# TestMonkey 2 – User Manual

## V2.2.0

**29/11/2012**

# Contents


Introduction ..... 4
About ..... 4
What's new ..... 4
Version History ..... 5
Getting Started ..... 7
Getting help ..... 7
Connecting to a Module (Serial) ..... 8
Connecting to a Module (USB) ..... 10
Connecting to a Module (Telnet) ..... 12
Using a 'Demo' Module ..... 14
Adding New Modules ..... 16
Hot-Swap Modules ..... 17
Performing Actions ..... 17
Signal Assignment ..... 18
Hot Swap Patterns ..... 21
Pin Bounce ..... 22
Mux Modules ..... 27
Port Connection ..... 28
Port Forwarding ..... 30
Port Off ..... 32
Mux Modules Config ..... 33
Array Controllers ..... 34
CRITICAL NOTES ..... 34
Connection to a Module on an Array ..... 34
Exporting Scripts ..... 36
Custom Module Names ..... 37
Controlling Multiple Modules ..... 39
Programmable Power Module ..... 40
Important Notes! ..... 40
Main Screen ..... 41
Power Up ..... 44
Margining Output Voltage ..... 45
Averaging Levels ..... 46

Output Patterns ..... 47
Repeating a Pattern ..... 50
Trigger Modes ..... 52
Record Triggers ..... 52
Pattern Triggers ..... 53
View Recorded Data ..... 54
Export Data ..... 57
Statistics ..... 58

# Introduction

## About

### Quarch TestMonkey 2



**Software Requirements**

- Microsoft Windows XP, 2000, Vista, Windows 7
    - Some versions of Windows Server may also work but this is not specifically supported
- Microsoft .NET framework 4.0

**Hardware Requirements:**

- Standard x32 or x64 PC
- Optional RS-232 Serial port for control of a module
- Optional USB port for control of a module
- Optional Ethernet port for control of a module

## What's new

**Enhancements**

TestMonkey 2 is a complete re-write of the TestMonkey suite. This was done allow more rapid innovation as new modules and abilities are added to the Torridon product line.

**Now updated to run on DotNet 4.0!**

# Version History

**v2.2.0 - Power Module Updates**

- Added support for modules with external triggering
  - Trigger out when pattern is run
  - Trigger out when recording starts
  - Trigger out when Current / Power exceeds a set limit
  - Trigger in can start a pattern running
  - Trigger in can start recording
- Added support for Output Limits
  - Requires firmware version 4.101 or higher
  - Allows a hard limit to prevent output voltages exceeding set levels
- Various changes to GUI due to the large number of additional controls

**v2.1.10 - Bug Fixes**

- Fixed bug where PC's that had Visual Styles disabled (or were using remote desktop), would crash the application

**v2.1.9 - Support for v2 glitch engine**

- Added support for extended glitching on HS Modules (Modules with updated FW only)
  - PRBS patterns extend from 1:256 up to 1:65536
  - Glitch length options extended from 31 up to 255 levels
  - Glitch cycle length now set in the same way as the glitch length (same range limits)

**v2.1.8 - Power Module Updates**

- Added config file for -03 version modules
- Added commands to monitor and reset fault states
- Added 6.5v regulator monitoring
- Improved PPM trace export to 1024x768 in JPG format

**v2.1.7 - Power Module Updates**

- Various minor fixes to the power pattern editor
- Added option to view repeating patterns
- Added ability to load/save power patterns
- Further fixes for issues with larger forn sizes

**V2.1.6 - New Module Supported**

- Added initial support for PCIe Card Module
- Various display fixes for use with enlarged font sizes

**V2.1.5 - Power Module Update**

- Added calibration support for QTL1455 current measurements

**V2.1.4 - New Module Supported**

- Supports Programmable Power Module - Firmware v4.005 and higher
- Updated to .NET Frameword 4.0. This is now a requirement for the software to run!

**V2.1.2 - Beta release**

- Beta version to support Programmable Power Module - not for general release
- Updated to .NET Frameword 4.0. This is now a requirement for the software to run!

**V2.1.1 - Minor Updates**

- Added support for QTL1521 (03 version) HD mini SAS module
- Added support for QTL1490-01 SATA Keyed HS module
- Fixed an error when an Array controller is connected without any sub-modules attached

**V2.1.0 - Array Controller Support**

- Modules can be controlled through an array controller
- Multiple modules can be controlled at the same time

**V2.0.17 - New Module Supported**

- Added support for Mini SAS HD Mux module

**V2.0.17 - Minor Updates**

- Added ability to give custom names to a module
- Removed invalid 'ALL' signal from SBB module config file
- Updated connection screen to make it more simple

**V2.0.16 - Minor Fixes**

- Further update to DP Mux module config file
- Implovements to error logging window
- Fixed an exception when USB device is removed without warning

**V2.0.15 - New Module Supported**

- Added support for Display Port Mux Module

**V2.0.14 - Config Updates Only**

- Updated SBB 2.0 Config File

**V2.0.13 - Full Support for MUX modules**

- Added full API control for Mux modules
- Added graphical display of Mux connections
- Added control over Mux connection delay

***Other Changes***

- Added control over user pin bounce patterns for hot-swap modules
- Added warning on connection screen if modules are connected but currently locked to another process
- Added telnet control for PoE modules

**V2.0.12 - New Modules Supported**

- Minor change to support connection delays on Mux modules

**V2.0.11 - New Modules Supported**

- Added basic support for USB and SATA Mux modules. This will be updated and improved in upcoming releases. For now it allows full control over the basic Mux functions.

**V 2.0.10 - Bug Fix**

- Fixed: Prevented USB disconnection warning when a module is controlled by RS-232 serial but also has a USB cable attached

**V 2.0.9 - Minor Updates**

- Fixed: Manual updated to show new layout of connection screen

# Getting Started

## Getting help

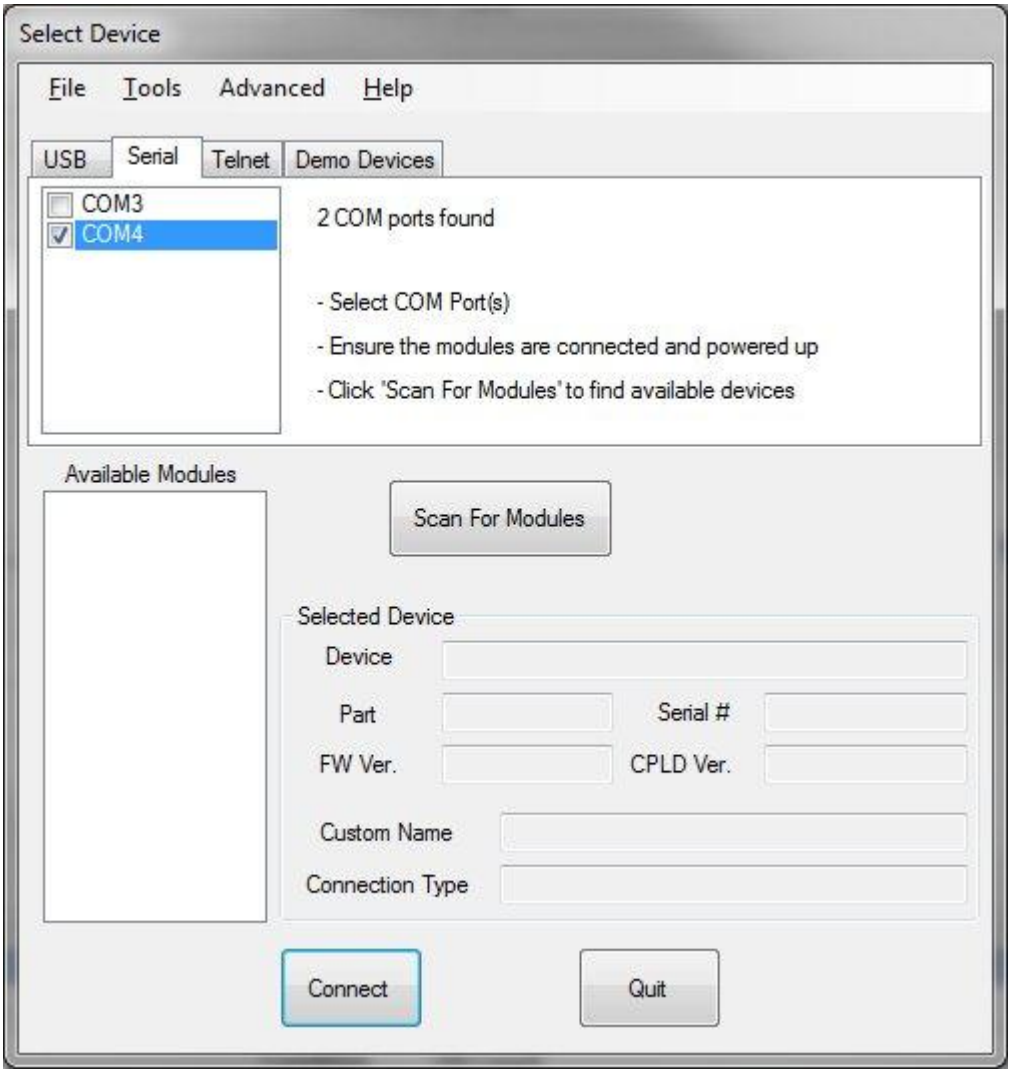
TestMonkey 2 is a constantly evolving product.  We would greatly appreciate your feedback if you find any bugs or have suggestions that could improve our software.

**Email:**
Support@Quarch.com

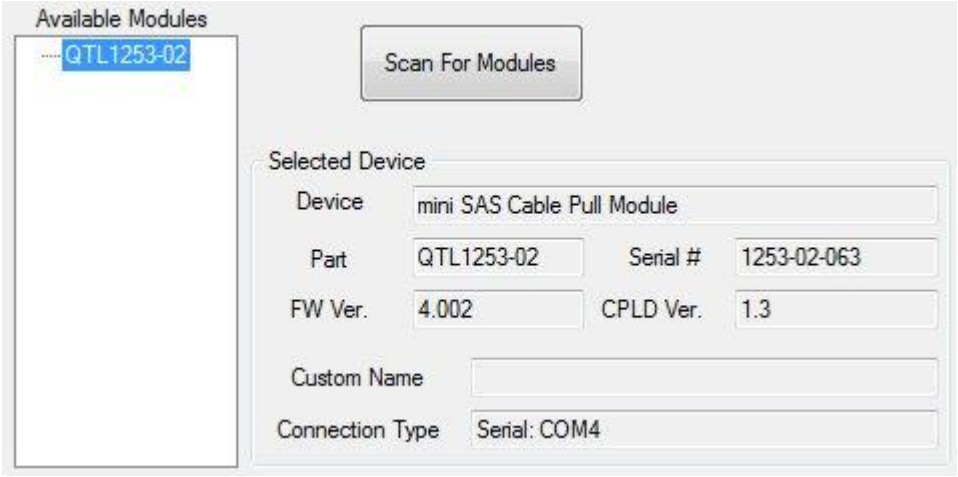
**Call:**
+44 1343 508 140

## Connecting to a Module (Serial)

- Connect your Torridon Module using an Interface Kit and the supplied Serial DB-9 cable
- Power up the module
- Start TestMonkey



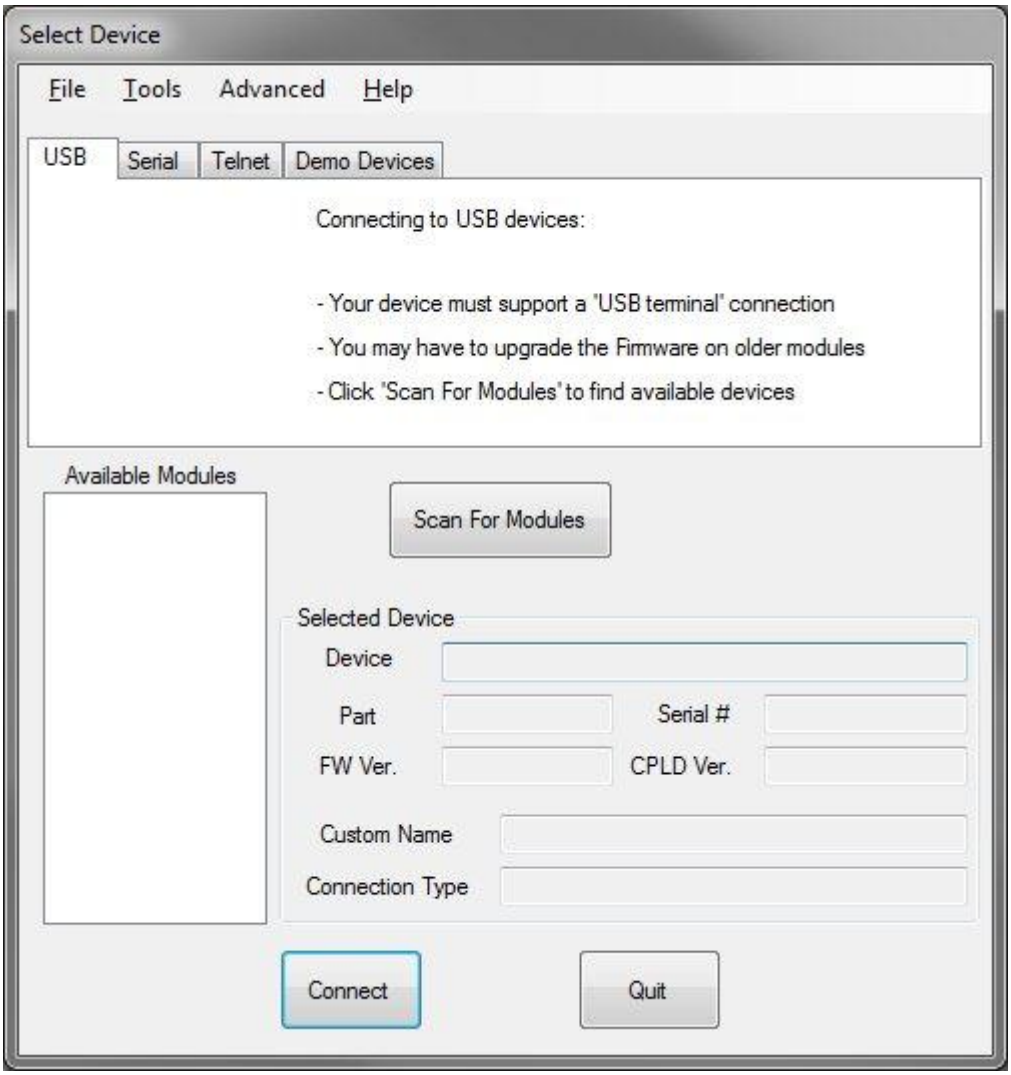

- Select the 'Serial' tab
- Select 1 or more of the COM ports on your PC to scan for Torridon Devices
- Press ***Scan For Modules*** to look for the connected devices



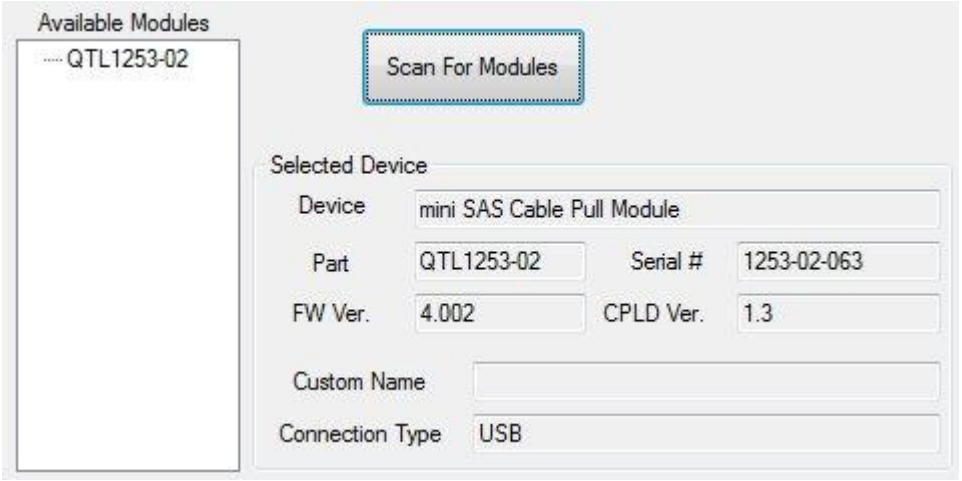

- Select the Module to use. Module details for the selected device are displayed.
- Press ***Connect*** to use the selected module

## Connecting to a Module (USB)

- Connect your Torridon Module using an Interface Kit and a USB cable
- Power up the module
- Install the driver
    - Download the latest version of the "Quarch USB Driver" from the Quarch website
    - Follow the install instructions in the READ_ME file
- Start TestMonkey




- Select the 'USB' tab
- Press ***Scan For Modules*** to look for the connected devices

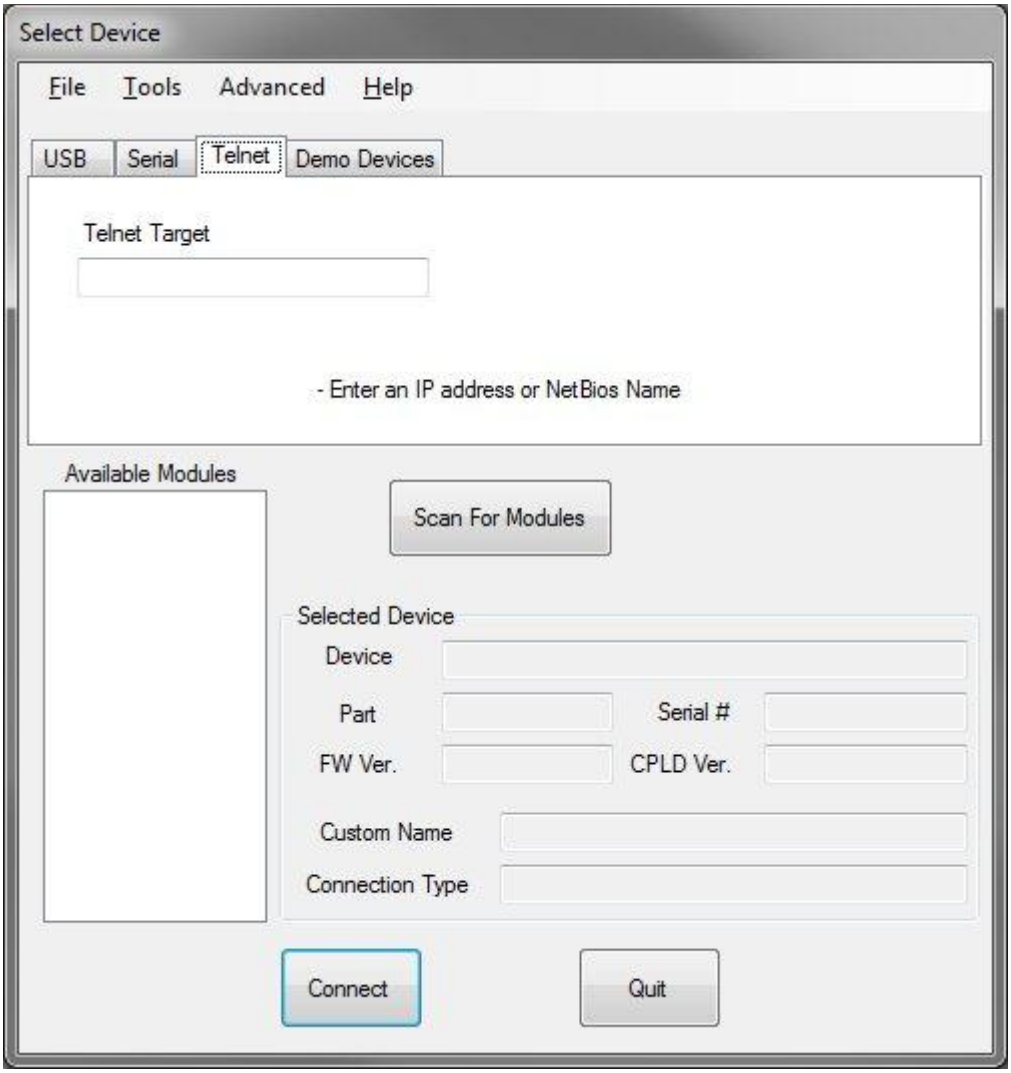



- Select the Module to use. Module details for the selected device are displayed.
- Press ***Connect*** to use the selected module
- Modules require firmware version 4.0 and the latest "Quarch USB Driver" or higher to be controlled by USB
    - QTL1069 (Legacy SBB module) can use USB from firmware version 3.61

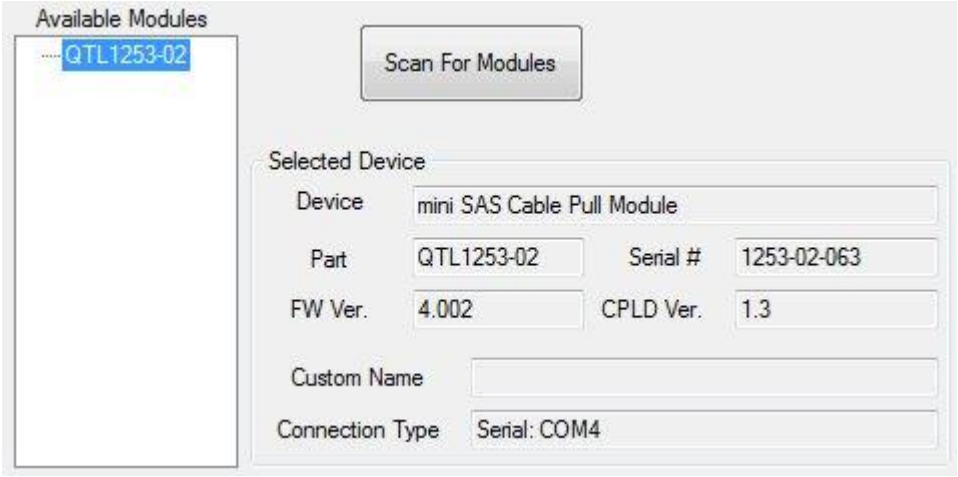
## Connecting to a Module (Telnet)

TestMonkey can only open a Telnet connection to modules that have a native Ethernet port. This includes the PoE (Also called stand-alone) MUX modules. Although the Array Controller also has an Ethernet port, you cannot control it, or it's attached modules through TestMonkey.

- Connect your Torridon Module to the LAN
- Ensure that LAN permissions and firewalls allow access to the device from your PC
- Power up the module
- Start TestMonkey




- Select the 'Telnet' tab
- Enter the target device IP address or NetBios name
- Press ***Scan For Modules*** to look for the connected devices. This may take a few seconds




- Select the Module to use. Module details for the selected device are displayed.
- Press ***Connect*** to use the selected module

**Connection Problems**

If you cannot find the module IP/Name:

- The command "***conf:eth:ip?***" on a serial/usb terminal will give the IP address
- The command "***conf:eth:name?***" on a serial/usb terminal will give the NetBios name

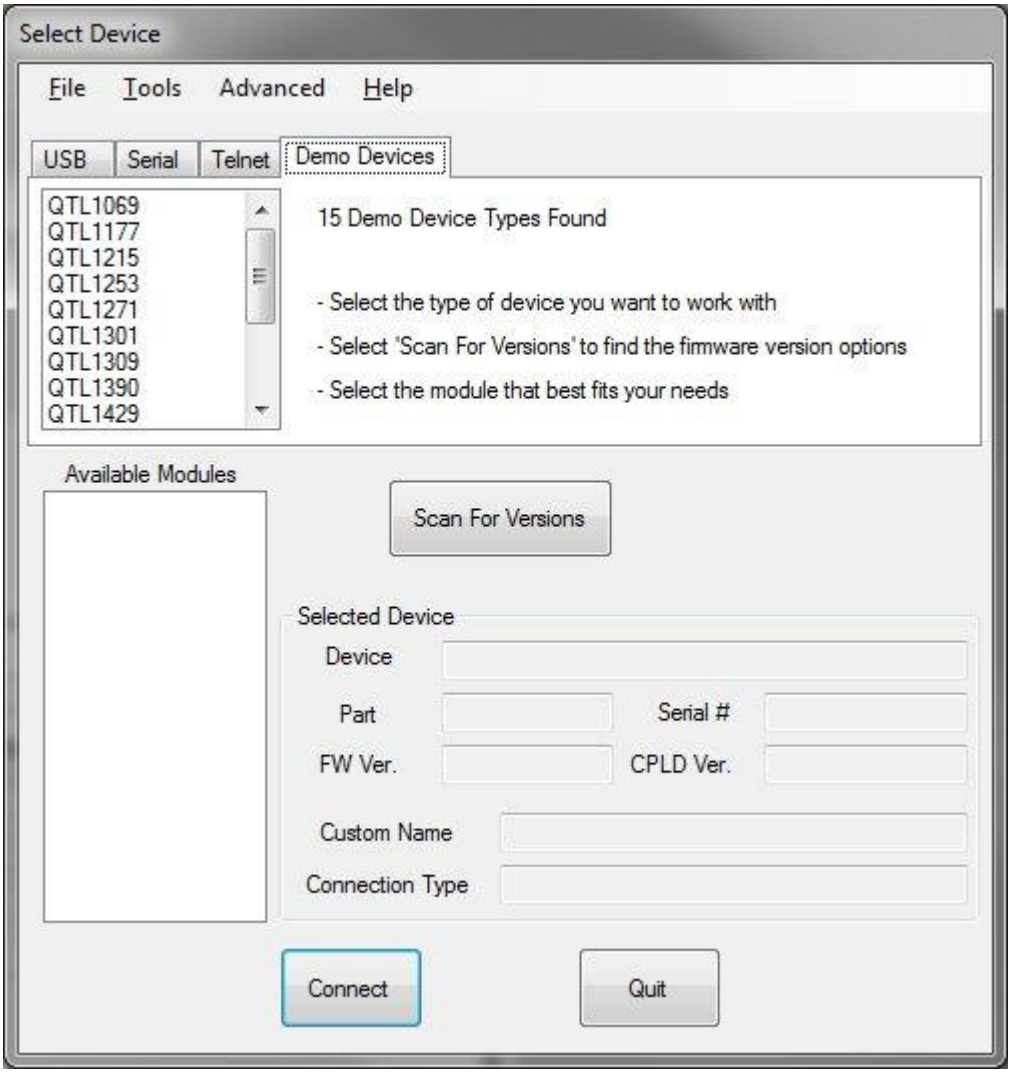
If you are sure of the IP/Name but the connection does not work:

- Run "***cmd.exe***"
- run the command "**ping xxx**" where xxx is the IP address or NetBios name. If the ping fails, you may have a problem with your network permissions or cabling

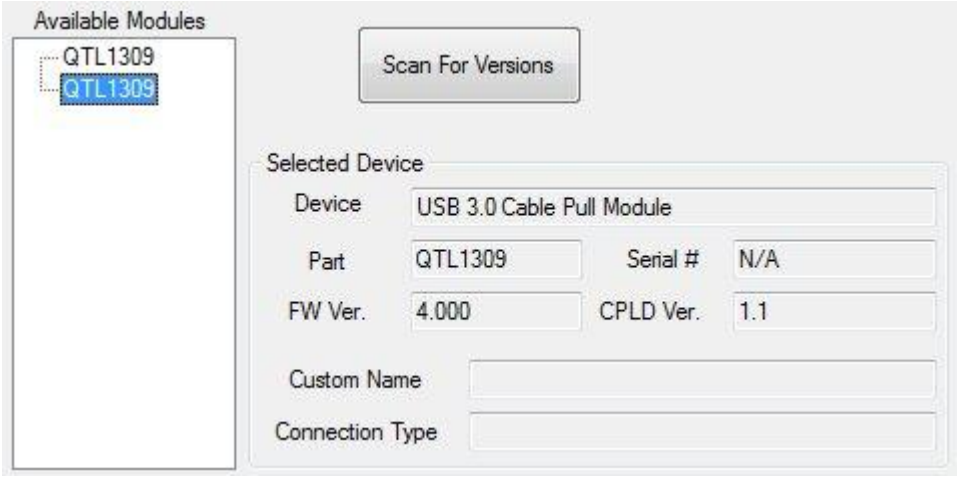
## Using a 'Demo' Module

TestMonkey 2 supports the use of virtual 'Demo' devices. This allows you to generate scripts for a module or just try out the system without needing any connected Torridon hardware

- Start TestMonkey
- Select the ***Demo Devices*** tab




- Select the part number of the Torridon device you want to demo
- Press ***Scan For Versions*** to look for matching configuration files




- Select the Module configuration to use. Details for the selected device are displayed. If multiple selections are available, choose the one that matches the firmware version of the device you want to control with any exported script
- Press ***Connect*** to use the selected module

## Adding New Modules

If you have a new module that is not supported by your version of TestMonkey, you can add this:

This can be done by installing the latest version of TestMonkey. Alternatively, you can just import the config file you need:

- Contact Quarch to get an updated config file
- Open TestMonkey 2
- On the connection screen, select ***Tools->Import Config File***
- Select the file(s) to add
- You should now be able to use the new module as normal

# Hot-Swap Modules

## Performing Actions

On startup, TestMonkey 2 reads the current state of the module and loads the settings into the GUI.

Modules have a default setup that includes a basic hot-swap timing sequence. Most modules start 'plugged' while older Drive modules start 'un-plugged'.

### Hot Swapping

You can simulate a hot-plug or hot-pull of the device. This sequences the connections at precise times as would happen in a real hot-swap. Pin bounce
can also be simulated although this is not done by default.

- Press ***Plug Module*** or ***Pull Module*** to initiate a hot-swap event. The current Source and Signal timing patterns will be used

### Repeated Hot Swapping

If you want to perform a number of repeated hot-swaps, you can setup an automated hot-swap cycle

- Enter an ***On Time*** (the time to keep your device connected)
- Enter an ***Off Time*** (the time to keep your device disconnected)
- Optionally, set the ***Cycles to Run*** if you want a limited number of repeats
- Press ***Start Cycling*** to begin the hot-swaps
- **WARNING**: Rapid hot-swaps may damage your device, especially drives that take several seconds to power up

### Glitching Signals

You can interrupt one or more Signals by using Glitching. This function is not available on all modules. You will find this function on HS Drive modules (NOT HS Lite) and Cable Pull Modules.

The glitch length is set by an initial factor (50nS, 500nS, 5uS, 50uS, 500uS, 5mS, 50mS, 500mS). This is multiplied by a second number (1 to 31 on old Firmware, or 1 to 255 on newer Firmware).
TestMonkey will allow you to enter any value, but will correct it to the closest available value.

- Enter the ***Glitch Length***. Most Glitch modules support between 50ns and 15.5S for the glitch length
    - Modules with updated Firmware extend this to 127.5 seconds
- Select the Signals to glitch on the ***Signal Assignment*** tab (See help on Signal Assignment)

**Single Glitch**

- Press ***Glitch Once*** to create a single glitch on all the selected Signals

**Cycling Glitch**

- Enter a ***Glitch Cycle Count***. (Glitch Length * Glitch Cycle Count) = the gap between glitches
  - Modules with updated Firmware set the Glitch Cycle length in the same way as the Glitch Length
- Press ***Start Glitch Cycle*** to begin the glitch sequence
- Press ***Stop Glitching*** to end the repeating glitch sequence

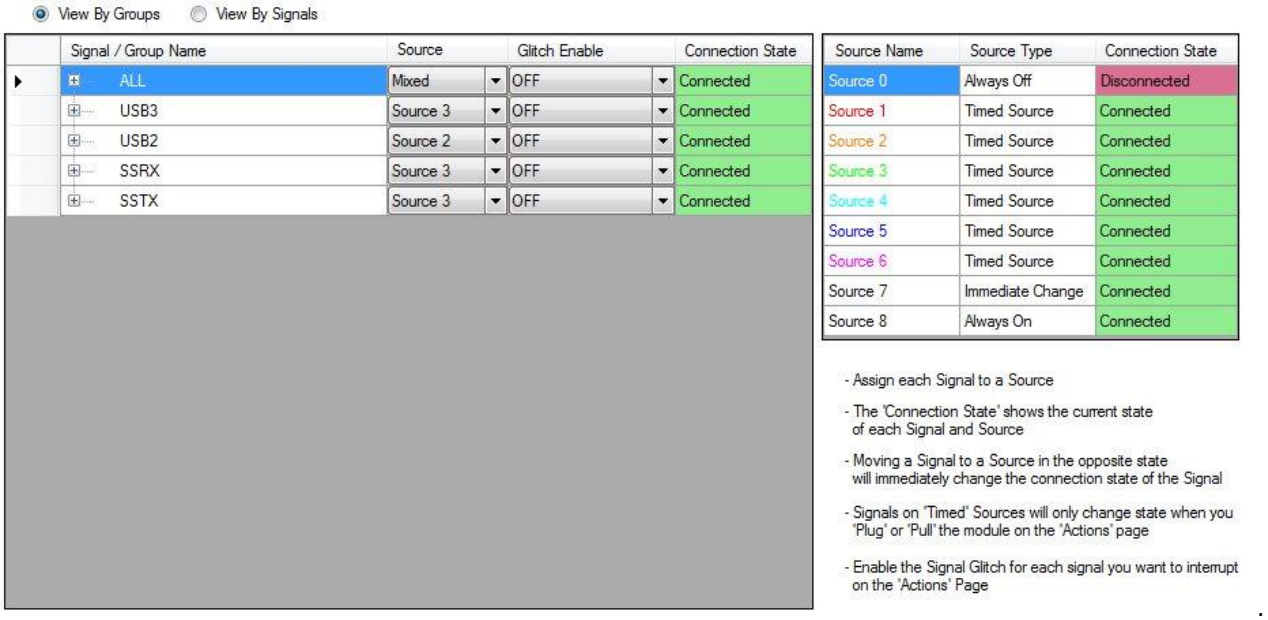
**PRBS Glitch**

- Select a ***PRBS ratio***. The ratio defines the ratio of glitch to no-glitch
- Press ***Start Glitch PRBS Sequence*** to begin a Pseudo Random sequence of glitches

Both Cycle and PRBS glitches keep running after the button is pressed. To stop repeating glitches, press ***Stop Glitching***

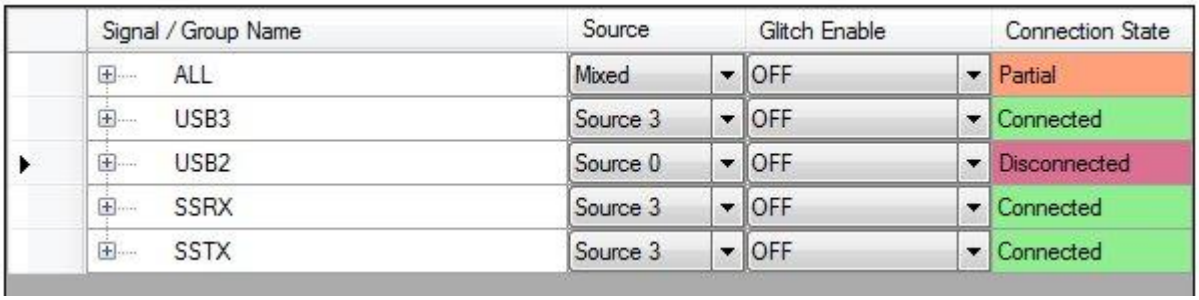
## Signal Assignment

Each Signal is assigned to a Source which controls its connection state. Moving a Signal to a Source will cause it to immediately connect or disconnect if the state of the new Source is different.



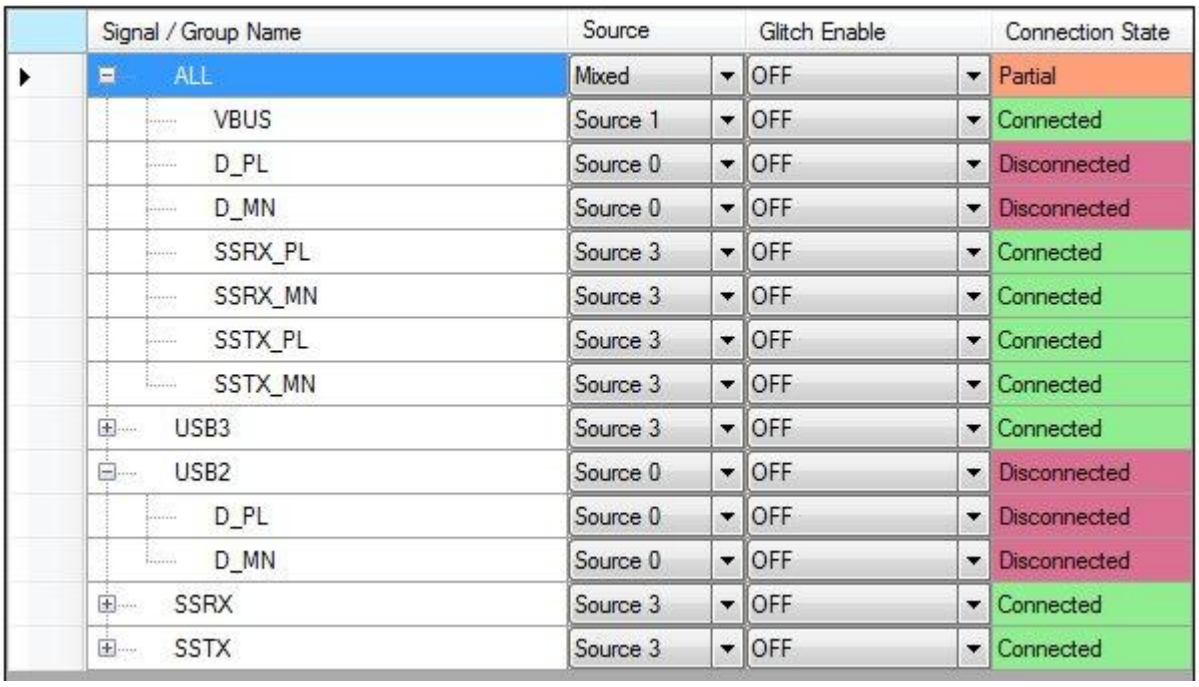

This module is 'plugged' and all signals are connected

- To simulate a failure of the USB2 group (which contains all USB2 related signals)
  Select the Source drop down for ***USB2*** and assign it to ***Source 0*** (a special Source that is always off)



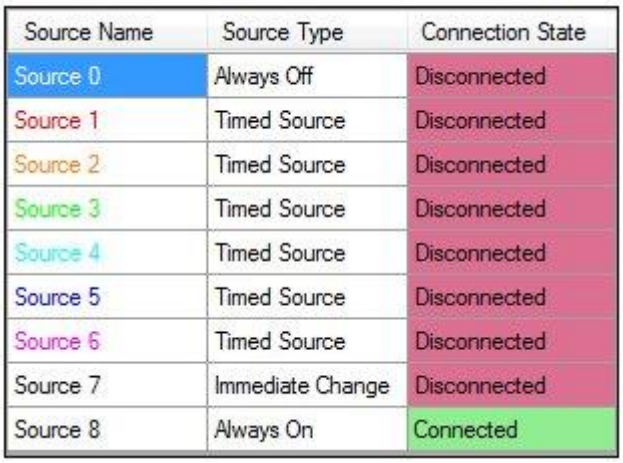
| Signal / Group Name | Source | Glitch Enable | Connection State |
|---|---|---|---|
| ALL | Mixed | OFF | Partial |
| USB3 | Source 3 | OFF | Connected |
| USB2 | Source 0 | OFF | Disconnected |
| SSRX | Source 3 | OFF | Connected |
| SSTX | Source 3 | OFF | Connected |

When the selection is made, the state of the signals will occur immediately. We can now see that the ***USB2*** signal group is disconnected.
The ***All*** group is showing partial connection as some, but not all of it's signals are connected.
We can expand the groups to see the signals they contain by clicking on the **+** sign



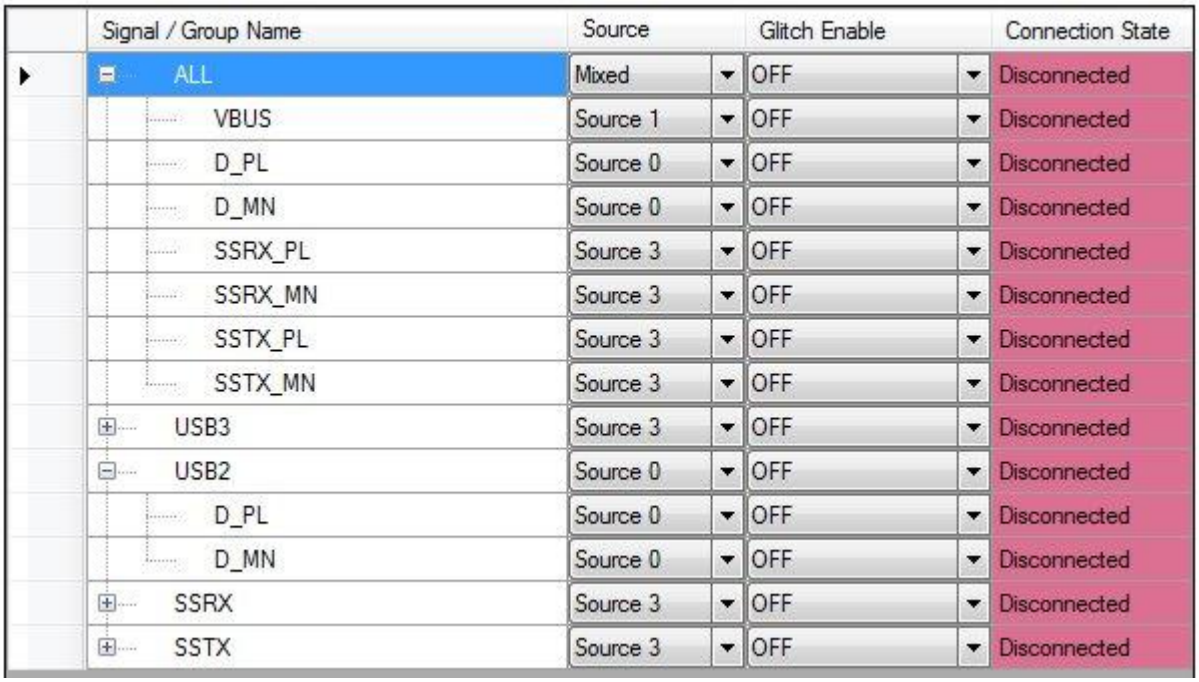
| Signal / Group Name | Source | Glitch Enable | Connection State |
|---|---|---|---|
| ALL | Mixed | OFF | Partial |
| VBUS | Source 1 | OFF | Connected |
| D_PL | Source 0 | OFF | Disconnected |
| D_MN | Source 0 | OFF | Disconnected |
| SSRX_PL | Source 3 | OFF | Connected |
| SSRX_MN | Source 3 | OFF | Connected |
| SSTX_PL | Source 3 | OFF | Connected |
| SSTX_MN | Source 3 | OFF | Connected |
| USB3 | Source 3 | OFF | Connected |
| USB2 | Source 0 | OFF | Disconnected |
| D_PL | Source 0 | OFF | Disconnected |
| D_MN | Source 0 | OFF | Disconnected |
| SSRX | Source 3 | OFF | Connected |
| SSTX | Source 3 | OFF | Connected |

Currently all the timed Sources are Connected. This is because the module is in the plugged state. If we click ***Pull Module*** on
the ***Actions*** page, all the timed sources will be disabled:



| Source Name | Source Type | Connection State |
|---|---|---|
| Source 0 | Always Off | Disconnected |
| Source 1 | Timed Source | Disconnected |
| Source 2 | Timed Source | Disconnected |
| Source 3 | Timed Source | Disconnected |
| Source 4 | Timed Source | Disconnected |
| Source 5 | Timed Source | Disconnected |
| Source 6 | Timed Source | Disconnected |
| Source 7 | Immediate Change | Disconnected |
| Source 8 | Always On | Connected |

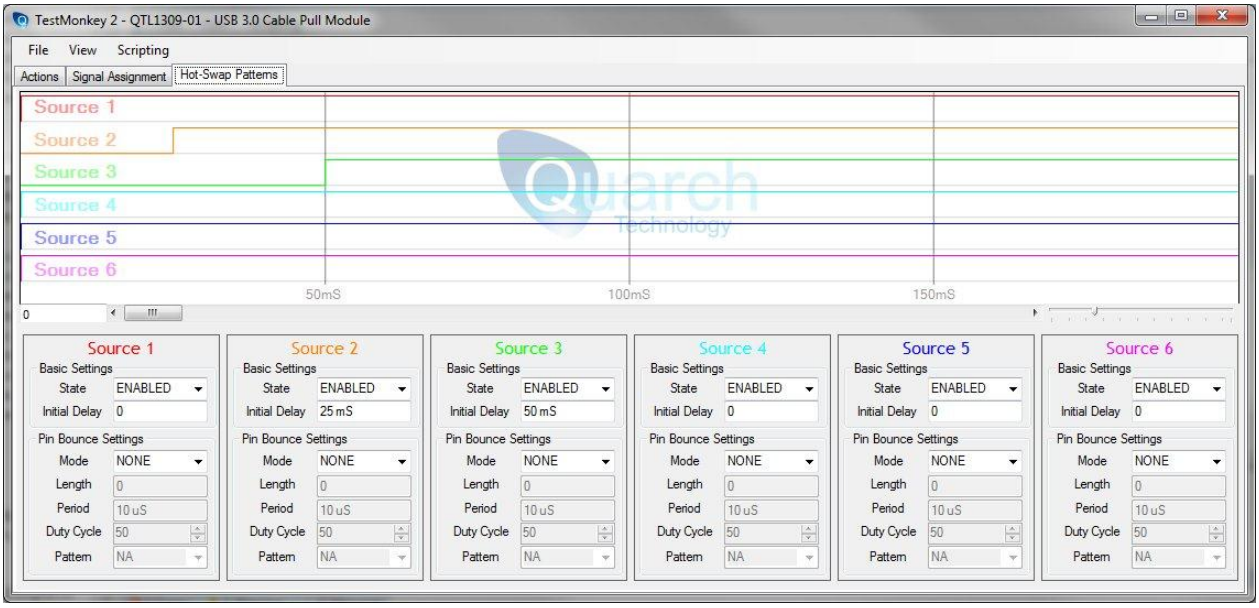
Ass the Sources are disabled, all the Signals assigned to them will be disabled as well:



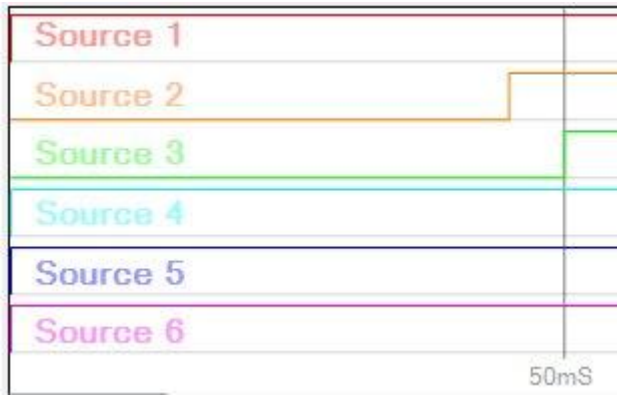
| Signal / Group Name | Source | Glitch Enable | Connection State |
|---|---|---|---|
| ALL | Mixed | OFF | Disconnected |
| VBUS | Source 1 | OFF | Disconnected |
| D_PL | Source 0 | OFF | Disconnected |
| D_MN | Source 0 | OFF | Disconnected |
| SSRX_PL | Source 3 | OFF | Disconnected |
| SSRX_MN | Source 3 | OFF | Disconnected |
| SSTX_PL | Source 3 | OFF | Disconnected |
| SSTX_MN | Source 3 | OFF | Disconnected |
| USB3 | Source 3 | OFF | Disconnected |
| USB2 | Source 0 | OFF | Disconnected |
| D_PL | Source 0 | OFF | Disconnected |
| D_MN | Source 0 | OFF | Disconnected |
| SSRX | Source 3 | OFF | Disconnected |
| SSTX | Source 3 | OFF | Disconnected |

## Hot Swap Patterns

On the ***Hot Swap Patterns*** tab, you can alter the timing profile of each timed Source. The default timings here shows a 3 step connection pattern at 25ms spacing

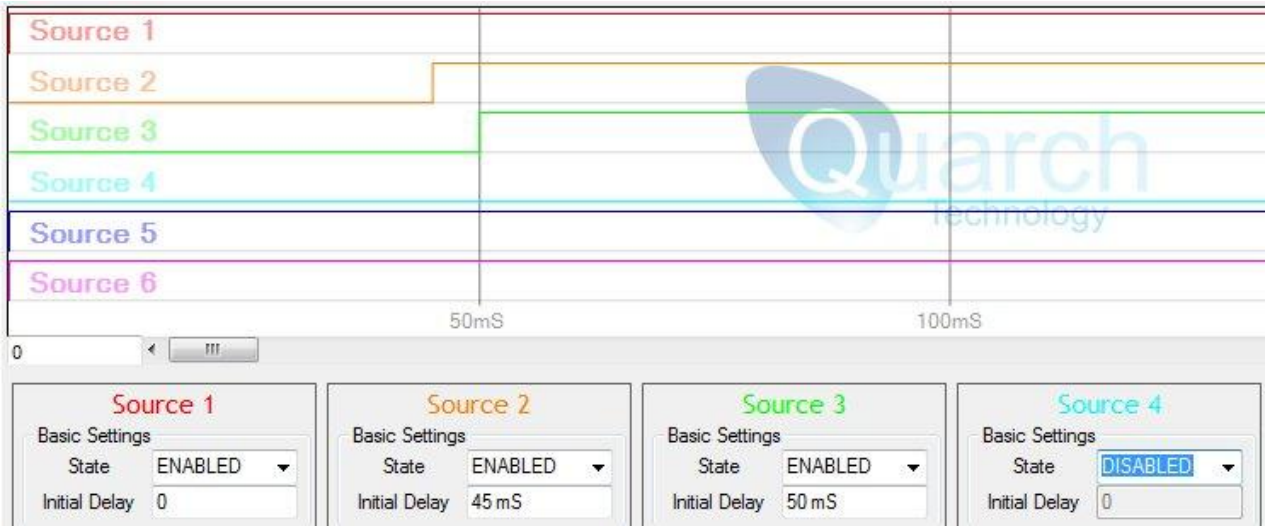



- Drag and drop the initial delay values to the time you want, or edit them directly in the ***Initial Delay*** box




The timing graph shows the new connection sequence with a much shorter time between the connection of Source 2 and Source 3

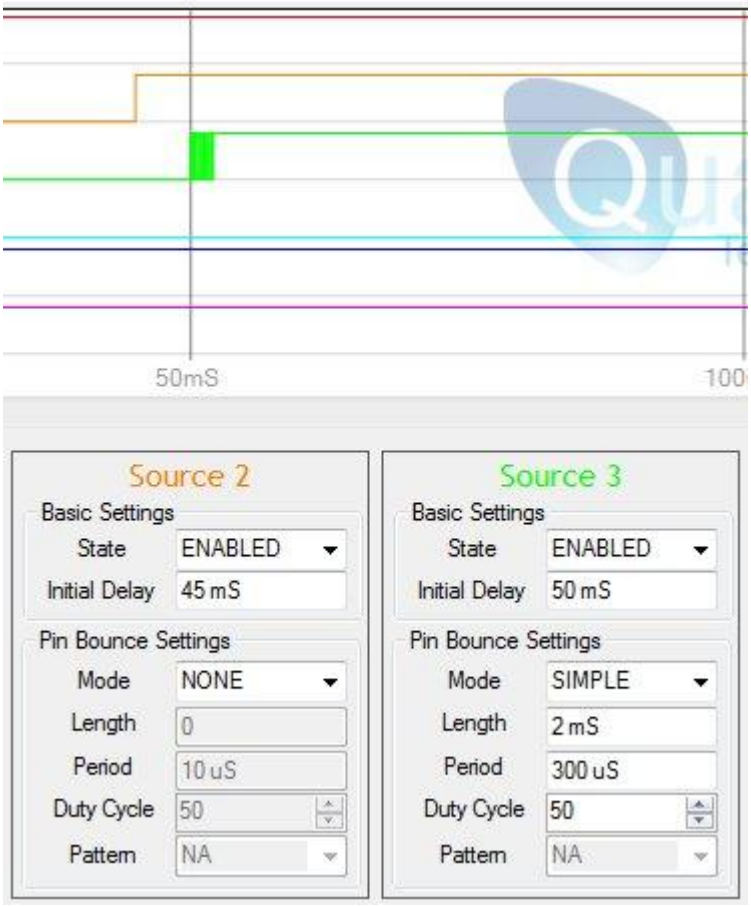
- Change the ***State*** of ***Source 4*** to ***DISABLED*** to turn off the Source completely. This will disconnect any Signals assigned to it and stop them from changing during hot-swap events




## Pin Bounce

More advanced modules, such as the HS Drive module and most Cable Pull modules, support pin bounce. This allows the simulation of an erratic connection during the time that the cable is mating. In real life, this is caused by the pin contacts rubbing and bouncing along each other as the connection is made.

- Turn on Pin-bounce on Source 3: Select ***SIMPLE*** from the Pin Bounce ***Mode*** setting
- Select a bounce ***Length*** of 2 mS and a ***Period*** of 300 uS




The Pin-Bounce region is now shown on the timing graph.

- Use the slider control below the graph (or the mouse wheel) to zoom in on the bounce region


This is the precise bounce pattern that will be applied on hot-plug. The pattern will be applied on both plug and pull and begins immediately after the initial delay for the source has expired.

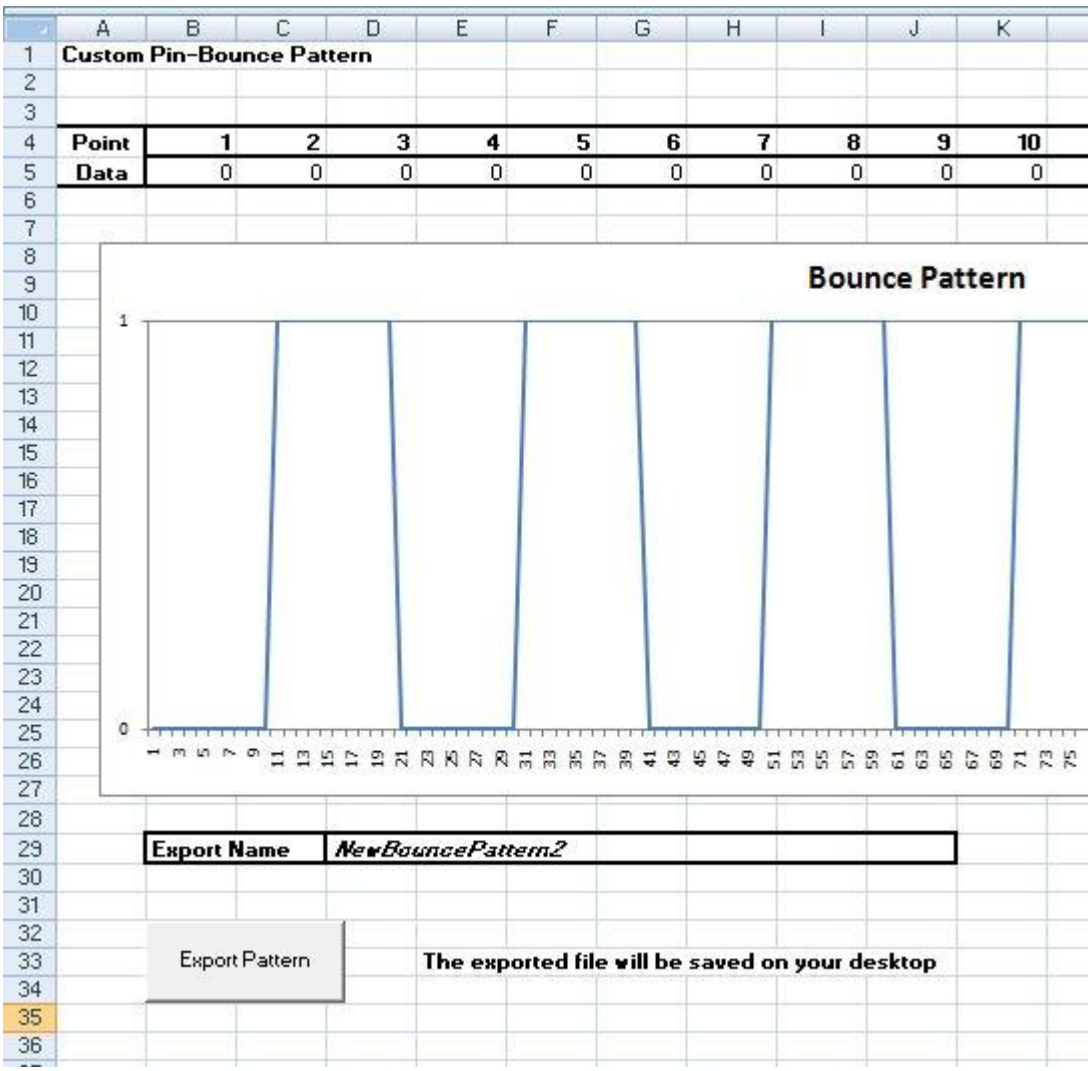
Pin Bounce Parameters:
'***Length***' = This is the total time that the pin bounce will run for in mS
'***Period***' = The period of the square wave which can be as low as 10uS
'***Duty Cycle***' = A 50% duty cycle is the default and will produce an equal square wave. Higher values will mean more of pattern is 'on'

## *Custom Pin Bounce*

Most modules that support pin bounce also have the option for custom pin bounce patterns. Instead of describing a simple repeating wave, the user can program a fixed length bounce pattern (normally 112 bits long). This pattern will be used to produce the bounce effect.

- Create a custom pattern

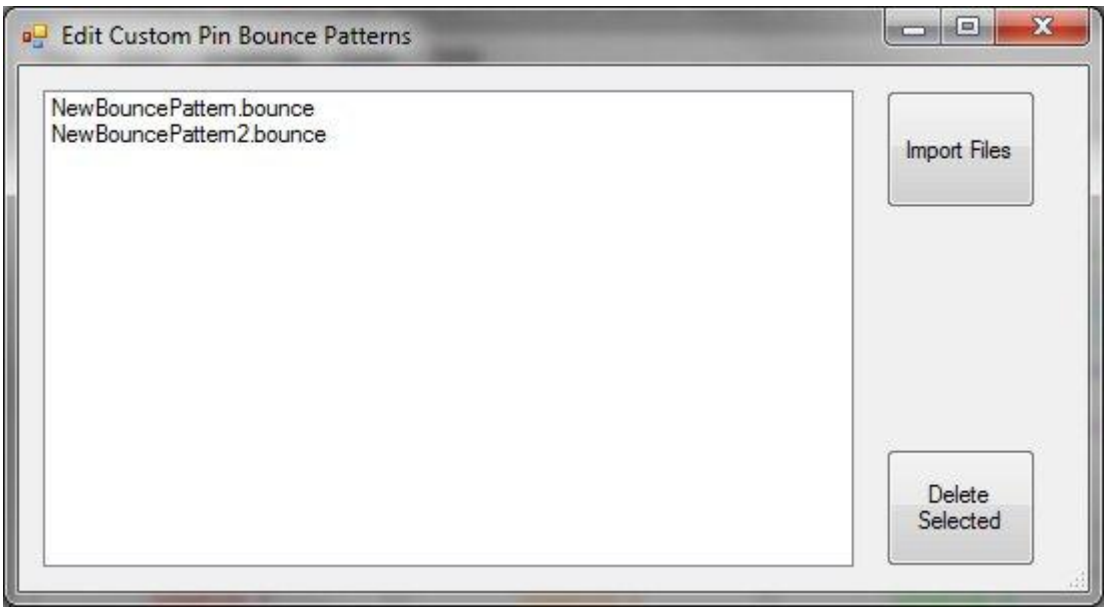
***1.*** Select ***Tools->Create Custom Patterns***
2. A macro enabled spreadsheet will open
3. Edit the data points as required and export the pattern



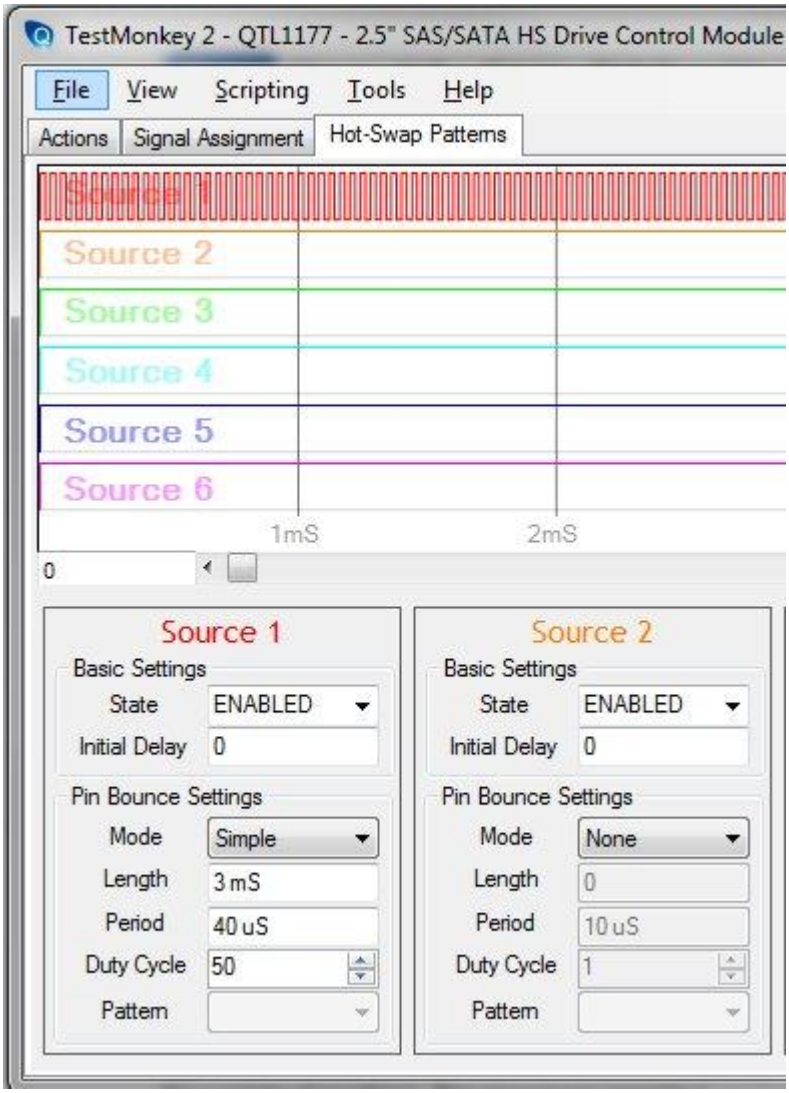

NOTE: The saved pattern file is a simple ASCII text file with one line of text. 112 characters (either 1 or 0) describe the pattern. You can also create these patterns yourself using any normal test editor! Any files you create must have an extension of '.bounce'

- Import a new custom pattern

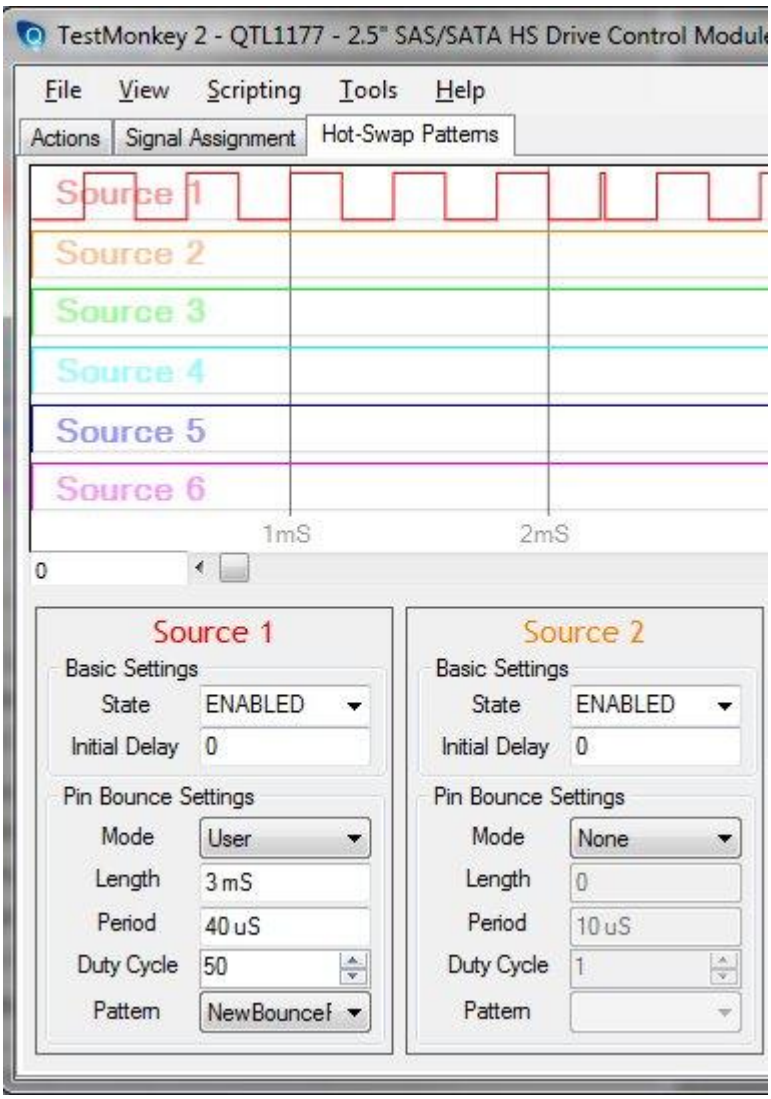
***1.*** Select ***Tools->Import Custom Patterns***
***2.*** Select ***Import Files***
3. Browse to and select the files to import, these will be copied to the TestMonkey data folder
4. The new pattern will now be available to use




On the source control, we can now select '***User***' from the '***Mode***' selection drop down




Finally, select the custom pattern to use from the '***Pattern***' drop down. Different custom patterns can be chosen for each source if required.



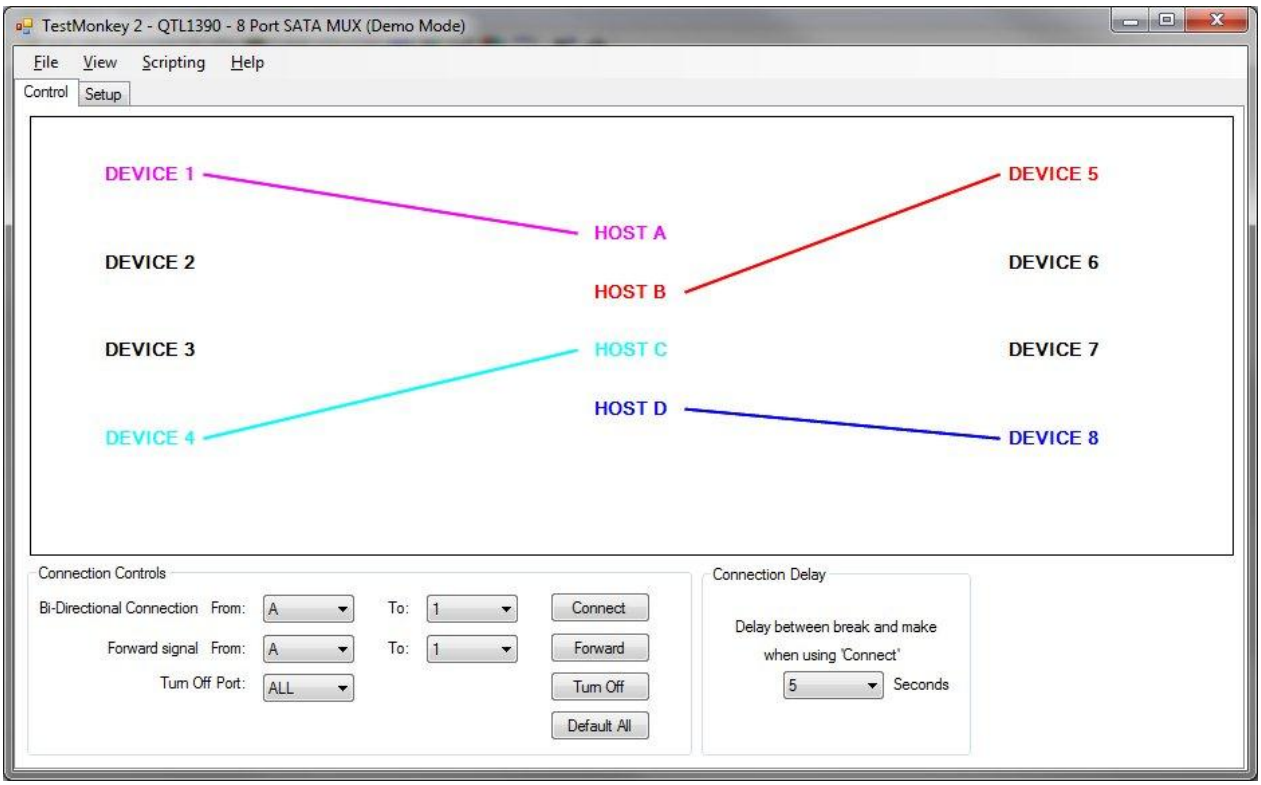

The new pattern is not shown on the timing display. The example pattern is another simple square wave but of a different frequency. The extra spike just after 2mS is the discontinuity at the end of the pattern.

***Length*** = This is the total length of bounce that will occurr
***Period*** = This is the time taken up by ***2*** bits of the pattern (Pattern length = (112 / 2) * Period)
***Duty Cycle*** = This has no effect on custom patterns

In this example:
Pattern Length = ~2.2mS, thus the pattern will have to repeat 1 and a bit times (creating the discontinuity and the repeat point)

# Mux Modules

If a MUX/Physical Layer Switch product is attached, you will be given a different control screen:




- The number of ports shown will depend on the module attached.
- Each port shows the connection(s) it has. Where possible, the port label and line color is changed to show this
- A solid line (as above) indicates that the parts are fully connected with a bi-directional link

In the above example, 4 bi-directional (full) port connections have been made:
A and 1, B and 5, C and 4, D and 8

**Underlying Architecture**

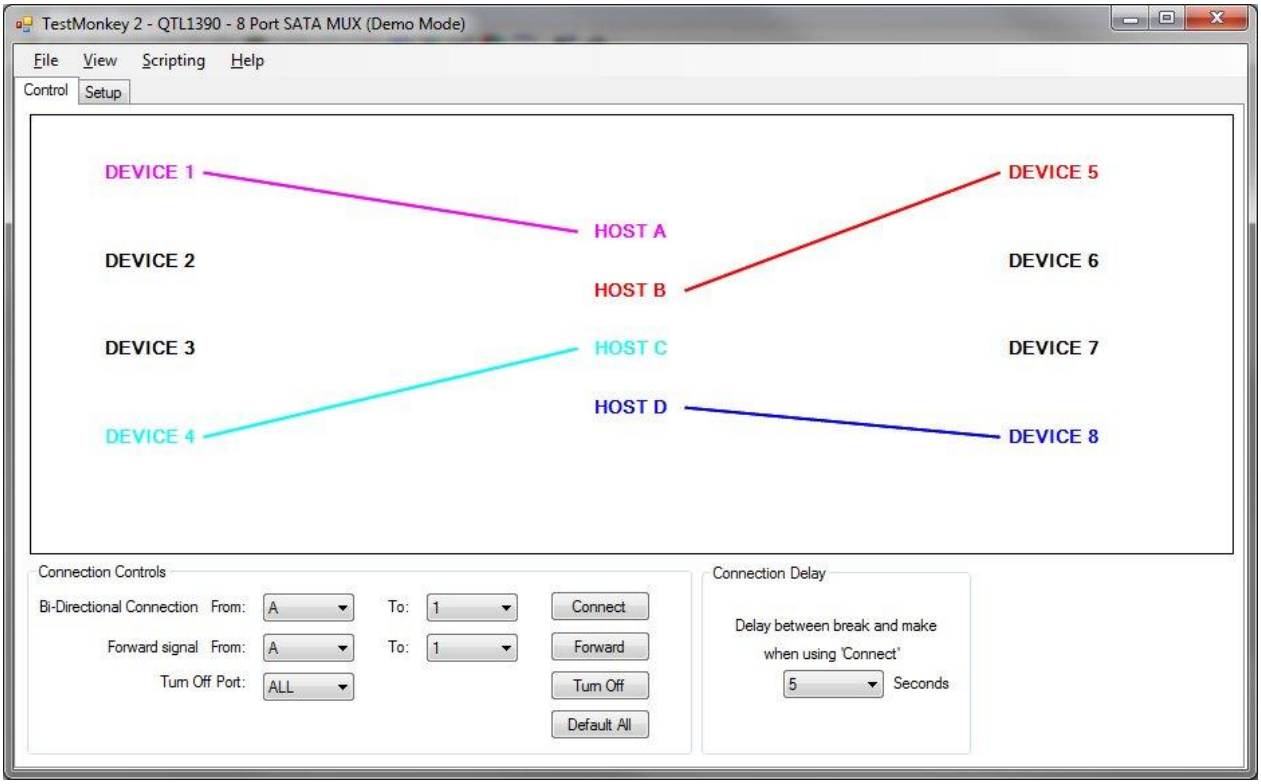
Each port on the Mux is a single 'Item' that has a number of settings. Parameters such as Amplitude Pre-Emphasis and Equalisation are set per-port and are found on the ***Setup*** tab of the application.

To setup data connections, each port has a single parameter that specifies where it's data will come from. To have a 'full' connection between **A** and **1** we must set the source of **1** to **A** and the source of **A** to **1**. Unlimited forwarding links can be created. To forward **A** to multiple other ports, we must set those other ports to take data from **A**.

Links are normally created between a host port and a target device port. It is also possible (on 4-8 Port Mux modules) to create links between 2 Host ports or 2 Devices ports. This can be useful setting up routing that would allow an analyser to be switched in and out of a signal path.

## Port Connection

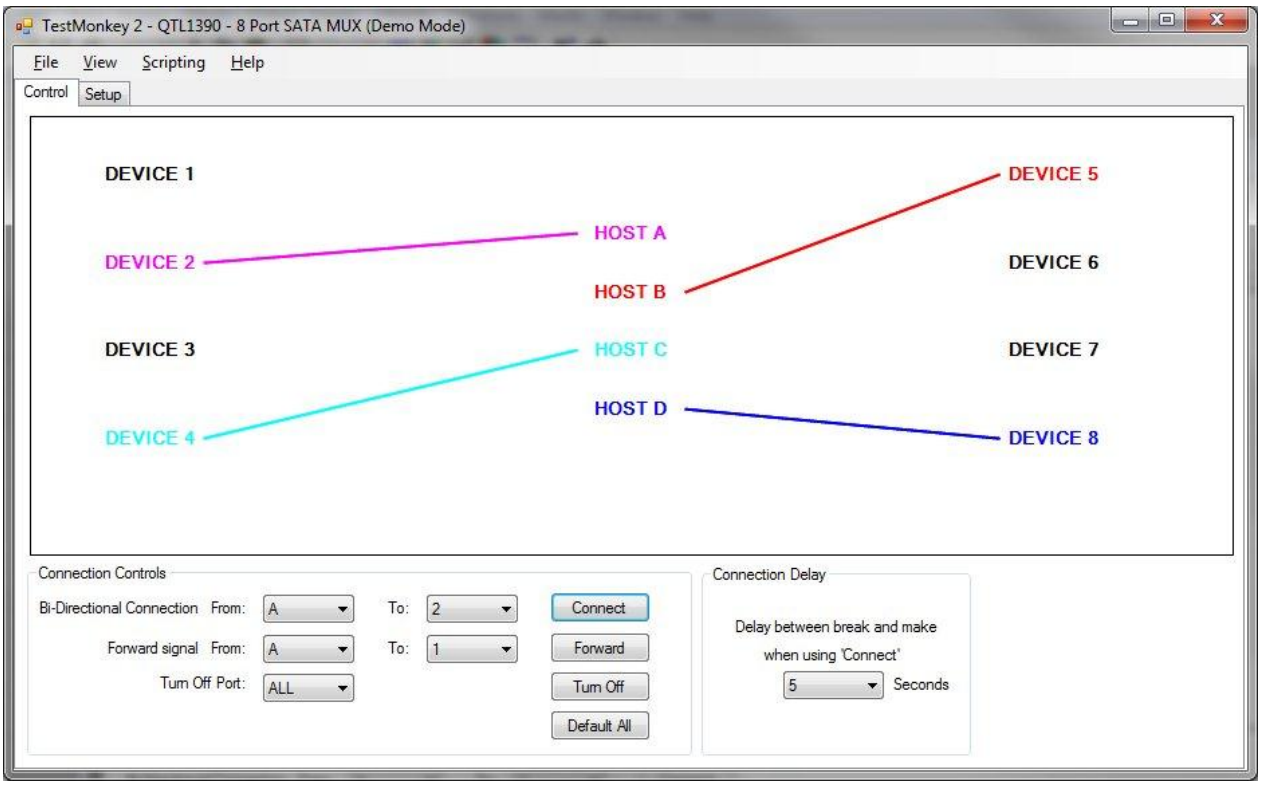
In the default state, a number of connections are already made.




To make a new bi-directional (full) connection between 2 ports:

1. Select the host port in the ***From*** selection
2. Select the device port in the ***To*** selection
3. Click ***Connect***

For example, connect ***A*** and ***2*** This will create a new link in both directions between ***A*** and ***2***. Any other links with either ***A*** or ***2*** will be removed, as seen below. Device port 1 is no longer connected to anything.




**Connection Timing**

Any connection command that is run follows the same flow:

1. Break EVERY link that comes from either of the two ports
2. Delay for a set time (Set by the ***Connection Delay*** box: 5 seconds on the above screen)
3. Create a simultaneous connection in both directions between the two selected ports

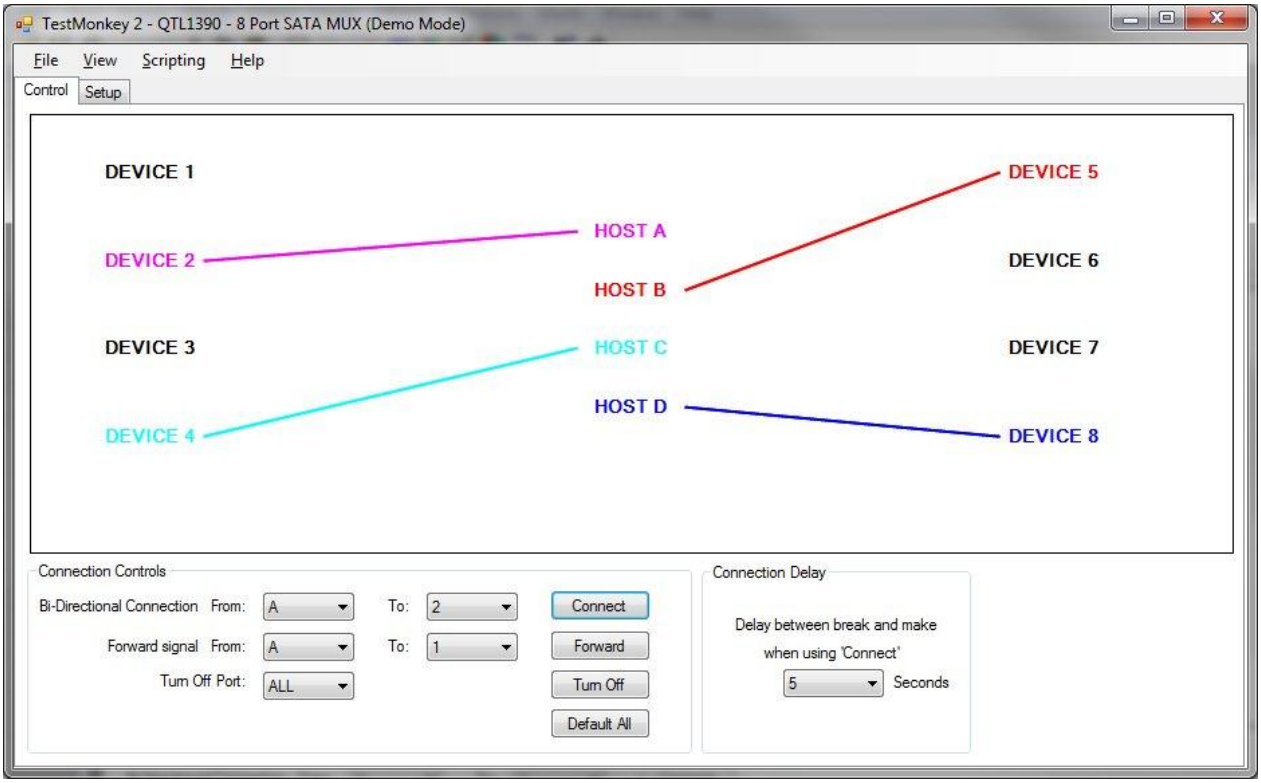
Thus, if ports **A** and **2** are connected:

**1.** Break the link **A**->**2** and the link **2**->**A**
2. Wait for 5 seconds
3. Create the link **A**->**2** and the link **2**->**A** (at exactly the same time)

The disconnect/delay/reconnect process is required to ensure that the host and device respond correctly to the change. Changing the ***Connection Delay*** parameter will alter the time taken before the reconnection. If this is set to 0 then the entire process will be instant.

## Port Forwarding

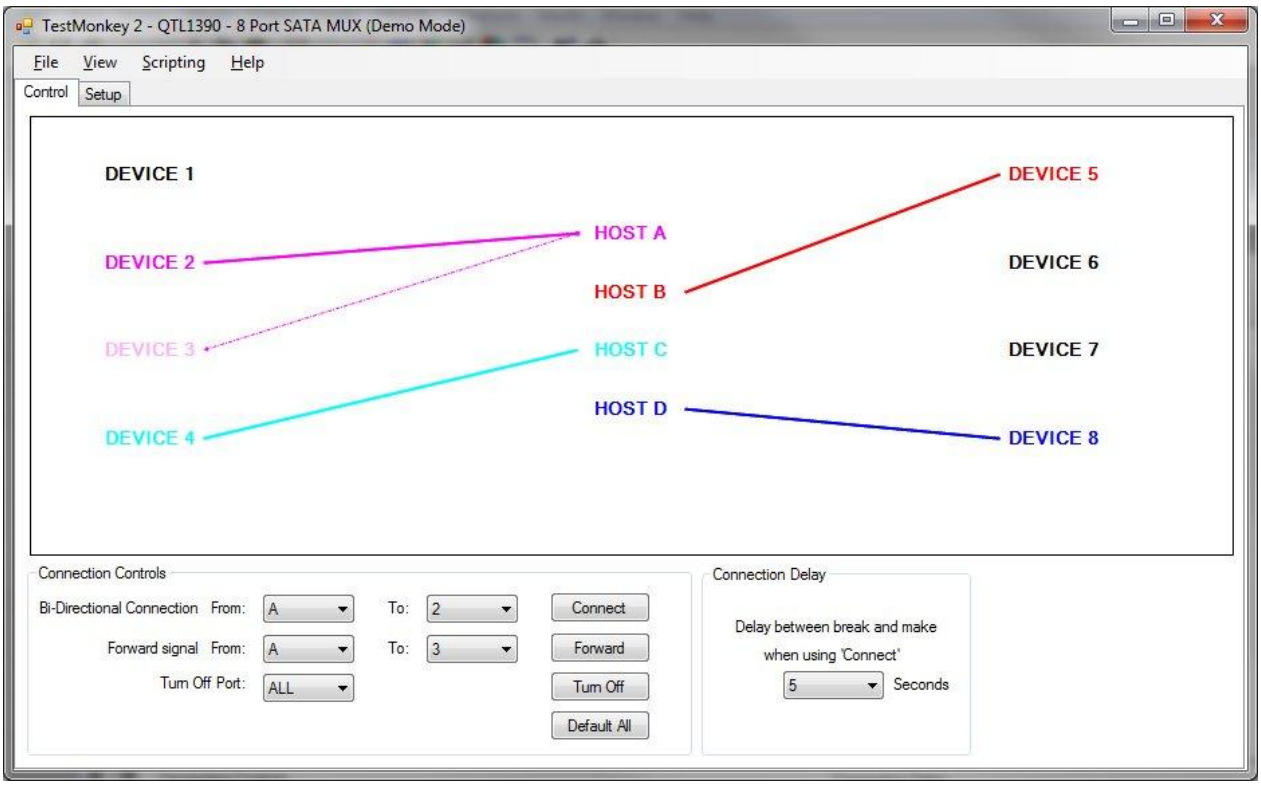
A single direction link can be created, to forward data to a second port. A Mux port can be forwarded to any number of other ports




***NOTE: The USB 3.0 MUX can forward ONLY the USB 3.0 signals. USB 2.0 signals are passively switched and cannot be forwarded to a second port.***

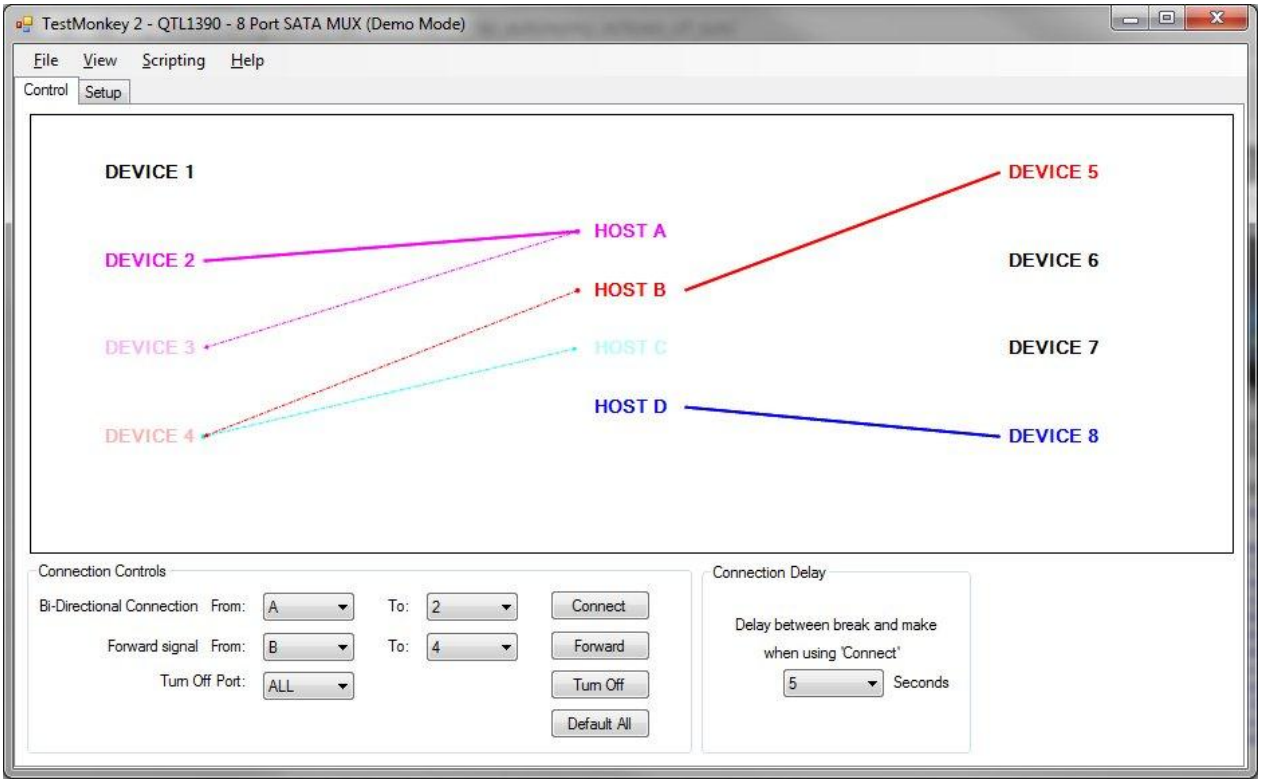
1. In the Forward section, select the port that will supply the data in the ***From*** box.
2. Select the port that will receive the forwarded data in the ***To*** box
3. Click ***Forward***

If we Forward from ***A*** to ***3***, then data received into Host A will be output at Device 3. This could allow an analyzer to be attached without altering the data path:




The dashed line indicates the forwarding link. The arrow head also shows the direction of the link.

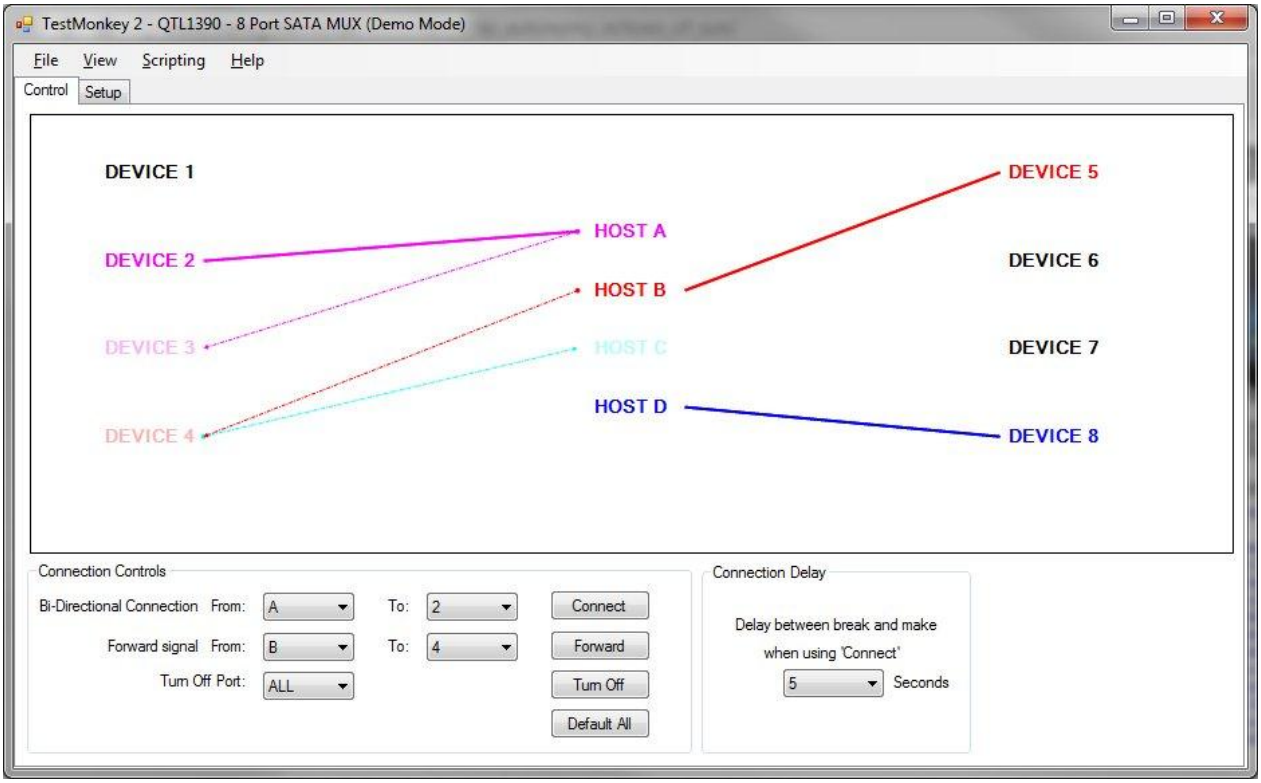
While a port can forward data to any number of other ports, it is only possible to receive data from a single port. If we now forward **B** to **4** then this will override the existing link from **C** to **4**.



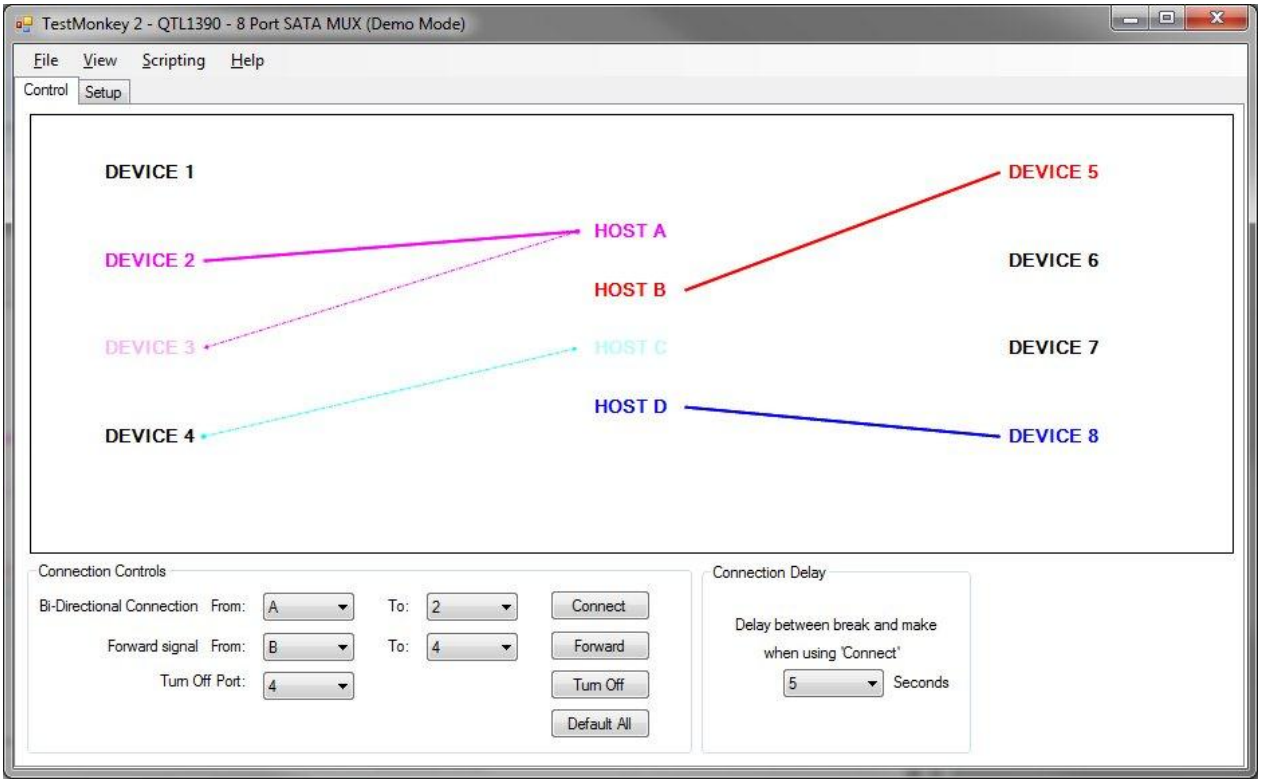

Unlike the 'Connect' command, the 'Forward' command does not automatically remove any related links. In this case the link from **4** to **C** has been left intact

## Port Off

A port that is 'off' will not transmit or receive data. The port LED will be off in this case (As with ports 1, 7 and 7 below)




If we use the ***Turn Off*** port command we will remove the link that goes TO that port. If we turn off Port 4, we will remove the link from **B->4:**

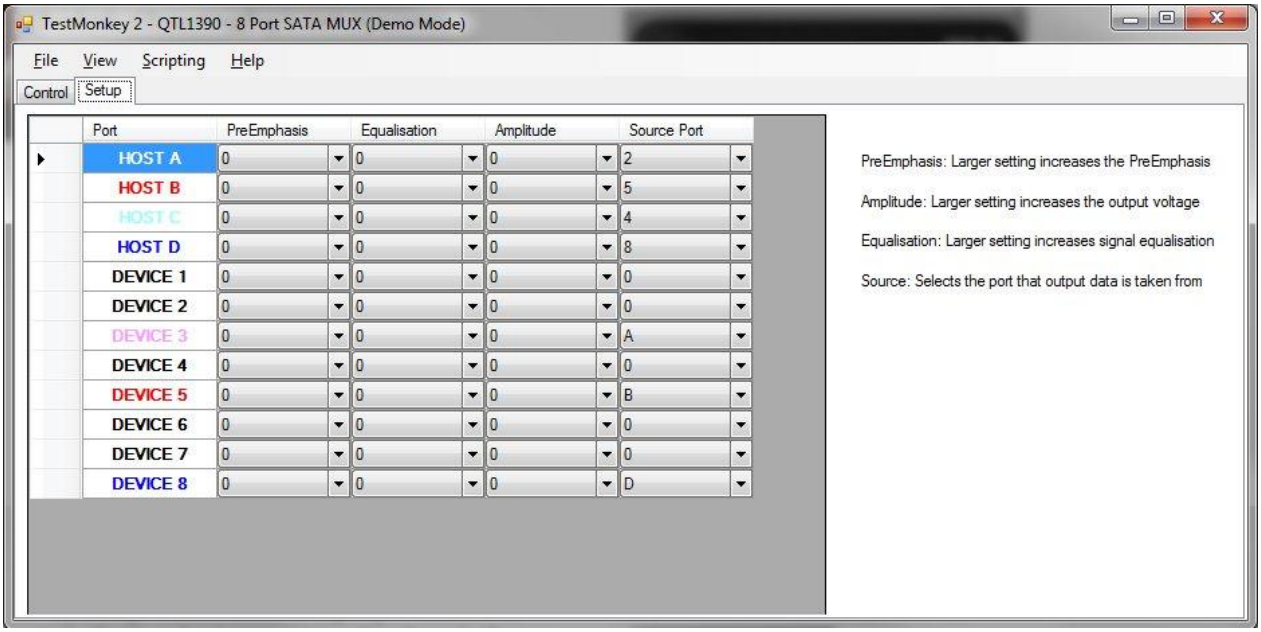



We can also turn off '***ALL***' ports, this will remove ALL connections.

## Mux Modules Config

Mux ports can be configured separately to alter their output parameters.

***NOTE: On the USB 3.0 Mux, only the USB 3.0 signals are affected my this, USB 2.0 signals are passively routed and so not re-driven***




On the Mux setup page, you can select the value to use for each port. See the technical manual for the module for details on the affects of these settings.

***NOTE: The amplitude setting changes the output voltage of the port. Ensure this is within the acceptable range of your devices before connection!***

# Array Controllers

TestMonkey 2 is designed to control single modules. Normally these modules are directly connected to the host PC via USB Serial or Telnet.

Array Controllers are designed to attach multiple modules to a host PC, normally for automated control by PERL scripts or similar. Sometimes though, you may want to use TestMonkey to control a module that is connected via an Array Controller.

With v2.1.0 of TestMonkey, this is now possible

## CRITICAL NOTES

- TestMonkey requires to sync fully with the module it controls. This process is MUCH slower through an array controller that it is with a directly connected module. Complex modules, such as QTL1069 - SBB Canister Module may take up to 60 seconds to sync fully.

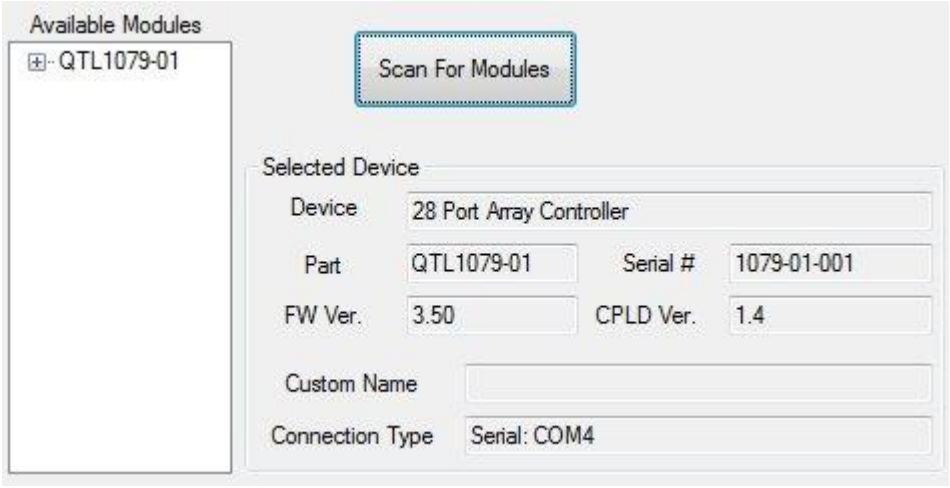
- If TestMonkey is controlling a module on an Array Controller, it will hold the connection to the Array Controller open. This means that no other connection can be made to it. Example: If you connect to a module on port 1 of the Array Controller, none of the other modules can be controlled by any other means.

- If you are scanning on USB; USB capable modules will ALSO show up as directly connected modules as the Array Controllers have internal USB hubs. In this case you could connect to the module using direct USB or connect it as a sub-module of the Array.

## Connection to a Module on an Array

- Connect the Array Controller to the PC.
- Select the connection type USB/Serial/Telnet as described in getting started
- Scan for modules

You should now see the Array Controller in the ***Attached Modules*** list. In this case, a 28 port controller is attached on serial




- Click the '**+**' icon next to the Controller to show the attached Modules. This may take a few seconds to populate
- To refresh the list of modules, click the '**-**' icon to close the controller, then open it again as in the last step

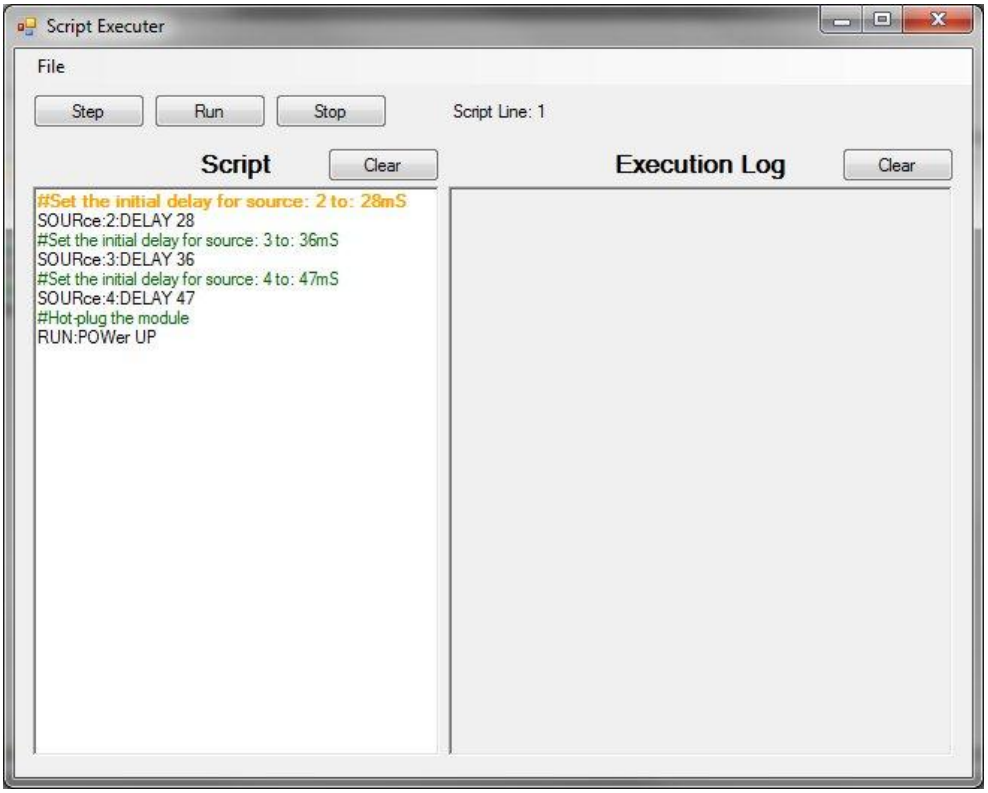

We can see that 2 QTL1253 (mini SAS modules) are connecte.  The selected one is connected to Port 1 of the Array Controller. The Custom Name (See ***Custom Module Names*** help) that we gave it also helps to identify the one we have selected.

- Click ***Connect*** to open the module as normal

---

# Exporting Scripts

- Select ***View->Script Executor*** to open the logging dialog




Any actions you have recently performed will be visible on the left hand 'Script' box. If you continue to perform any actions on the main TestMonkey control window, these actions will be appended to the current list. You can also type directly into the window to edit or extend the script.

***File->Save*** will allow you to export the script to a text file for use later.
***File->Load*** will allow you to load an existing script (this will replace any currently displayed script)
The ***Clear*** buttons clear the script and execution logs respectively.

- Running a Script

The line in yellow is the line that will be run next. ***Double click*** on a line to select a different execution line.

Press the ***Step*** button to execute the current script line. The ***Execution Log*** will display both the command run, and any response from it. If you run a line that causes a change to the state of the module, the 'sync' window will appear briefly. As the script lines are run, the main Monkey window will track the current state of the module.

Press the ***Run*** button to execute the entire script from the currently selected line. While running, press ***Stop*** to halt the execution. When a script is running , you will not be able to make any changes on the main TestMonkey window.

- Errors

Any errors that occur when a line is run will appear in the ***Execution Log***. The most common errors are due to an invalid command if you have entered your own. You will also get an error if you try to set a value that is out of range or if you try to 'plug' the module when it is already plugged!

# Custom Module Names

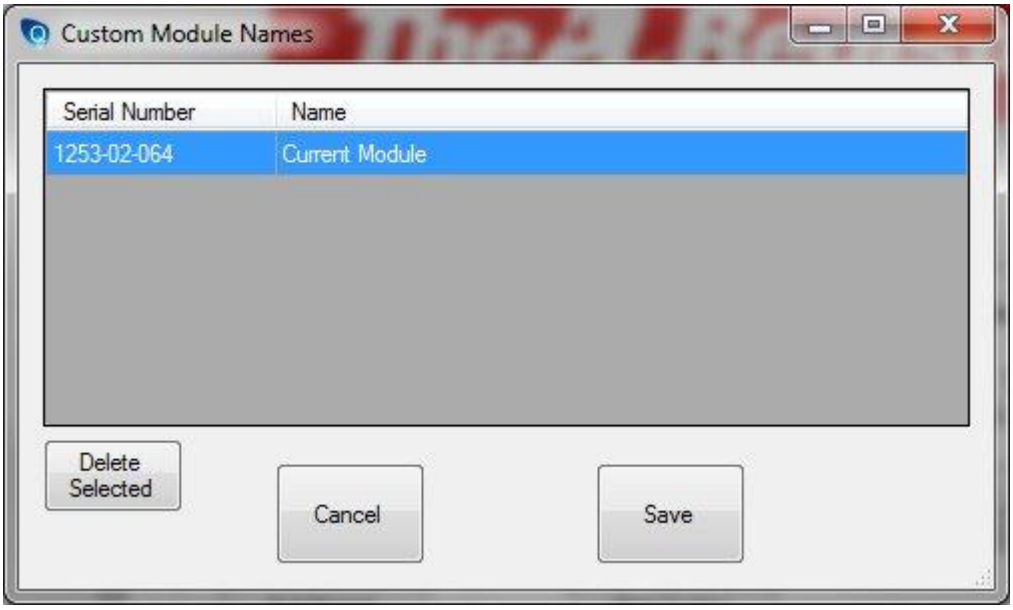
You can set a custom name for each module you use. Each module has a serial number, this is used to recognise when the same module is attached.

Custom module names can help you control your test system more easily.

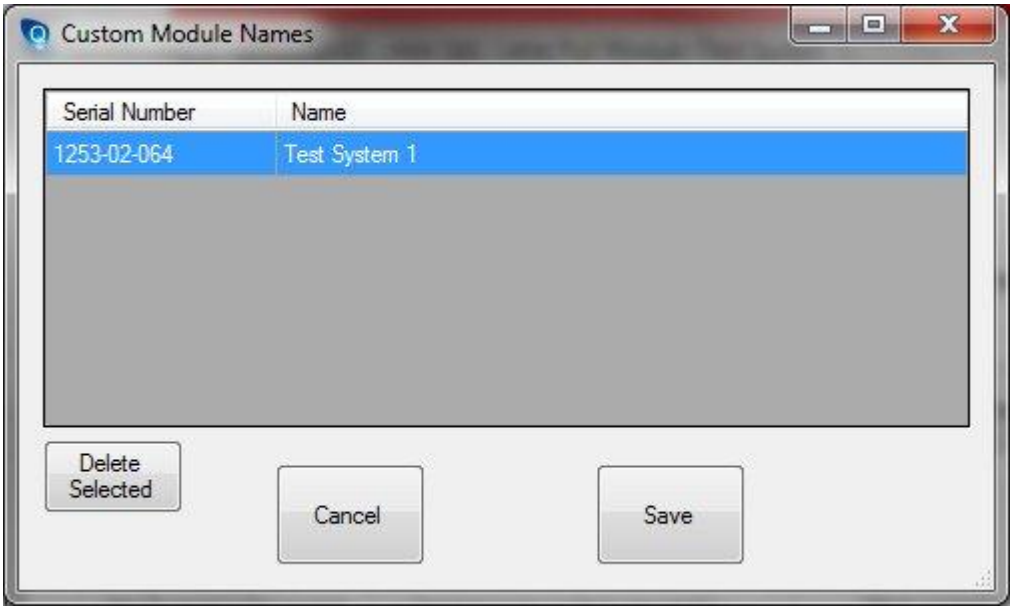
Connect a module in TestMonkey 2

Goto ***Tools->Custom Module Names***

Your current module will be shown in the list with a default name:




Double click on the ***name*** and edit the text as required:




Select ***Save*** to confirm and close the dialog. If you have named multiple modules, they will all be shown in this list.

The module name will now appear on the Application title bar. When you come to connect the module again, you will see the Custom Name in the module details:


Available Modules
QTL1253-02
Scan For Modules
Selected Device
Device
mini SAS Cable Pull Module
Part
QTL1253-02
Serial #
1253-02-064
FW Ver.
4.002
CPLD Ver.
1.3
Custom Name
Test System 1
Connection Type
USB

# Controlling Multiple Modules

New with TestMonkey 2.1.x is the ability to control more than one module at the same time. This is NOT a normal mode of operation for TestMonkey but it may be easier than running 2 instances of the application to run 2 different modules.

In addition, if you want to control more than one module at a time on an ***Array Controller***, you cannot do it with separate instances of the application as you can only open a single connection to the controller. You would have to use this procedure.

- On the ***Select Device*** screen, select ***Advanced->Open Multiple Modules***
- You can now select a module as normal and click ***Connect***
- The module screen will open as normal but the Select screen wil also remain open
- Select the next module you wish to open and repeat
- When you are done, select ***Done*** to hide the connection window

You will now have multiple module control windows open at the same time and you can operate them as normal. Note that if you perform a long action on one module (such as a Sync command), you must wait for it to complete before further commands can be issued to any other module.

**CRITICAL NOTE**
- If you wish to open multiple modules on an Array Controller, you MUST select them all in sequence without performing any other module scans. If you open one module on the Array then move to a different tab or re-scan for modules, you will find that you cannot access other modules within that array.

This is due to the architecture of TestMonkey which was not originally intended to control multiple modules in this way.

# Programmable Power Module

## Important Notes!

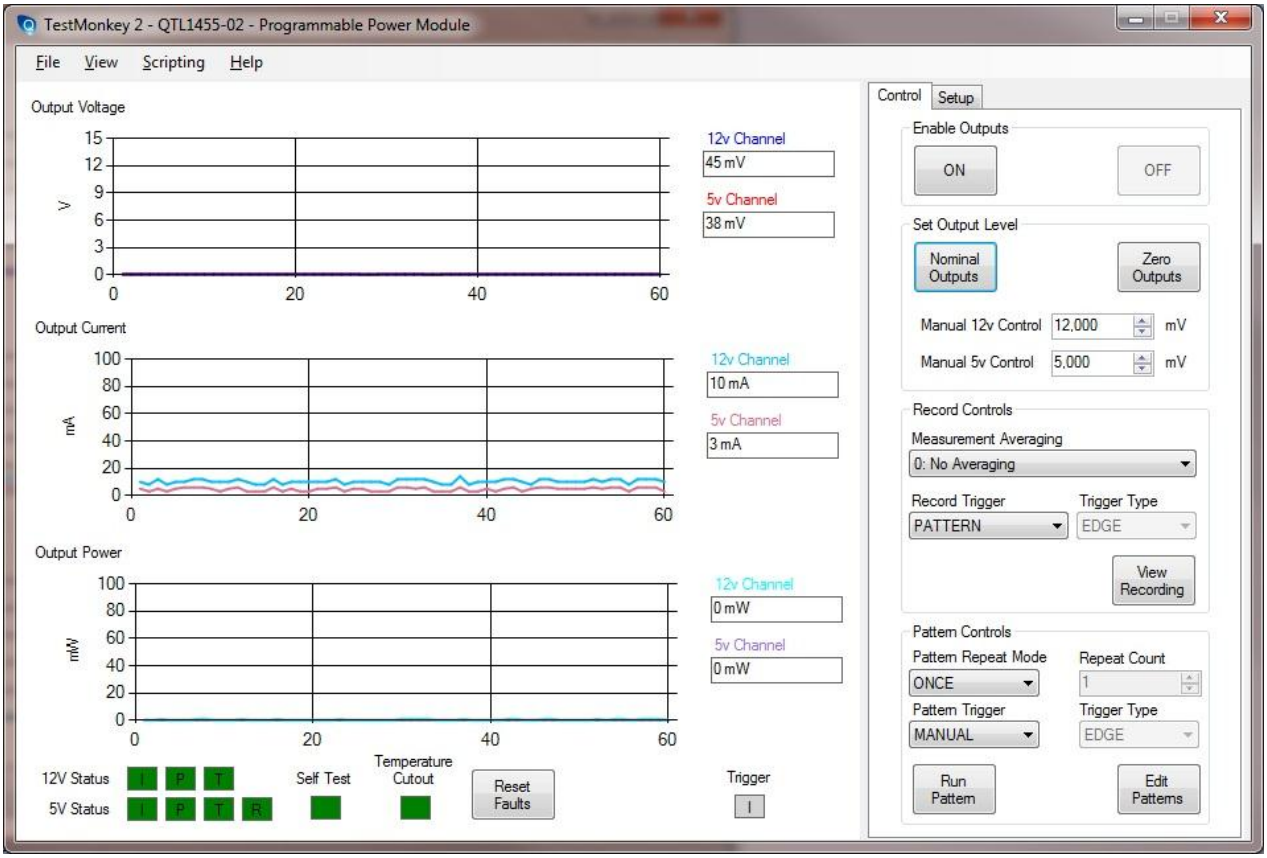
- This version of the PPM control software is designed to work with a module connected on USB. Recording functions will not work when the module is connected on Serial.

- Images are taken from a module with the External Triggering option. On basic modules, these triggering options will not be available and will be greyed out.

- Some functions will require updated Firmware code on the power module. At the time of writing v4.101 was the latest version.

## Main Screen

When the module is first connected, the following screen will be shown

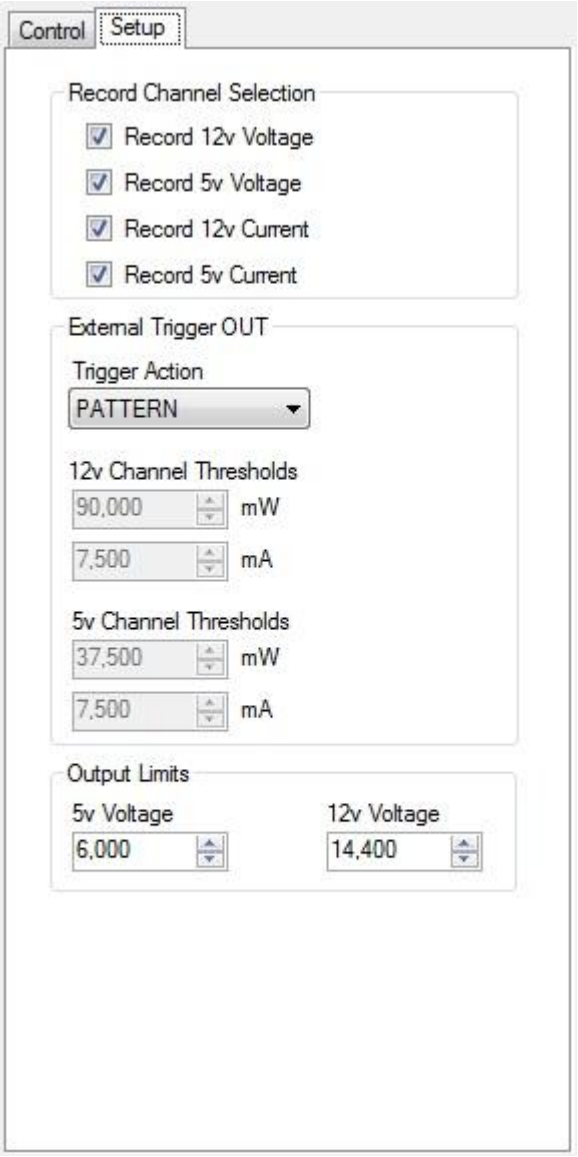



Left Side (Status Display)

- The graphs show the last 60 seconds of the measurements at the output of the module, this updates every second
- The numbers to the right of the graphs are the current measurements (updated every second)
- The status indicators at the bottom will be green, unless there is a fault
    - ***I*** - Current output exceeded the alloved maximum level
    - ***P*** - Power output exceeded the allowed maximum level
    - ***T*** - Temperature for the port is high
    - ***R*** - Regulator protection cut-out has tripped
    - ***Self Test*** - Unit self test has failed
    - ***Temperature Cutout*** - Temperature was high enough to force shuddown of the outputs
- ***Reset Faults*** will reset the unit and try to clear any faults. If faults still remain, they will be listed in a dialog box
- The ***Trigger*** indicator flashes if an external trigger caused an action to run

Right Side (Controls)

- ***Enable Outputs*** allows you to turn the main outputs ***ON*** and ***OFF***. When OFF, the outputs will always be at 0v
- The ***Output Level*** controls set the basic output voltages on the ports
    - ***Nominal Outputs*** - Sets the outputs to their 'normal' running output levels, normally 12v and 5v
    - ***Zero Outputs*** - Sets both outputs to 0 volts
    - ***Manual Controls*** - Sets the output level, changes here immediately alter the output voltages
- Record Controls
    - ***Measurement Averaging*** - Averages multiple samples together to give a more stable measurement.
        - Higher values also increase the total time that can be recorded. 8 Sample averaging increases record time by 8x
    - ***Record Trigger*** - Selects what will cause recording to start
        - ***PATTERN*** - Record will start whenever an output pattern is started
        - ***POWER*** - Record will start whenever power ***ON*** is pressed
        - ***MANUAL*** - Record will start when the user presses the ***Start Recording*** button
        - ***EXTERNAL*** - Record will start based on the external trigger
    - ***Trigger Type*** - Selects how an external trigger will start/stop a record event
        - ***EDGE*** - Recording will start on a rising edge and will end when memory is full.
        - ***LEVEL*** - Recording will start on a rising edge and continue while the trigger is high. A falling edge will stop the recording

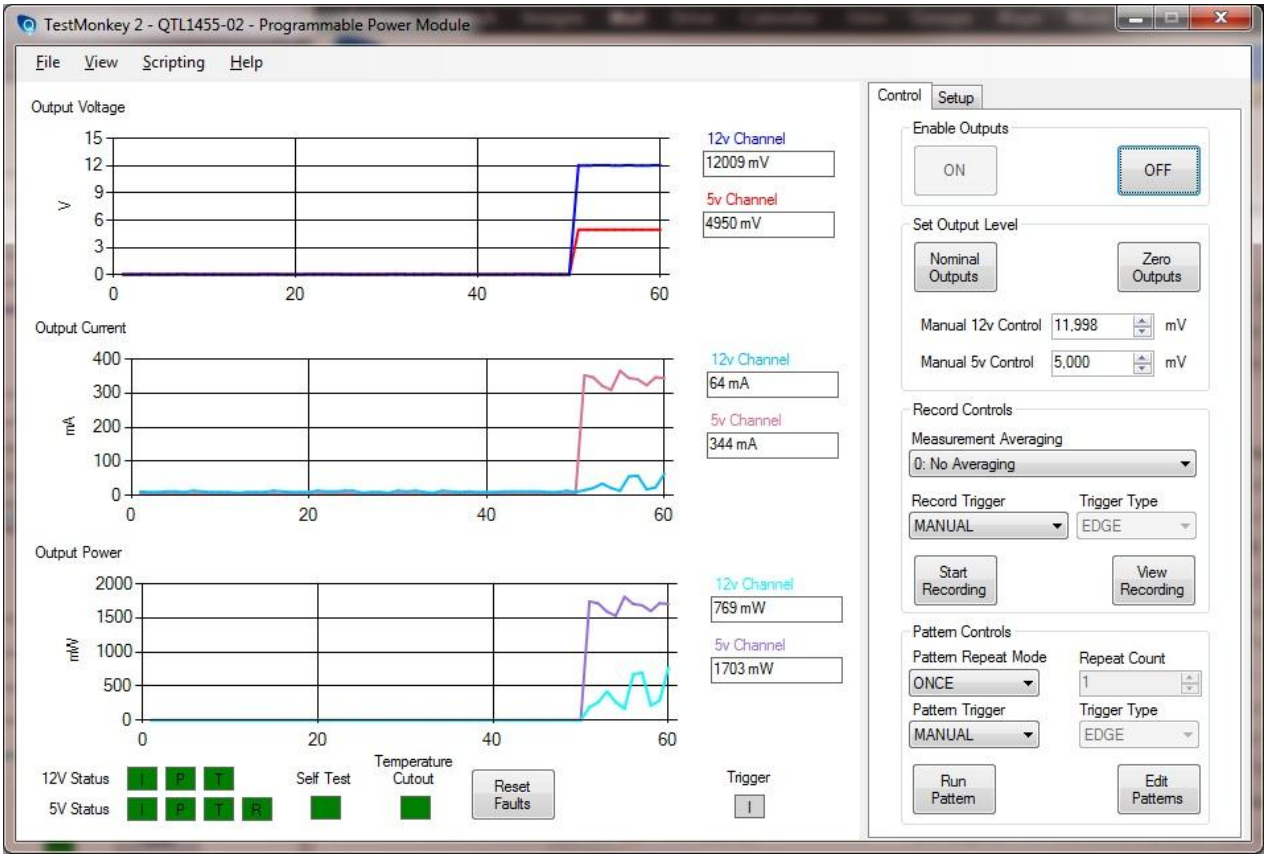



- ***Record Channel Selection*** - Selects which measurements should be stored during a record operation
    - Reducing the number of recorded items will increase the available recording time
- ***External Trigger OUT*** - Sets up the external trigger (if available)
    - ***Trigger Action*** - Sets the action that will cause a trigger out
        - ***PATTERN*** - Trigger out is high while a pattern is running
        - ***RECORD*** - Trigger out is high while recording is running
        - ***THRESHOLD*** - Trigger out is high when any of the set threshold levels are exceeded
- ***Output Limits*** - The output voltage will never be above these set levels. This can be used to protect devices that could be damaged by the 6v / 14.4v maximum levels that the unit can produce

## Power Up

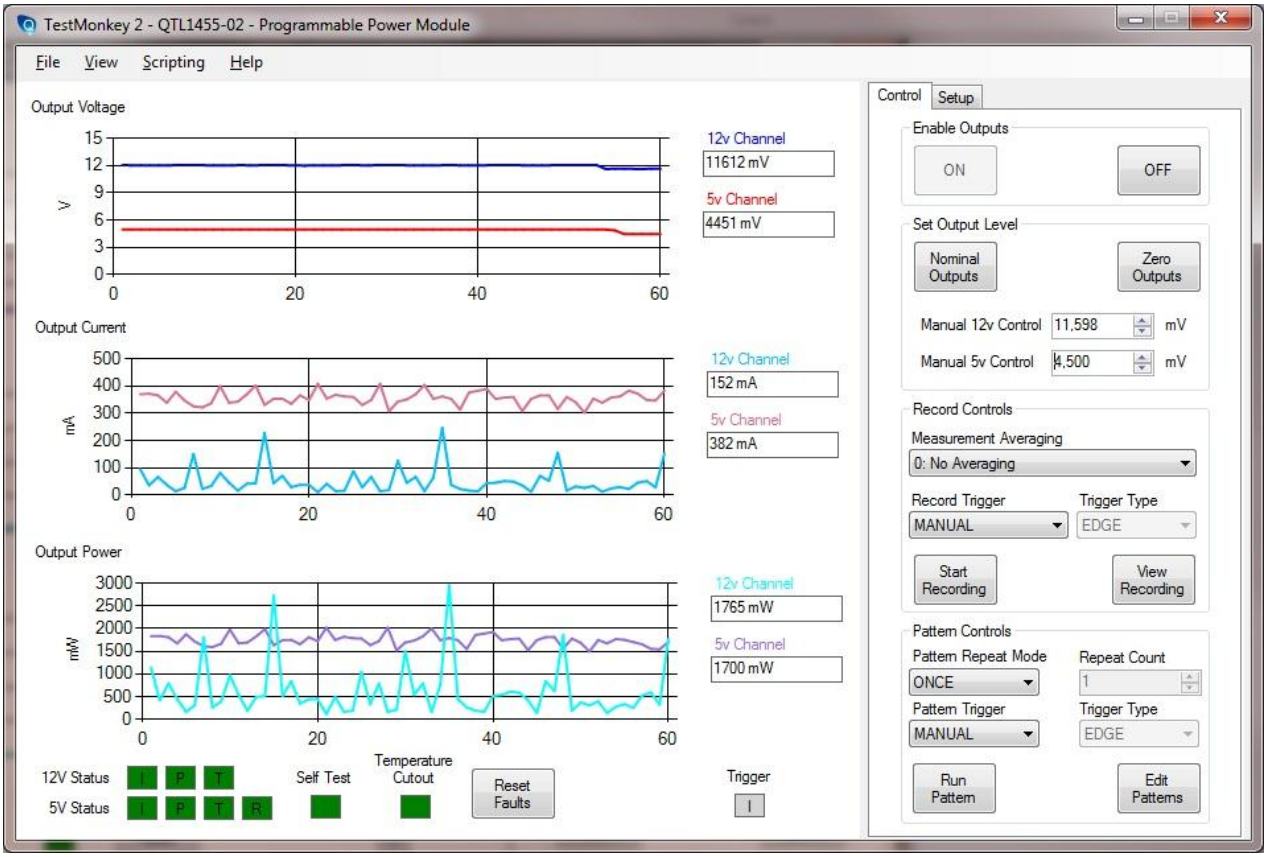
The default startup state is with the outputs disabled and the default voltage levels set to the 5v/12v nominal values.

To power up the attached device, select the ***ON*** button. The outputs will start up and output the set values.




The voltage graph shows the increase in output voltage. In this case there is a disk drive attached to the PPM but it is not set to spin up.

## Margining Output Voltage

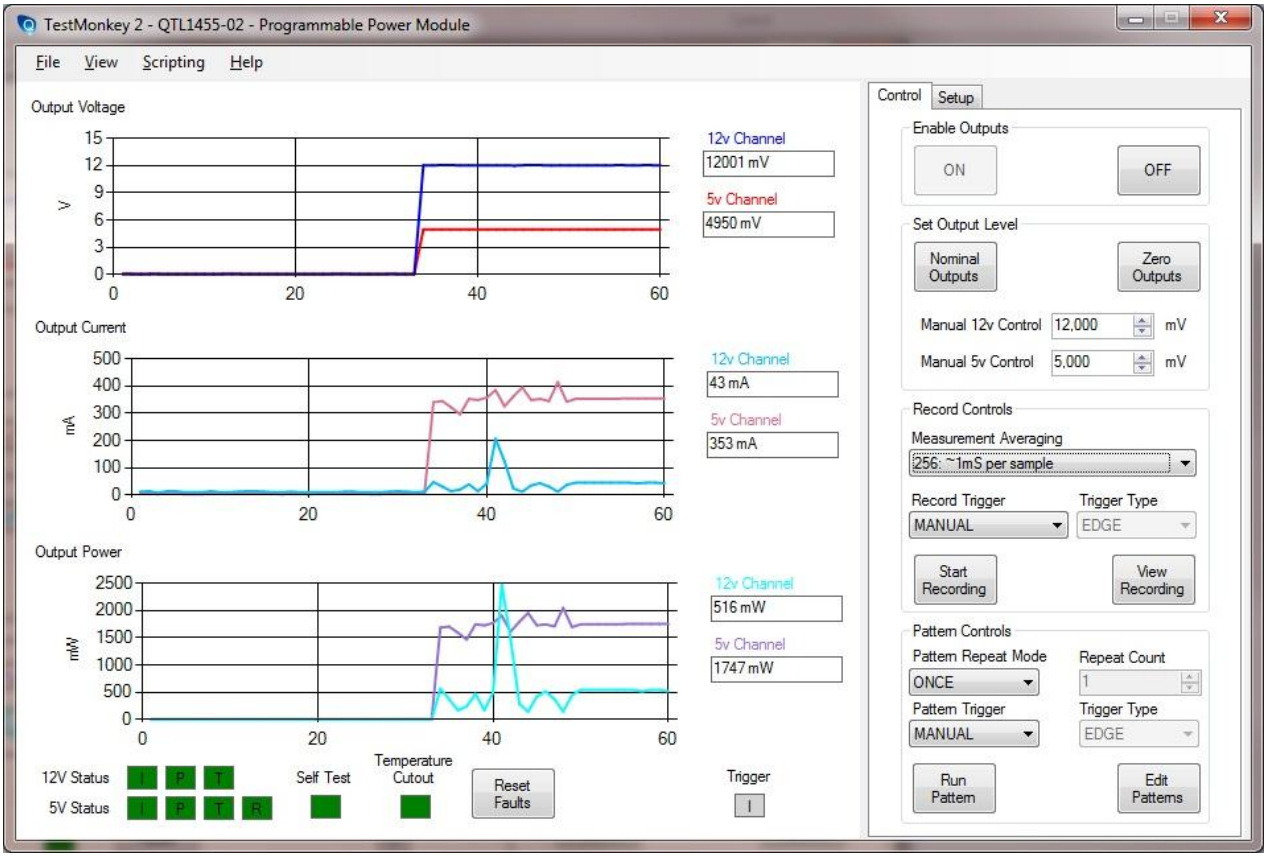



You can edit the output voltage level by editing the ***Manual Control*** boxes. Changes here will happen immediately (assuming the outputs are enabled).
On the voltage graph you can see the changes as the manual outputs were edited

Clicking the UP/DOWN arrows will alter the voltage level by 100mV

***Nominal Outputs*** will change both outputs to their nominal output values
***Zero Outputs*** will change both outputs to 0mV

## Averaging Levels




The graphs display data based on the current ***Measurement Averaging*** rate. By default there is no averaging, increasing this to average over 1mS greatly reduced the noise level that we see on the current and power graphs.

Samples are normally measured at 4uS intervals. Averaging modes allow multiple samples to be averaged together to create a single reading. In the above window, 256 samples are averaged together:

256 x 4uS = 1024uS = ~1mS

The numbers of samples that can be averaged come from the equation 2^n (2 raised to the power n). This greatly simplifies the math required in the device.

## Output Patterns

As well as allowing simple voltage margining, the module can output high speed, complex patterns.

Patterns are described as a set of points. A different pattern can be created for each of the 5v and 12v outputs

Patterns are RELATIVE to the output voltage at the time the pattern is started. This allows patterns such as square waves to be described and run easily on top on a base output level (using a -ve voltage value).

There are two main types of pattern

### 1. Transition Patterns

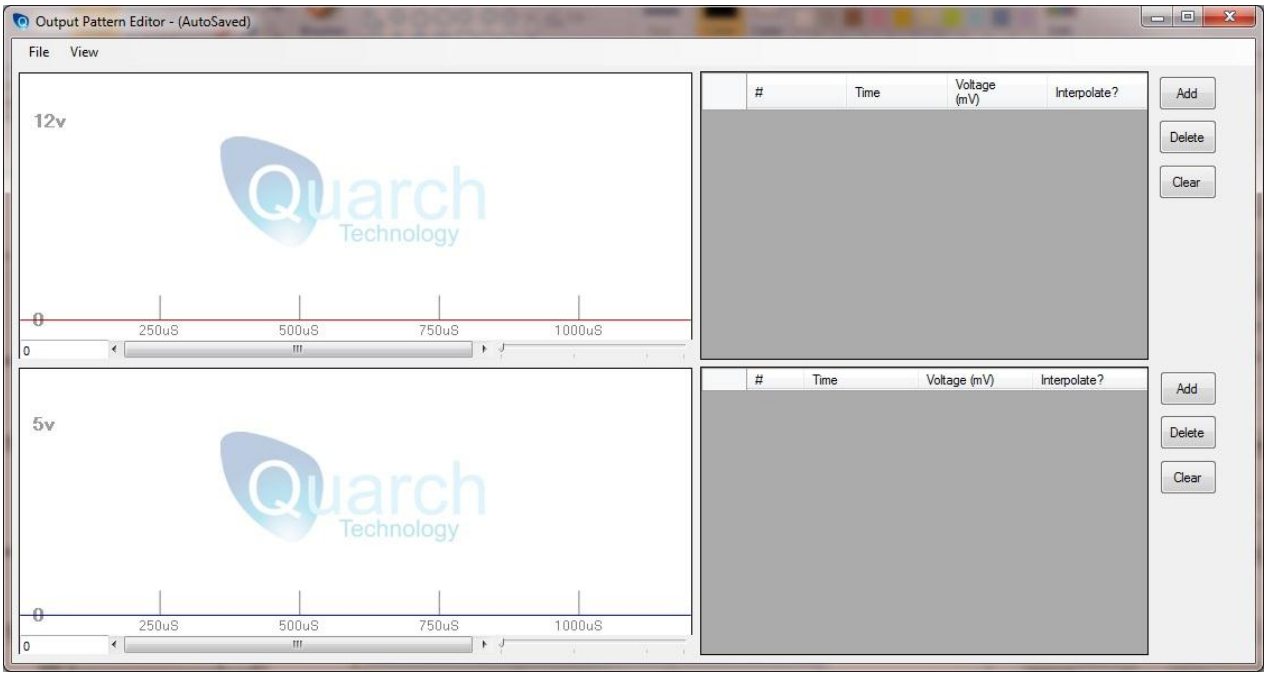
These describe a pattern whene the end votage is different from the start. This would include ramping up to a level or decreasing the output from a set level.

In this case, the pattern would normally be run once only. After being run, the output voltage will be at a new level

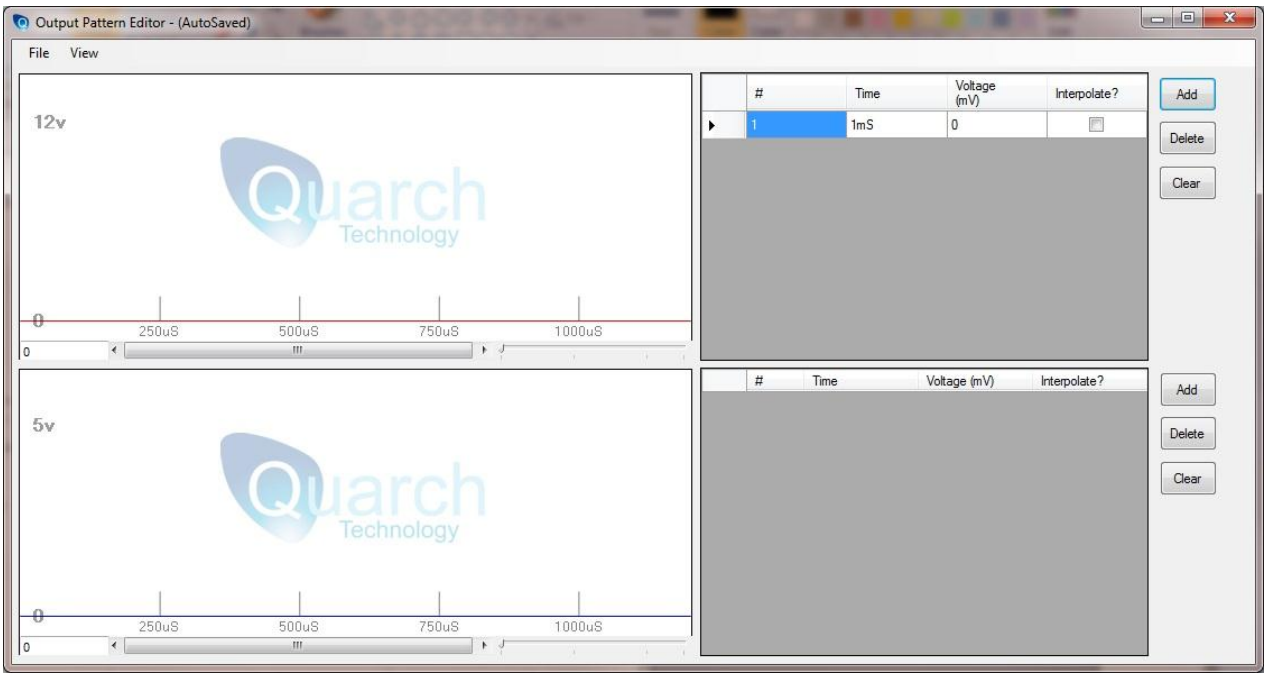
### 2. Returning Pattern

These describe patterns that start and end at the same voltage, this includes spikes, glitches and square waves.

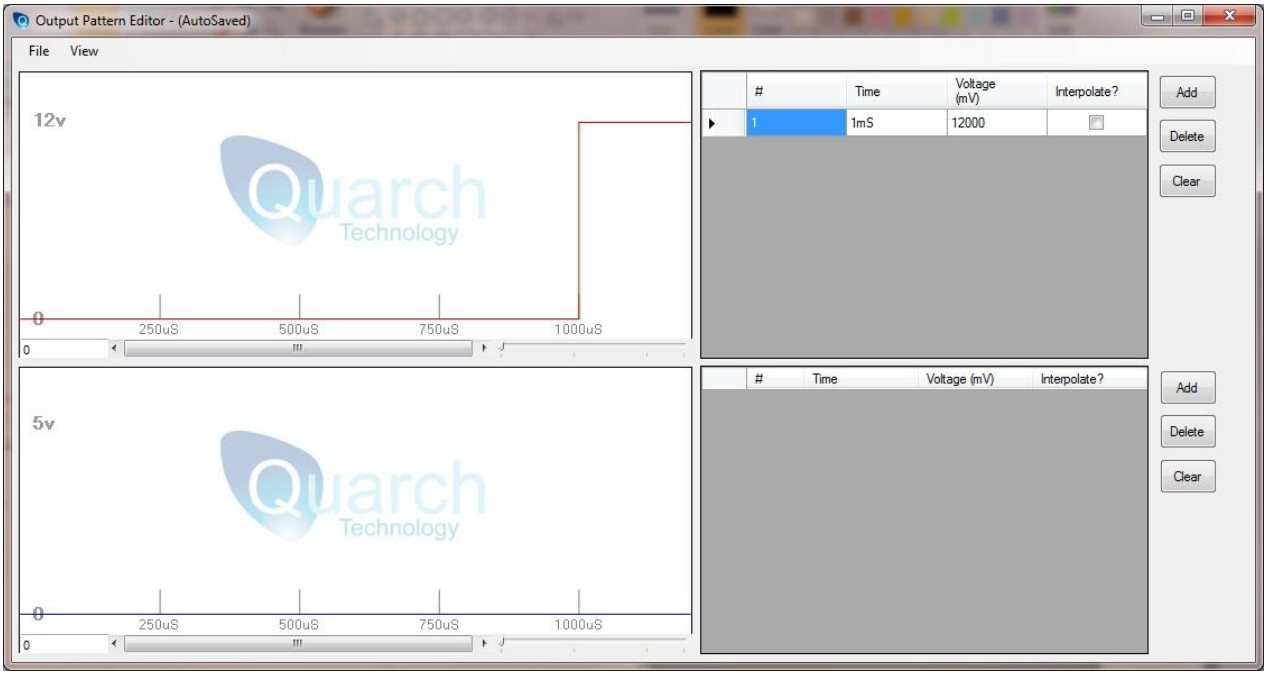
In this case the pattern can be run multiple times, after each run, the outputs will be at the same level as at the start of the pattern.



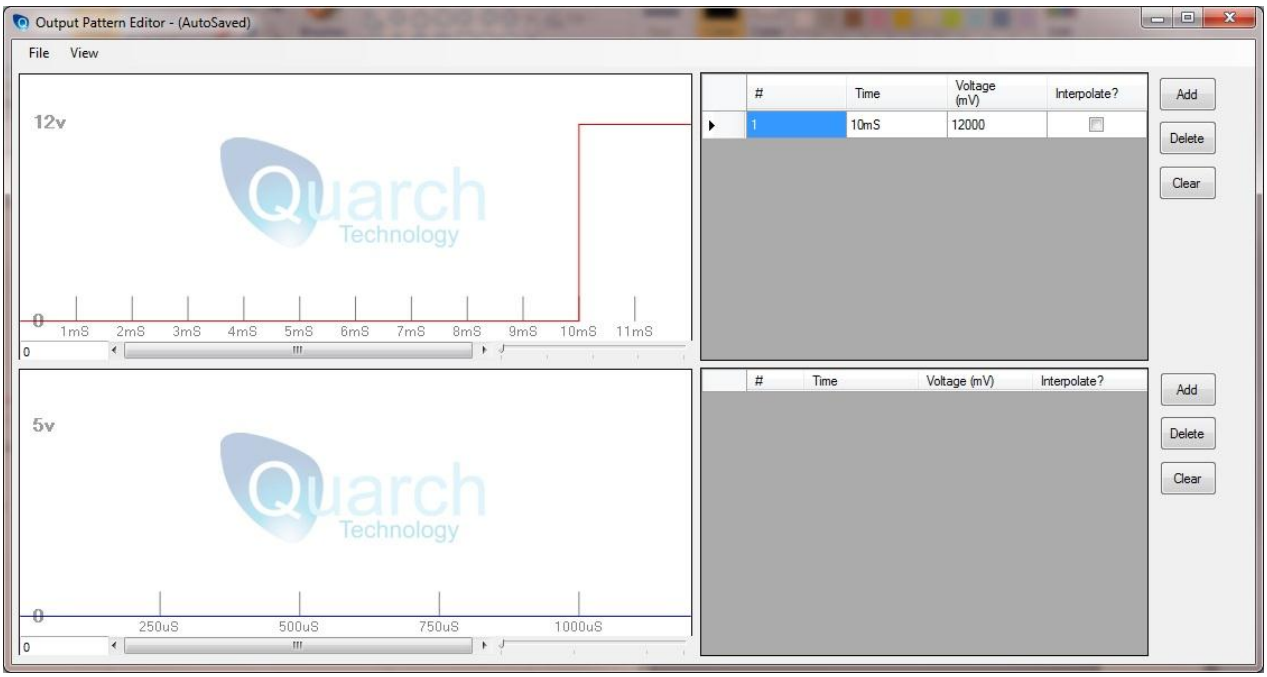

To create a pattern, first set the output voltages to the level you require at the start of the pattern. In this example we set the outputs to 0mV so we can create a ramp.
Select **Edit Output Pattern** from the main window, the editor will be displayed as above



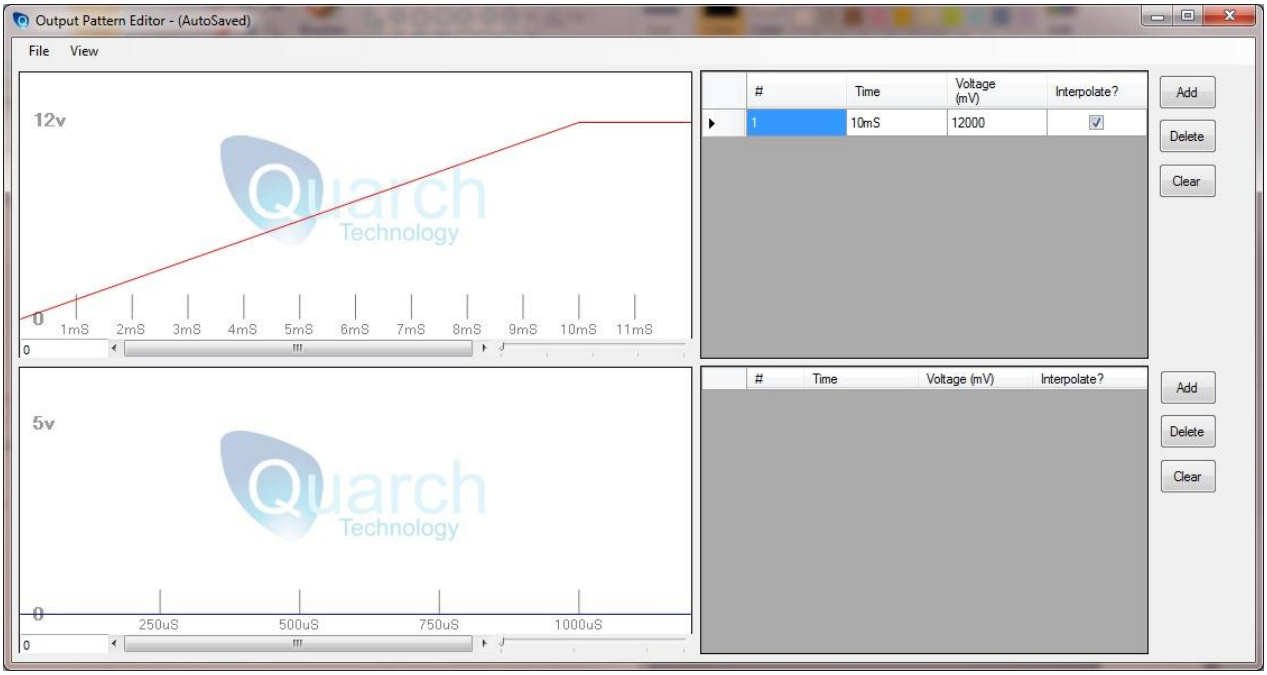

Select ***Add*** to create a new point in the pattern




Edit the point so the output will be 12v after 10mS.  The result of this action can be seen of the graph.

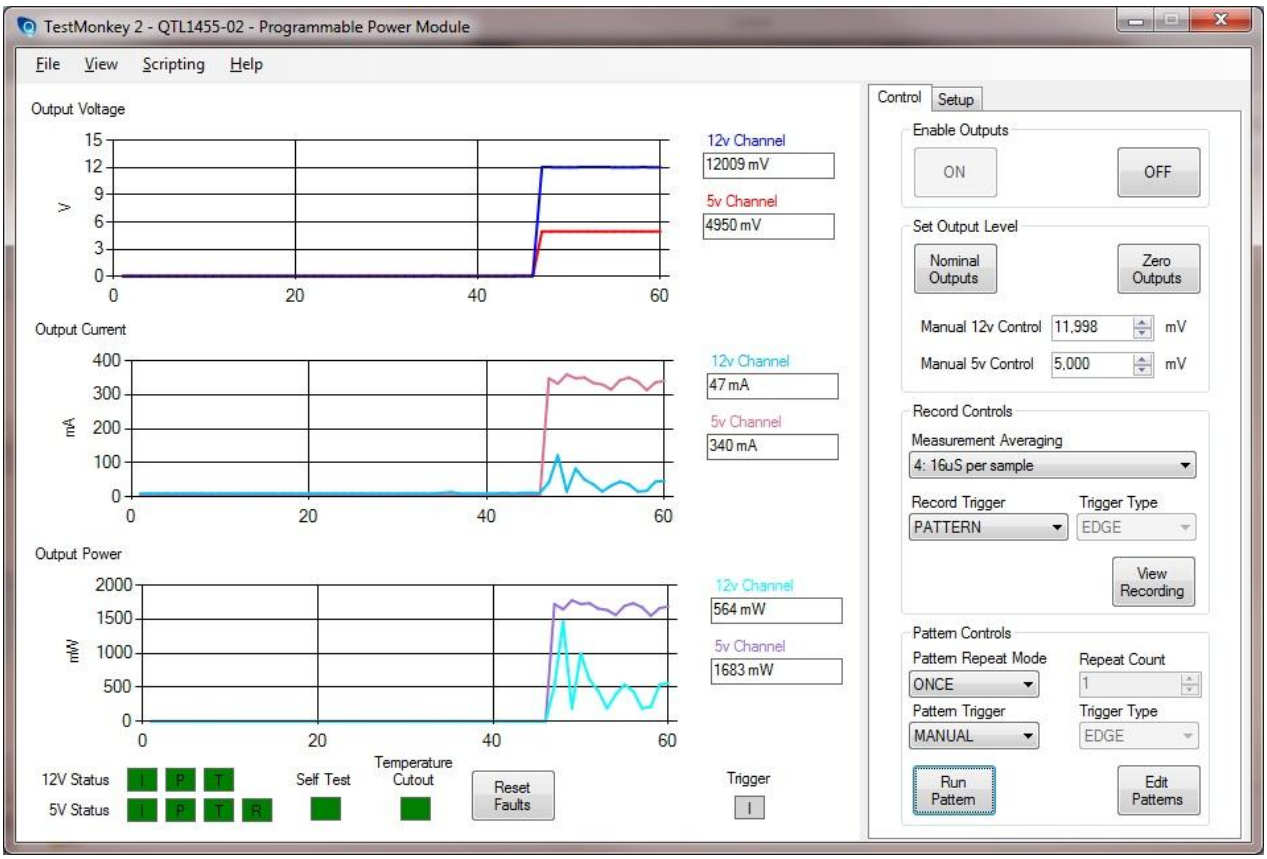



We want a ramp rather than a step, so select the ***Interpolate*** option so the output moves steadily up to the selected value over 10mS




Now repeat this with the 5v pattern so it ramps up to 5v over the same time
When this is done, you can close the window

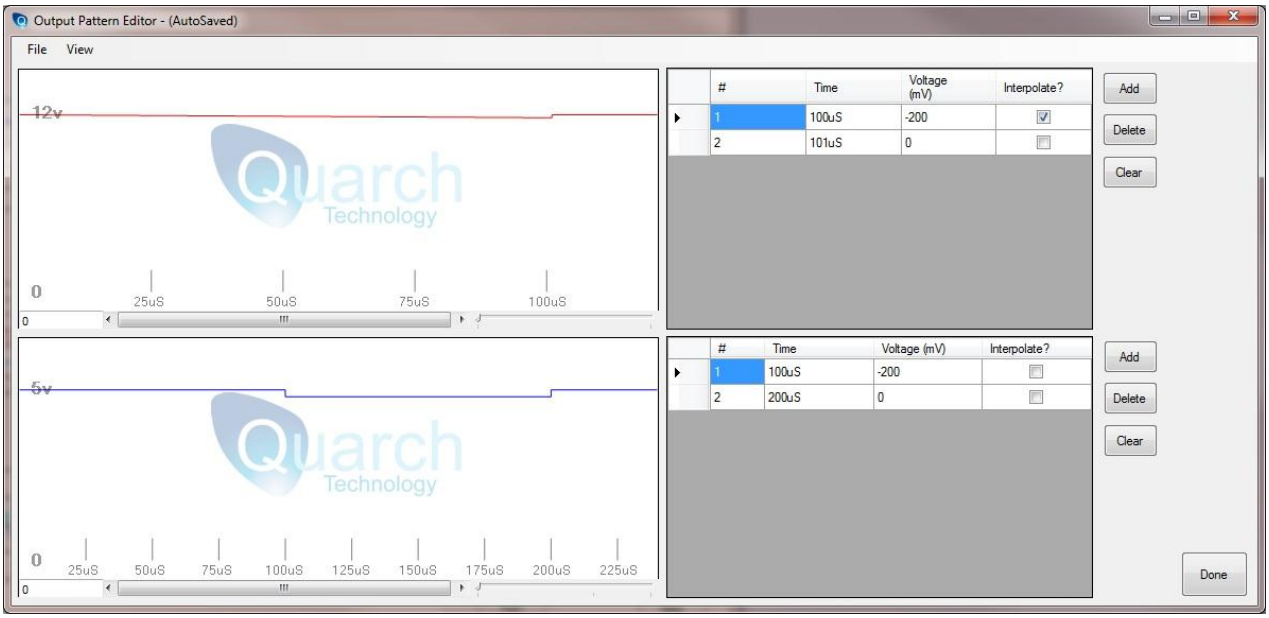
On the main screen, select **Run Pattern**. This will start the pattern which will take 10mS to complete. The outputs will be recorded automatically as the current trigger mode for recording is ***PATTERN***.



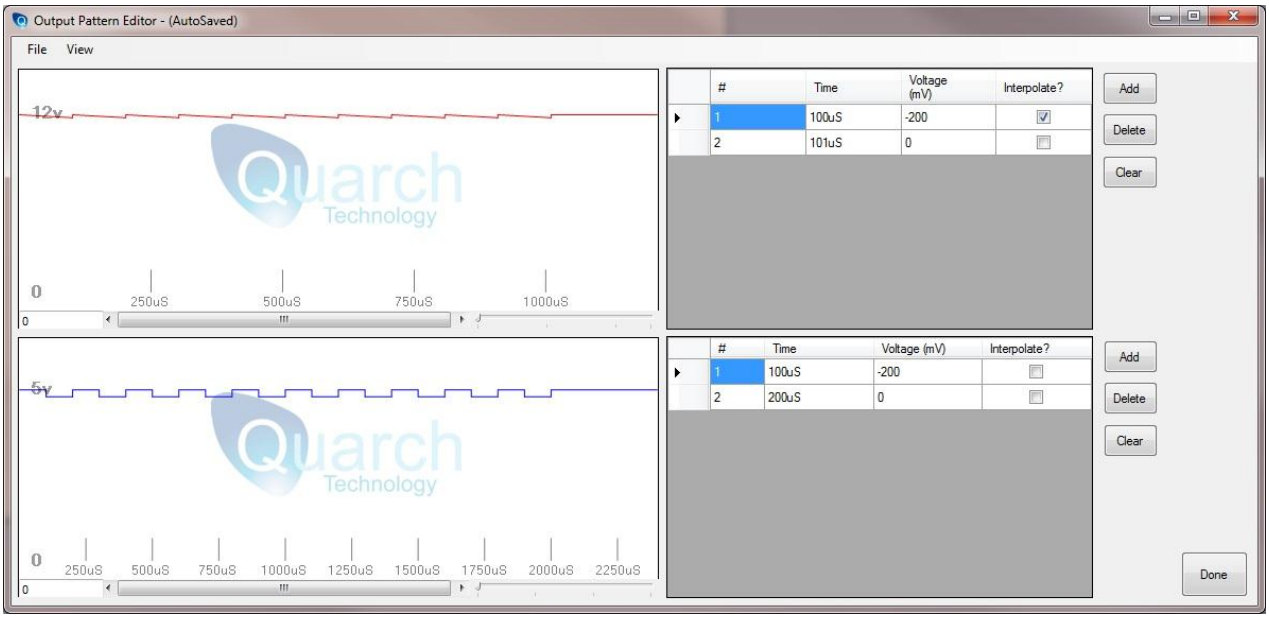

The effect of the pattern will be to ramp up the voltages, we can see this on the graphs as the attached device powers up. In this case, the pattern is set to ***ONCE***. This is a transition pattern, so running it a second time would result in the ouput levels hitting the maximum output limits.

## Repeating a Pattern

Repeated patterns are very useful for adding noise to a power rail. Here we see a simple pattern on the 12v and 5v channel:



On the main control screen, we can set the ***Pattern Repeat Mode*** to ***COUNT*** and the. This means that the count value to 10. This will mean the pattern is repeated 10 times when ***Pattern Run*** is pressed.




When the pattern repeat controls are altered, we see the effect this will have on the pattern editor. In this case, the 12v pattern is a sawtooth shape while the 5v pattern is a square wave.

The pattern editor only displays up to 20 repetitions of a pattern, If you request more than this (or a constant pattern cycle), only the first 20 will be shown.

# Trigger Modes

Two types of Triggering can be implemented in power modules:

1. Internal Triggering - For example, where recording can be set to begin when a pattern is run

2. External Triggering - External IN/OUT triggers are fitted to some modules and allow triggering from an external source

## Record Triggers

The module supports sever trigger modes that cause data to be recorded. When a trigger occurs, data will begin recording. The duration of the record depends mainly on the ***Measurement Averaging*** selection. If many samples are averaged to give a single recorded reading then the record memory will take longer to fill.

The memory can hold 65535 measurements, arranged into stripes, where a stripe contains current and voltage readings for both 5v and 12v channels (4 measurements per stripe). Each stripe of measurements is taken at ***exactly*** the same time.

If 2 channels are disabled for record, the number of stripes that can be held would double. (Each stripe would now have only 2 measurements).

The unit can read back the recorded data while the record function is still running. You will simple get a truncated data set up to the point that has been recorded so far.

***- Example***

With all record channels enabled and with no averaging, the time of each record will be 4uS:

65535 / 4 channels = 16382 stripes
16382 * 4uS = 65535uS = 65.535 mS

In this case the record will complete in around 65mS

If we increase the averaging level to ***1k*** (1024 samples)
16382 * 4uS * 1000 = ~65 Seconds

Now it will take over 1 minute to complete the record cycle

### *Trigger on PATTERN*

This is the default trigger mode and causes data to be recorded each time a pattern is started by pressing the ***Run Pattern*** button. This mode is used to capture the behaviour of the attached device when given an output voltage pattern such as a rising ramp or a power glitch.

The record action will also be happen if the pattern was started by an External trigger, rather that the ***Run Pattern*** button

### *Trigger on POWER*

This mode will trigger data record to start when the Ouput Enable **ON** button is pressed. This is used to measure the response of the attached device when initially powered on

### *Trigger on MANUAL*

This mode allows recording to be started at any time. When in manual trigger mode, a ***Manual Trigger*** button will be shown. This button will start the data recording.

### *Trigger on EXTERNAL*

In this mode, an external trigger will start the recording. This allows you to record a very specific section of time, perhaps using the output of an Analyzer to sync with a certain read/write action

Using an ***EDGE*** type trigger will start recording on any rising edge (a second rising edge will restart the recording). The recording will stop when memory is full
Using a ***LEVEL*** type trigger will start recording on any rising edge and will stop recording as soon as the trigger goes low again.

## Pattern Triggers

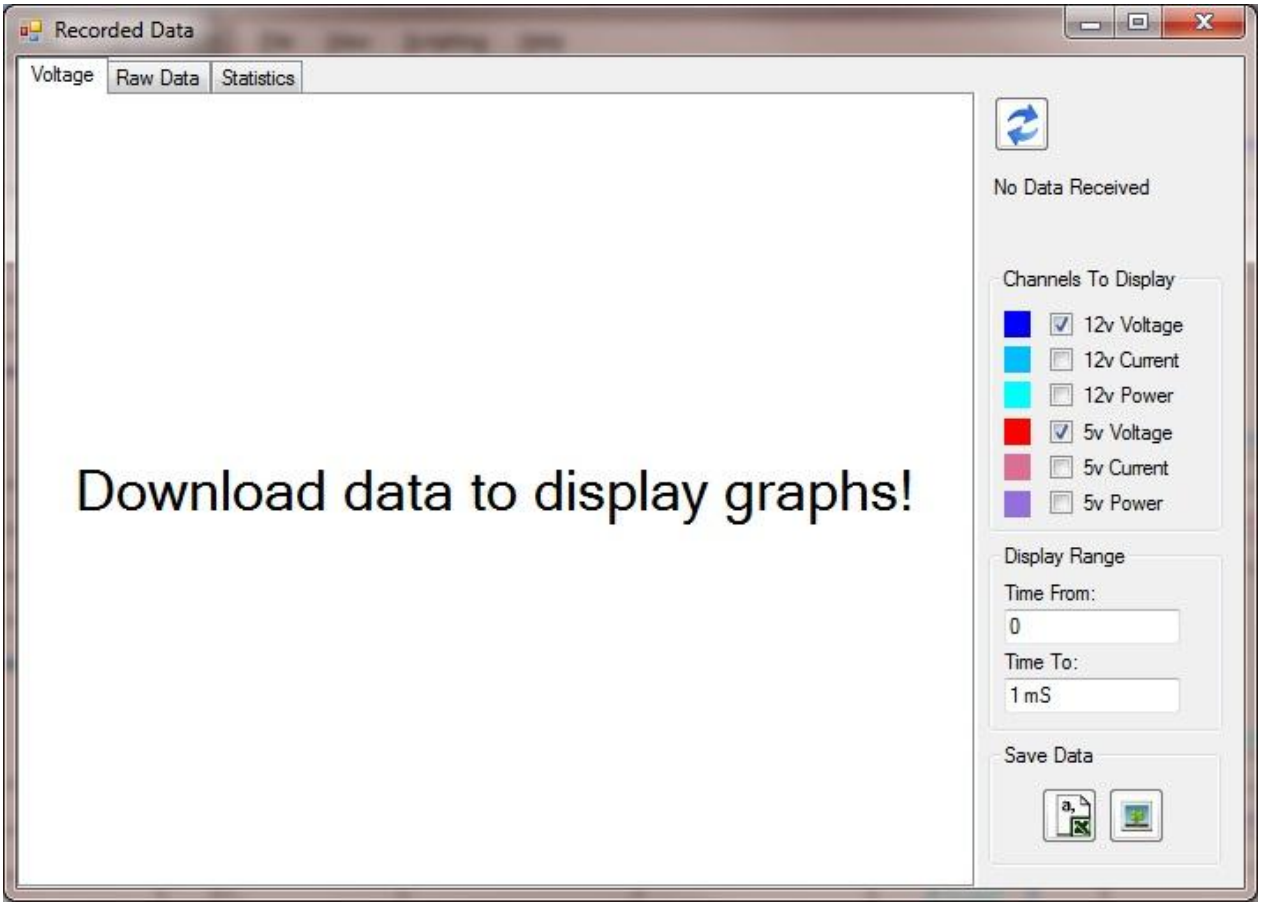
Patterns can be triggered by an external trigger as well as by the ***Run Pattern*** button

Using an ***EDGE*** type trigger will start the pattern each rising edge.
Using a ***LEVEL*** type trigger will start the pattern on a rising edge and stop it on the falling edge (or when the pattern is done, if this is sooner)

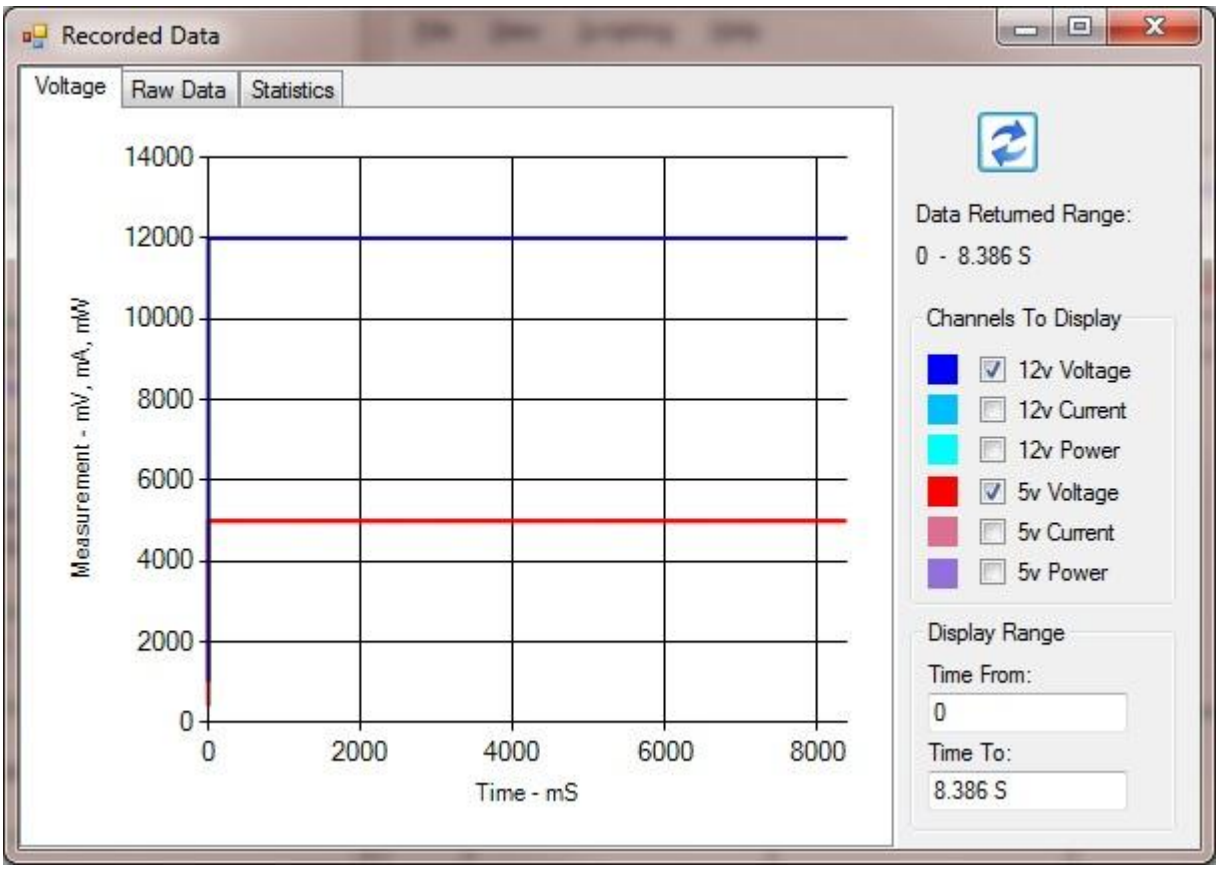
## View Recorded Data

After recording data (from a recorded pattern or manual / external record trigger), the module will have data to return.

On the main screen select the ***View Recording*** button

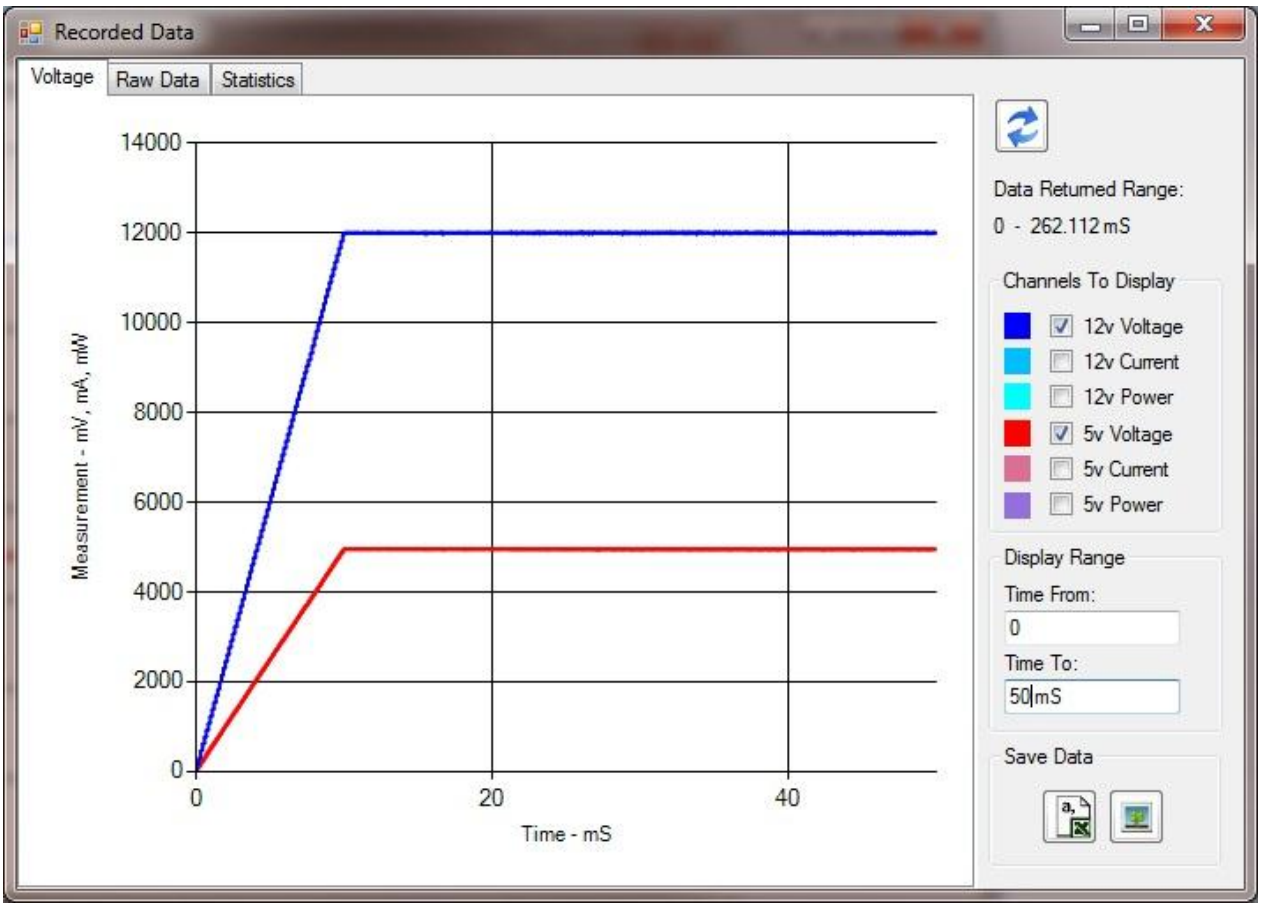



Select the refresh button to download the current data set, this will take a couple of seconds over USB.

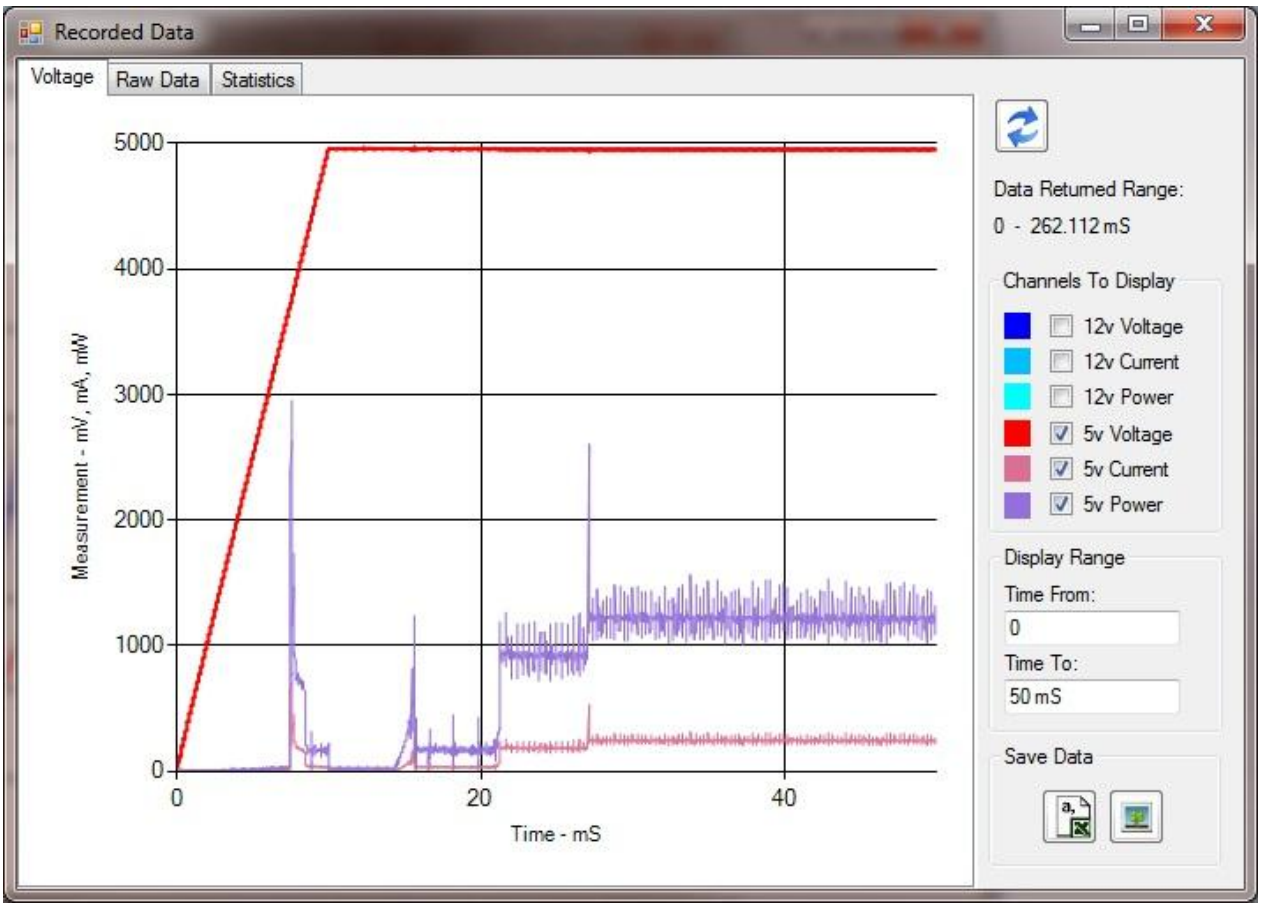



Only the voltage measurement are shown in this example. Check the check boxes to add and remove other items
from the graph. If you chose to disable some of the record options on the main screen then you will not be able
to display them here, the options will be greyed out.

Initially, the full data range is displayed but our pattern only lasted for a short time.




Zooming in to 50mS, we can see that the voltage outputs ramped up exactly as expected and with very little noise

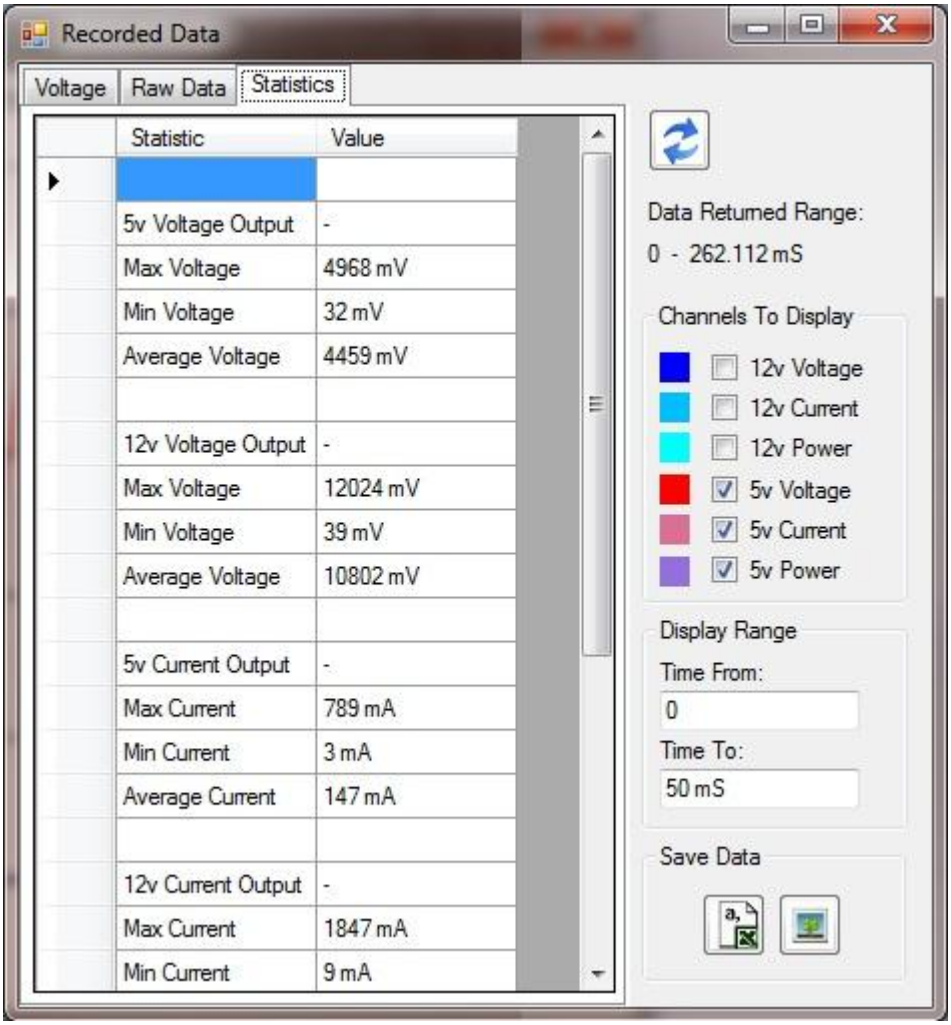



Looking only at the 5v channel, we can add the ***power*** and ***current*** selections and see that the drive does not pull much power until the 5v rail reaches over 3000mV
This recording was taken with the ~16uS measurement averaging selection. If we edit this on the main screen to ***No Averaging*** and run the same pattern again then we will get more detail:

## Export Data

We can see the data behind the graphs on the ***Raw Data*** tab. The ***Save Data*** controls allows this to be exported as a CSV file. Images of the graphs can also be saved in .JPG format

## Statistics




We can also see the statistics over the current view area. These describe min/max and average output levels.