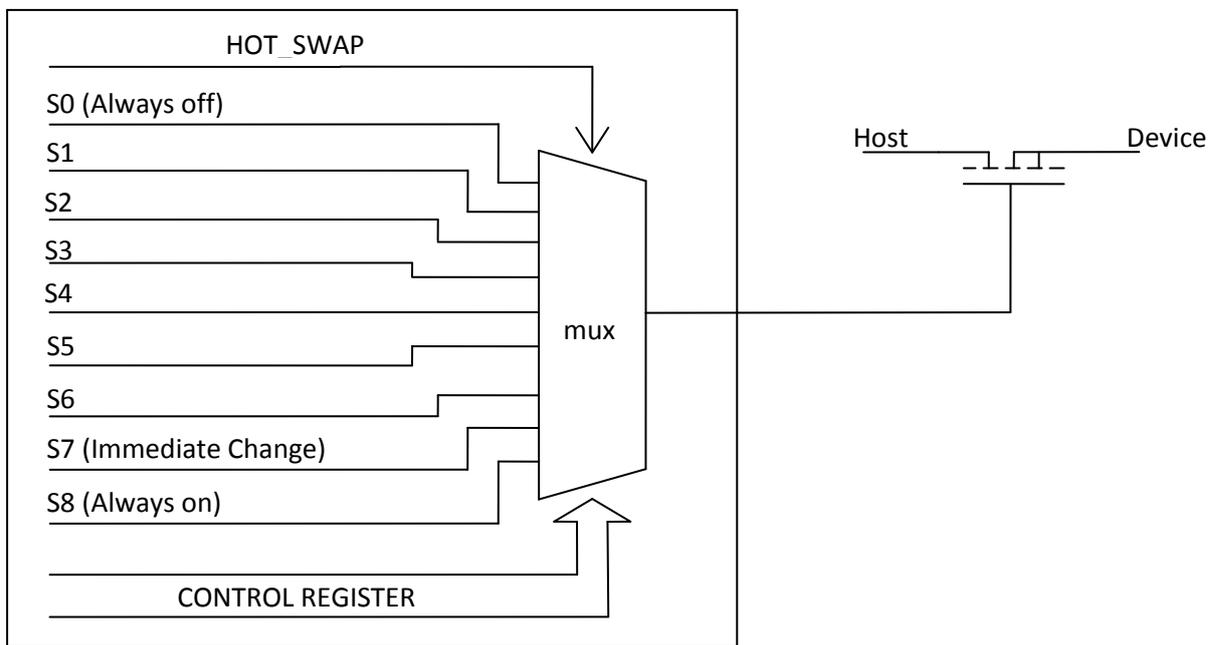


Signal Configuration

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:

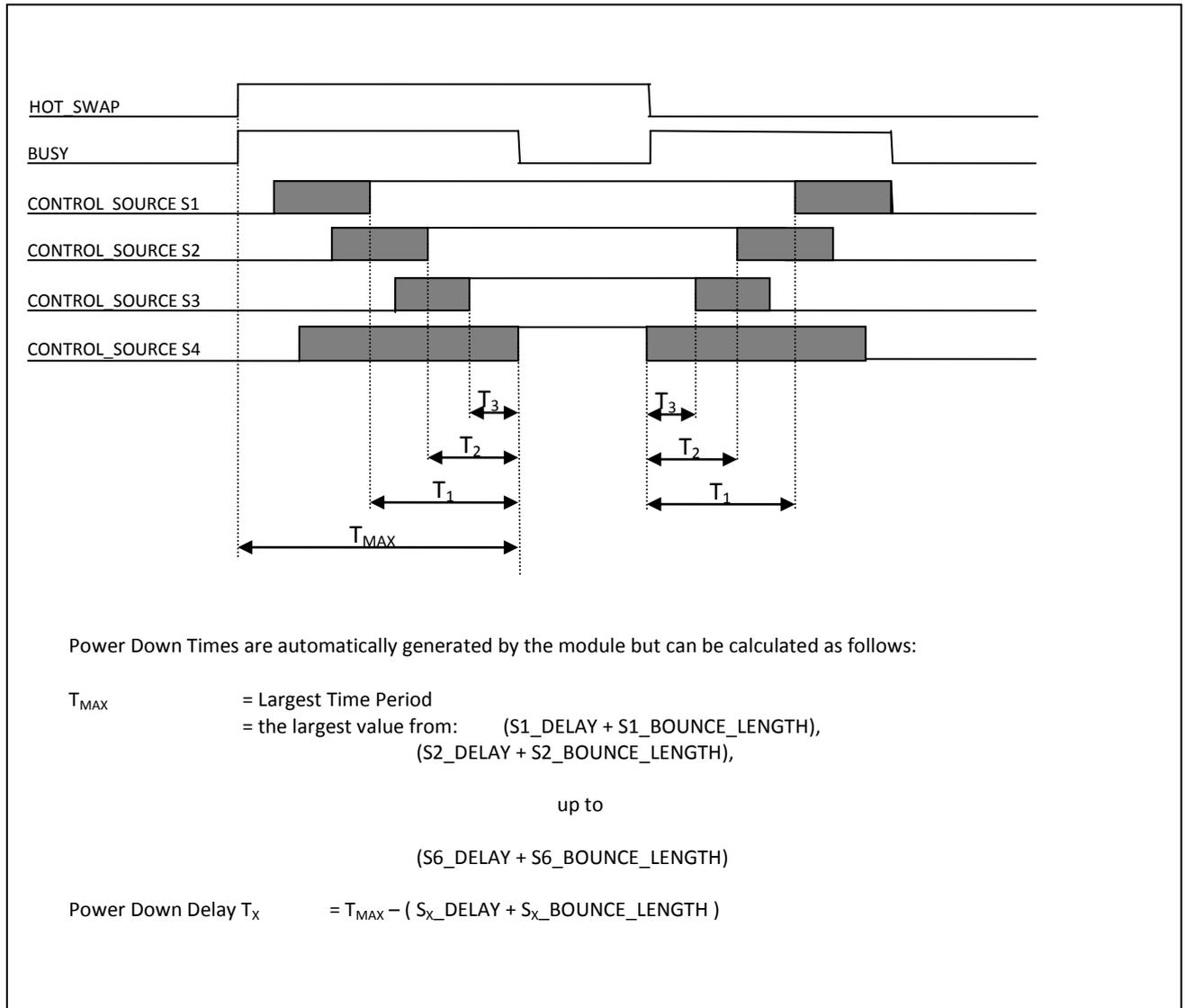
CONTROL_REGISTER Value	Description
0	Signal is always OFF
1	Signal assigned to control source 1
2	Signal assigned to control source 2
3	Signal assigned to control source 3
4	Signal assigned to control source 4
5	Signal assigned to control source 5
6	Signal assigned to control source 6
7	Signal changes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



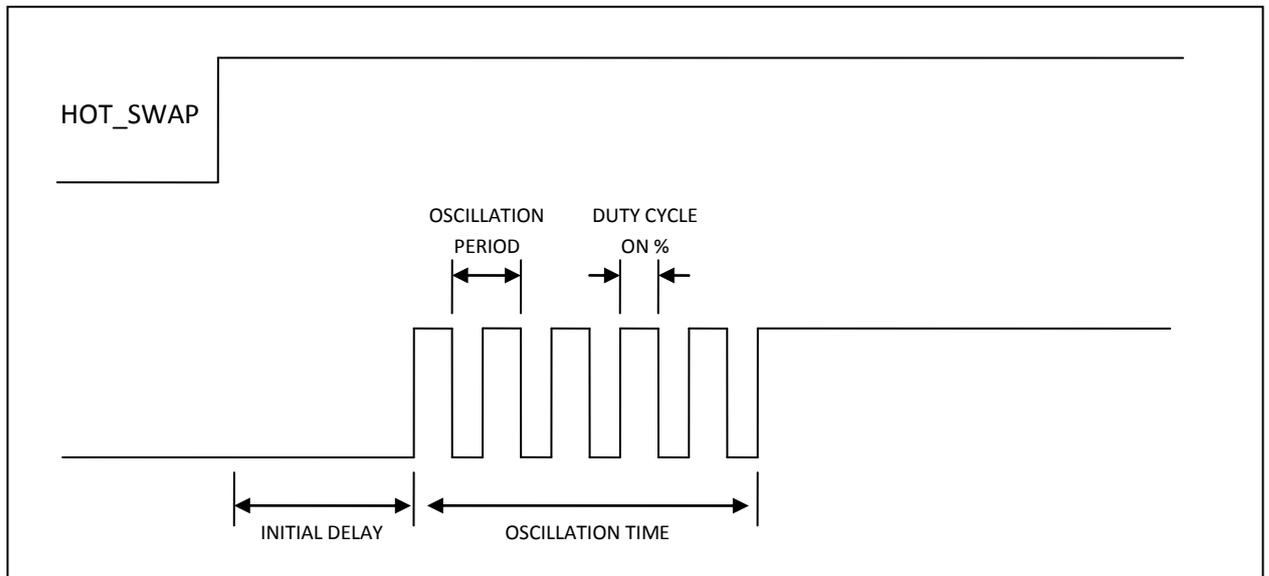
If you require a different power down sequence then you can alter any of the source timing values, pin bounce or signal assignments while the module is in the plugged state. When you initiate the 'pull' action, the new settings will be used.

Pin Bounce Modes

Pin Bounce can be set in two ways:

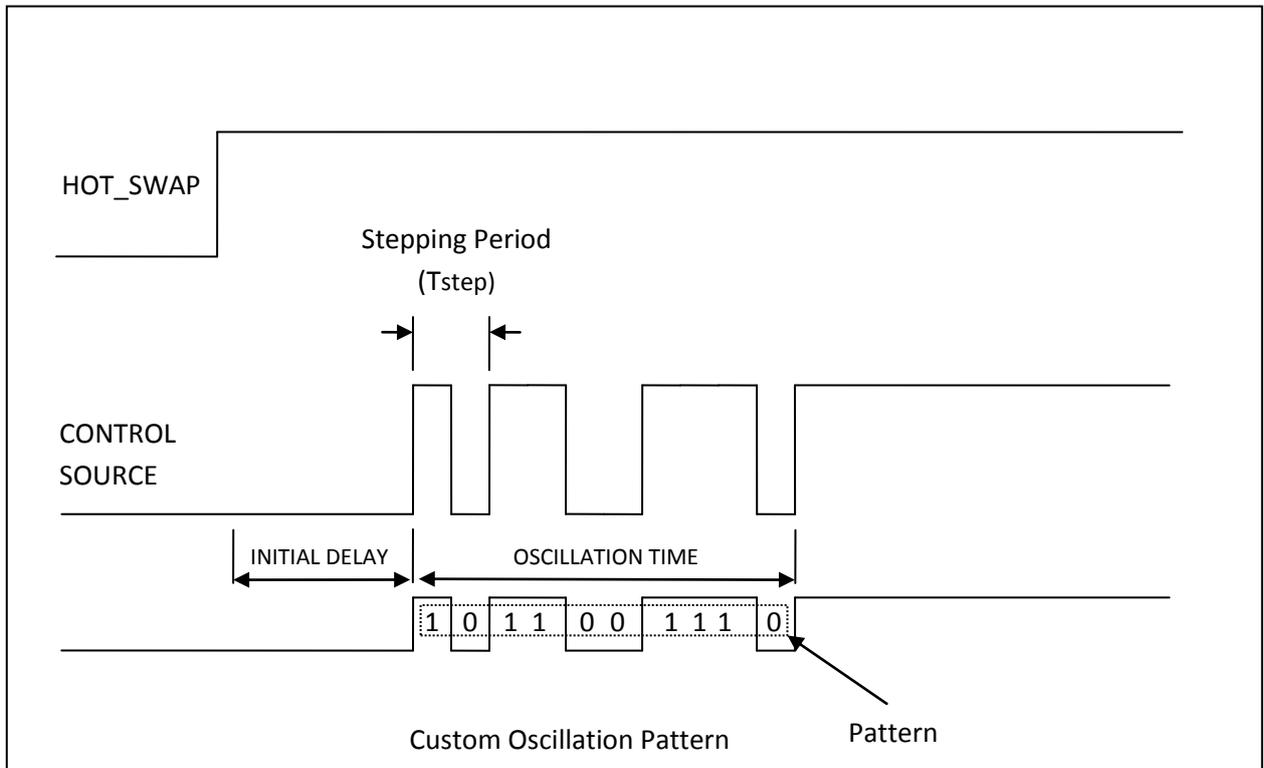
1. Basic Pin-Bounce (Constant Oscillation Frequency):

- 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

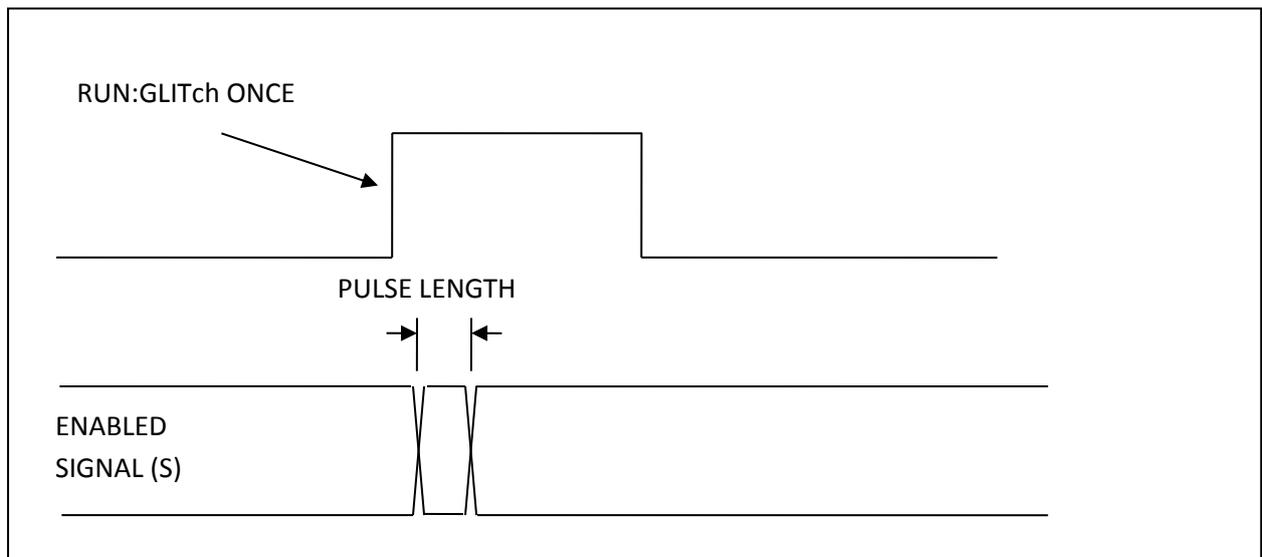
Any control signal may be glitched for a pre-determined length of time using the glitch generator logic.

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:



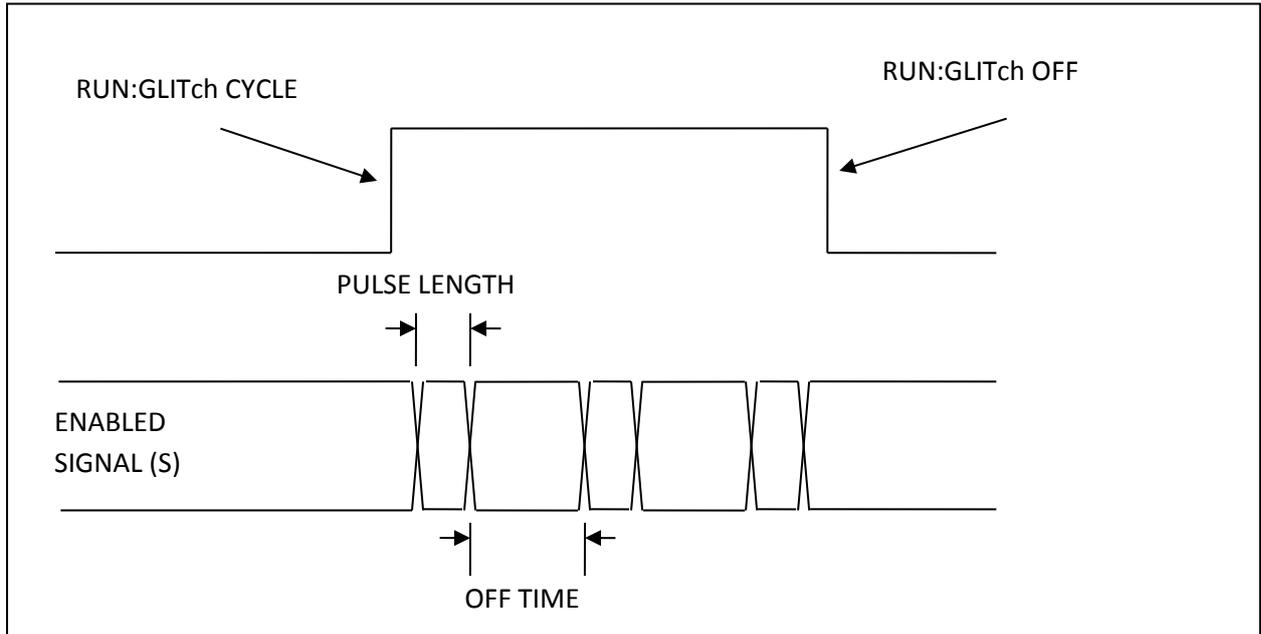
A single glitch is generated when the **RUN:GLITCh ONCE** command is executed

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.

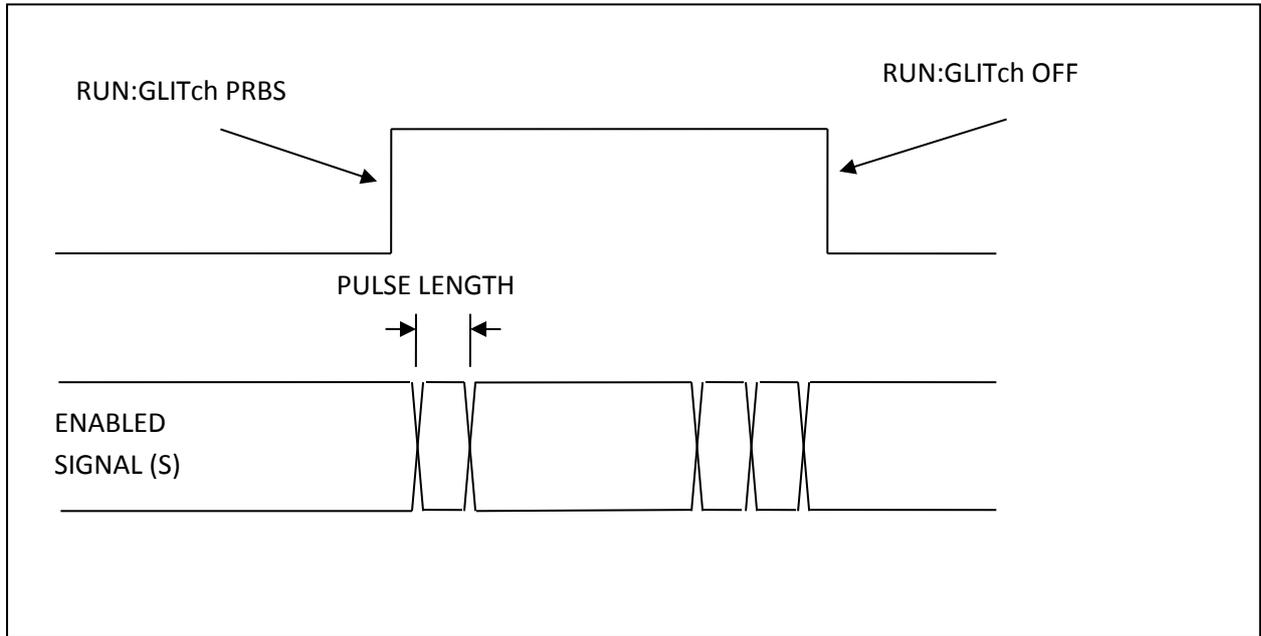
The length of the glitch is determined by using the **GLITCh:SETup** command or the **GLITCh:MULTiplier** and **GLITCh:LENGth** commands:

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

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 1v2?	Returns the voltage of the modules internal 1.2v rail	6mV / 1.5%
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 3v3in?	Returns the voltage of the 3.3V power pins on the backplane (unswitched) side of the Module	12mV / 3%
MEASure:VOLTage 3v3out?	Returns the voltage of the 3.3V power pins on the drive (switched) side of the module	12mV / 3%
MEASure:VOLTage 5vin?	Returns the voltage of the 5V power pins on the backplane (unswitched) side of the Module	15mV / 3%
MEASure:VOLTage 5vout?	Returns the voltage of the 5v power pins on the drive (switched) side of the module	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%

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
SPECIAL1	Source 1
3V3_CHARGE, 5V_CHARGE, 12V_CHARGE	Source 2
3V3_POWER, 5V_POWER, 12V_POWER	Source 3
PRI_OUT_PL, PRI_OUT_MN, PRI_IN_PL, PRI_IN_MN, SEC_OUT_PL, SEC_OUT_MN, SEC_IN_PL, SEC_IN_MN, SEC_OUT_PL, SEC_OUT_MN	Source 3

Hot-Swap State:

Drive is in the ‘pulled’ state, waiting for a “**RUN:POWER UP**” command to attach 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 ':' (LEVe11:LEVe12:LEVe13)
- 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 [3v3in? | 3v3out? | 5vin? | 5vout? | 12vin? | 12vout?]

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 [1v2?,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 instead of writing to the registers.

Address	Name	Description
0x00	Global Control	Trigger Power up/down and Glitch
0x01	Glitch Pulse Control	Glitch Pulse Length
0x02	Glitch Cycle Control	Glitch Off Time
0x03	Glitch PRBS Control	Glitch PRBS settings
0x04	Reserved	Reserved
0x05	S1 Initial Delay & Bounce Period	Source 1 Setup
0x06	S1 Bounce Length & Duty Cycle	Source 1 Setup
0x0D..0x07	S1 Custom Pattern	112 bit bounce pattern
0x0E	S2 Initial Delay & Bounce Period	Source 2 Setup
0x0F	S2 Bounce Length & Duty Cycle	Source 2 Setup
0x16..0x10	S2 Custom Pattern	112 bit bounce pattern
0x17	S3 Initial Delay & Bounce Period	Source 3 Setup
0x18	S3 Bounce Length & Duty Cycle	Source 3 Setup
0x1F..0x19	S3 Custom Pattern	112 bit bounce pattern
0x20	S4 Initial Delay & Bounce Period	Source 4 Setup
0x21	S4 Bounce Length & Duty Cycle	Source 4 Setup
0x28..0x22	S4 Custom Pattern	112 bit bounce pattern
0x29	S5 Delay & Bounce Period	Source 5 Setup
0x2A	S5 Bounce length & Duty Cycle	Source 5 Setup
0x31..0x2B	S5 Custom Pattern	112 bit bounce pattern
0x32	S6 Delay & Bounce Period	Source 6 Setup
0x33	S6 Bounce Length & Duty Cycle	Source 6 Setup
0x3A..0x34	S6 Custom Pattern	112 bit bounce pattern
0x6D	3V3_POWER	Signal Assignment Register
0x6E	3V3_CHARGE	Signal Assignment Register
0x6F	5V_POWER	Signal Assignment Register
0x70	5V_CHARGE	Signal Assignment Register
0x71	12V_POWER	Signal Assignment Register
0x72	12V_CHARGE	Signal Assignment Register
0x73	SPECIAL1 (MATED)	Signal Assignment Register
0x74	PRI_OUT_PL (Backplane primary out +ve)	Signal Assignment Register
0x75	PRI_OUT_MN (Backplane primary out -ve)	Signal Assignment Register
0x76	PRI_IN_PL (Backplane primary in +ve)	Signal Assignment Register
0x77	PRI_IN_MN (Backplane primary in -ve)	Signal Assignment Register

0x78	SEC_OUT_PL (Backplane secondary out +ve)	Signal Assignment Register
0x79	SEC_OUT_MN (Backplane secondary out -ve)	Signal Assignment Register
0x7a	SEC_IN_PL (Backplane secondary in +ve)	Signal Assignment Register
0x7b	SEC_IN_MN (Backplane secondary in -ve)	Signal Assignment Register
0xfffe	CPLD Part Number	4 character CPLD Part Number
0xffff	CPLD Part version	CPLD Part version

Register Definitions

Control Registers

0x00 - Global Control

15	14	13	12	11	10	9	8
RESERVED					GLITCH_PRBS	GLITCH_CYCLE	GLITCH_TRIGGER

7	6	5	4	3	2	1	0
SOURCE_6_ENABLE	SOURCE_5_ENABLE	SOURCE_4_ENABLE	SOURCE_3_ENABLE	SOURCE_2_ENABLE	SOURCE_1_ENABLE	BUSY	HOT_SWAP

Name	Description
GLITCH_PRBS	Start PRBS glitch mode. When set, this bit overrides GLITCH_CYCLE
GLITCH_CYCLE	Selects Glitch cycle mode when set, glitch once mode when clear
GLITCH_TRIGGER	Set this signal to start a single glitch, or a glitch cycle, clear this signal to stop a glitch cycle
SOURCE_x_ENABLE	Clearing this bit instantly disables all signals assigned to source x.
BUSY	This bit is cleared when the card is in a steady state, i.e. all delays/pin bounce/glitches are completed
HOT_SWAP	Setting this bit initiates a hot plug sequence, and clearing it initiates a hot pull

0x01 - Glitch Pulse Control

15	14	13	12	11	10	9	8
RESERVED					PULSE_MULTIPLIER		
7	6	5	4	3	2	1	0
PULSE_LENGTH							

Name	Description
PULSE_MULTIPLIER	Change step size, 0 = 50nS, 1 = 500ns, 2= 5us,3=50us,4=500us,5=5ms,6=50ms,7=500ms
PULSE_LENGTH	Set the number of steps to glitch for (1-255)

0x02 - Glitch Cycle Control

15	14	13	12	11	10	9	8
RESERVED					CYCLE_MULTIPLIER		
7	6	5	4	3	2	1	0
CYCLE_LENGTH							

Name	Description
CYCLE_MULTIPLIER	Change step size, 0 = 50nS, 1 = 500ns, 2= 5us,3=50us,4=500us,5=5ms,6=50ms,7=500ms
CYCLE_LENGTH	Number of steps to stay off for (1 – 255)

0x03 – Glitch PRBS Control

15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED				GLITCH_PRBS_DUTY			

Name	Description
GLITCH_PRBS_DUTY	Set the “on percentage” of the PRBS signal 0 = 1 in 65536 pulses is glitched

- 1 = 1 in 32768 pulses is glitched
- 2 = 1 in 16384 pulses is glitched
- 3 = 1 in 8192 pulses is glitched
- 4 = 1 in 4096 pulses is glitched
- 5 = 1 in 2048 pulses is glitched
- 6 = 1 in 1024 pulses is glitched
- 7 = 1 in 512 pulses is glitched
- 8 = 1 in 256 pulses is glitched
- 9 = 1 in 128 pulses is glitched
- 10 = 1 in 64 pulses is glitched
- 11 = 1 in 32 pulses is glitched
- 12 = 1 in 16 pulses is glitched
- 13 = 1 in 8 pulses is glitched
- 14 = 1 in 4 pulses is glitched
- 15 = 1 in 2 pulses is glitched

Source Registers

Each source is setup by a block of 9 16-bit wide registers. Below is the register map for a generic source. The list of registers in the title indicates the actual address of the byte in each of the 6 timed sources.

Source Delay & Bounce Period

[0x05, 0x0E, 0x17, 0x20, 0x29, 0x32]

15	14..8	7	6..0
Sx_BOUNCE_PERIOD_MULTIPLIER	Sx_BOUNCE_PERIOD	Sx_DELAY_MULTIPLIER	Sx_DELAY

Name	Description
Sx_DELAY_MULTIPLIER	When 0, Delay Multiplier is 1mS When 1, Delay Multiplier is 10mS
Sx_DELAY	The Total delay between HOT_SWAP being set or start of a power cycle event and the signal beginning to mate $T_{DELAY} = xV_DELAY \times xV_DELAY_MULTIPLIER$ i.e. 00000010 = 2mS, 10001001 = 90mS

Sx_BOUNCE_PERIOD_MULTIPLIER

When 0, Delay Multiplier is 10uS

When 1, Delay Multiplier is 1mS

Sx__BOUNCE_PERIOD

The Period of the bounce frequency when pin-bounce is enabled

Period = xV_BOUNCE_PERIOD x

xV_BOUNCE_PERIOD_MULTIPLIER

i.e. 00000010 = 20uS, 10001001 = 9mS

Source Bounce Mode, Bounce Length & Bounce Duty Cycle

[0x06,0x0F, 0x18, 0x21, 0x2A, 0x33]

15	14..8	7	6..0
Sx_PIN_BOUNCE_MODE	Sx_BOUNCE_DUTY_CYCLE	Sx_BOUNCE_TIME_MULTIPLIER	Sx_BOUNCE_TIME

Name	Description
Sx_BOUNCE_TIME_MULTIPLIER	When 0, Delay Multiplier is 1mS When 1, Delay Multiplier is 10mS
Sx_BOUNCE_TIME	The period of the bounce frequency when pin-bounce is enabled $T_{BOUNCE} = xV_BOUNCE_TIME \times xV_BOUNCE_TIME_MULTIPLIER$ i.e. 00000010 = 2mS, 10001001 = 90mS
Sx_BOUNCE_DUTY_CYCLE	The Duty Cycle of the bounce frequency, expressed as a percentage Values 0 – 100 are valid
Sx_PIN_BOUNCE_MODE	When 1, Custom Pin Bounce Mode is Enabled

Source Custom Pattern

[0x0D..0x07, 0x16..0x10, 0x1F..0x19, 0x28..0x22, 0x31..0x2B, 0x3A..0x34]

Offset	Bits
6	111..96
5	95..80
4	79..64
3	63..48
2	47..32
1	31..16
0	15..0

Name	Description
Sx_CUSTOM_PATTERN	The 112 bit custom pattern is held in 7 sequential registers

Signal Registers

Each signal is controlled by a single register. The low 4 bits of the register stores a single number that describes which source the signal should be following.

Each register assigns the named signal to one of the six control sources, on with HOT_SWAP, always off, or always on:

Nibble Value	Assigned Control Source
0	Always OFF
1	Follow Source S1
2	Follow Source S2
3	Follow Source S3
4	Follow Source S4
5	Follow Source S5
6	Follow Source S6
7	ON when HOT_SWAP is high
8	Always ON

Signal Control

15	14	13	12	11	10	9	8
0	0	0	0	0	0	0	GLITCH ENABLE

7	6	5	4	3..0
0	0	0	0	SIGNAL CONTROL SOURCE

Name	Description
SIGNAL CONTROL SOURCE	1-6 Sets control source for this signal, 0 is always off, 7 on with power up, and 8 always on
GLITCH_ENABLE	Enables the glitch logic to control this signal

Other Registers

0xFFFE – CPLD Part Number Register

15..0
4 digit CPLD part number

Name	Description
PART_NUMBER	Forms the QTLnnnn part number for the CPLD code

0xFFFF – CPLD Version Register

15..8	7..0
MAJOR REVISION	MINOR REVISION

Name	Description
MAJOR_REVISION	Should read as 1 or higher
MINOR_REVISION	Should read as 4 or higher

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

3V3_POWER	
3V3_CHARGE	
5V_POWER	
5V_CHARGE	
12V_POWER	
12V_CHARGE	
SPECIAL1	(MATED on ground pin P4)
PRI_OUT_PL	(Backplane primary out +ve)
PRI_OUT_MN	(Backplane primary out -ve)
PRI_IN_PL	(Backplane primary in +ve)
PRI_IN_MN	(Backplane primary in -ve)
SEC_OUT_PL	(Backplane secondary out +ve)
SEC_OUT_MN	(Backplane secondary out -ve)
SEC_IN_PL	(Backplane secondary in +ve)
SEC_IN_MN	(Backplane secondary in -ve)

Signal Groups

ALL	(Allows change of all signals at the same time)
PRIMARY ¹	(Affects all signals beginning 'PRI_', for the primary port)
SECONDARY ¹	(Affects all signals beginning 'SEC_', for the secondary port)

¹ – Version Specific: These commands are only available from firmware version 3.55 onwards