SWEET DREAMS, BABY!
POW!

SMASH Functionality

Release v0.6.1

SERENITY Collaboration

Rev2.0: March 4th, 2019

TABLE OF CONTENTS:

Scope of this Document

This document describes the low level functionality of the SERENITY Board (e.g. how to power parts on and off and also create a Xilinx Virtual Cable (XVC) to upload firmware to the daughter cards). For general (and first time) usage of the SERENITY it is suggested to follow these instructions: SERENITY first steps. The *Useful Commands* section describes basic operation such as powering on daughter cards and checking the status of devices on the board, along with instantiating the XVC (Xilinx Virtual Cable) for uploading bit files. The *Advanced Usage* page describes the plethora of commands available within SMASH and the ways in which to deploy them including clock configuration (jitter cleaners) and Fireflys.

CHAPTER

ONE

AN OVERVIEW OF SMASH: THE SERENITY MANAGEMENT SHELL

[edit this page]

1.1 SMASH, what and why?

SMASH is a framework for controlling hardware. And firmware. And software.

The usual problem in writing generic, reusable control software is the complex interdependence of different components: for example, to configure to an optical module you may first need to talk via I2C to a port-expander to read the module-present signal to check that the optical module is present; to then configure the port-expander via I2C to asset the optical module's module-select line; to talk I2C to the optical module itself; and to reconfigure the port-expander via I2C to deassert the optical module's module-select line. Yet the simple statement of "talk via I2C" may, itself, mask myriad complexities since this may be two independent I2C buses for the port-expander and the optical module; if our software is running on a device without embedded I2C controllers, we shall require an additional transport-layer to talk to a device which converts PCIe, USB or Ethernet to I2C. And if the I2C port-expander becomes deprecated, or we chose to drive the module-present and -select line from the GPIO of an FPGA or SOC, or your pesky hardware designer inserts an I2C bus-multiplexer on the I2C bus which itself requires its own layer of configuration, then there is typically nothing for it but to write completely new software, copying-and-pasting your old code, and hacking it around. If we start a new board, we start a new control software.

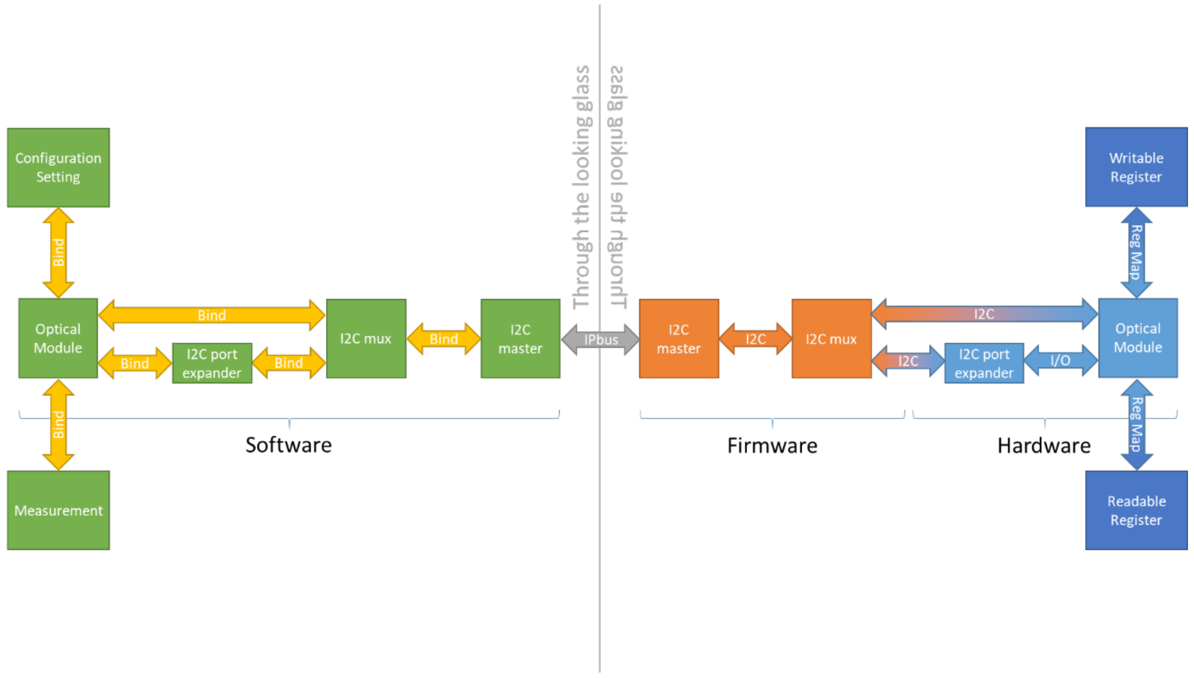
In the past, major structural changes to a board design and entirely new board designs happened on a sufficiently infrequent basis as to be manageable, albeit inefficient, way of working.

Serenity, however, is a different beast: its paradigm of structural reconfigurability and use of pluggable modules means that the same board revision may be being used simultaneously for use-cases with different types or counts of FPGAs, with or without a QSFP, and with different numbers, directions and speeds of serial links via passive-electrical, or uni- or bi-directional, 16G or 28G firefly optics. And with proposals for alternative form-factors and for different control modules (such as SOCs) this problem would only get worse.

This is the problem SMASH was intended to solve. Of course, if the system is sufficiently flexible to handle all these permutations, then it is a minimal leap to being sufficiently flexible for any hardware.

1.2 SMASH, how?

SMASH achieves this by building, in software, a mirror image of the hardware and firmware.



Each type of hardware or firmware has a separate class representation, which is referred to as an "element". Each element is a self-contained entity and has no dependence on any other element. Like physical components, which interact with each other by exposing interfaces, be that GPIO, I2C, SPI, JTAG, USB, GbE, PCIe or some other, so do SMASH elements.

SMASH elements do this through bound function calls, which in SMASH parlance are called "ports". Any element which provides a service, such as an I2C master or a GPIO-driver, creates and exposes a port. Any element which uses a service, such as an I2C slave or a component configured by a GPIO, uses a copy of a bound function (not caring where it came from) as a tool to perform the required operation, without any knowledge of the mechanism by which it is implemented, maintaining the conceptual independence of the service provider and the service consumer.

Because of the diversity of hardware, firmware and software elements, all exposed interfaces other than the "ports" use strings as the interface mechanism, which has the additional advantage of making SMASH highly suitable for scripting and interactive use.

The heart of SMASH is the `Smash` class. This is a singleton class managing the entire system and is the only "core" infrastructure, everything else is a plug-in module, dynamically loaded at run-time.

CHAPTER

TWO

USEFUL COMMANDS

A small number of commonly used SMASH commands are described here such as powering on the daughter cards. More complete and in-depth commands can be found here: *Advanced Usage*.

2.1 Set up the local environment

```
export LD_LIBRARY_PATH=/opt/cactus/lib:/opt/smash/lib:$LD_LIBRARY_PATH
export SMASH_PATH=/opt/smash
export PATH=/opt/smash/bin:$PATH
```

In the following commands a configuration/settings file is required in order to define the actual hardware board that is being interrogated/controlled via SMASH - *someBoardConfig.smash*. This file can define the board configuration (e.g. bindings between components and sockets) as well as settings (e.g. clock or firefly set-up). The directory `/opt/smash/etc/serenity/configuration` should be investigated in order to see which configurations are available and choose the appropriate one. The choice of daughter card FPGA (DC*) is obviously important along with the Firefly optical connections and whether they are transmit, receive or bi-directional. The basic configuration, `BareBoard.smash`, is a sensible starting point if you want to control and test components on the base board and can also then be customised according to your actual board. In addition it is possible to remove the `-c config_file` construction if you have already set the `SMASH_DEFAULT_CONFIG` environment variable.

SMASH can be run in a number of different ways to facilitate flexible deployment:

- smash.exe `-i` option: All the SMASH commands can be run interactively
- smash.exe `"my command1" "my command2"`: appended in the command-line
- a default configuration can be applied (obviates the `-c filename` requirement):

  ```
  export SMASH_DEFAULT_CONFIG=/path/to/someBoardConfig.smash
  ```

It is suggested to read the more complete list within *Advanced Usage* to understand better the different possibilities of many of the (more powerful) commands.

Sometimes the ARTIX FPGA is not recognised on the PCIe bus therefore the `pcie_reconnect` script needs to be run at least once:

```
sudo pcie_reconnect_serenity.sh
```

If your path is not recognised by sudo then:

```
sudo /opt/smash/bin/pcie_reconnect_serenity.sh
```

2.2 Common workflows:

Common workflows and the associated SMASH commands are described below. A more exhaustive list and documentation is provided here: *Advanced Usage*.

2.2.1 Power On or Off both of the processing FPGAs

Note that different daughter cards require slightly different voltages for operation. Automatic reconfiguration of the power supplies depending on the daughter card type is underway, but until it is complete do not swap daughter cards between boards unless you an expert and know how to reconfigure the power supplies.

```
smash.exe -c /path/to/someBoardConfig.smash "X0:Power On"
smash.exe -c /path/to/someBoardConfig.smash "X0:Power Off"
```

2.2.2 XVC (Xilinx Virtual Cable)

To connect a DC to the XVC:

- In interactive mode:
  - JTAGmux Select IPbusJTAG
  - DC0:FPGA Decorate XVC 5000

This can also be included in the configuration script, but note that XVC will only be running as long as smash is open. Alternatively the `-b` option can be added to the **smash.exe** command. This "blocking" behaviour will keep XVC running at all times.

Once XVC is running on the ComExpress it is then necessary to tell Vivado how to connect to the virtual cable (through ethernet). This is done using TCL commands which can either be inputed within the console of *Vivado* or run as a TCL script (tools -> run script). The above `XVC Run` will output to the screen the necessary lines to execute, e.g.:

```
open_hw
connect_hw_server
open_hw_target -xvc_url 128.141.YYY.XXX:5000
set current_hw_jtag [get_property hw_jtag [get_hw_target localhost:3121/xilinx_tcf/
↪Xilinx/128.141.YYY.XXX:5000]]
```

where XXX and YYY are your own unique subnet address. The 3121 is a Vivado default and the 5000 is the port which needs to be open for communication to work.

- If the XVC fails to connect then there is a chance that the firewall on the ComExpress is somehow blocking it. In which case first check that 5000 is open:

  ```
  sudo firewall-cmd --zone=public --list-ports
  ```

  If port 5000 is not listed then you can first try adding it:

  ```
  sudo firewall-cmd --zone=public --add-port=5000/tcp --permanent
  ```

  But if connection still fails (timeouts) then it is suggested to disable the firewall completely:

  ```
  sudo systemctl stop firewalld
  ```

  Then restart "XVC Run" and re-run the TCL script within Vivado.

2.2.3 Status Checks

The LTM4677 PSUs supply the FPGA transceivers (~1V), optical parts (1.8V) & general I/O (1.8V). The NDM3Zs supply the FPGA core voltage (~1V) and optical parts (3.3V)

```
smash.exe -c /path/to/someBoardConfig.smash "Foreach type LTM.* Measure .*"
smash.exe -c /path/to/someBoardConfig.smash "Foreach type NDM.* Measure .*"
smash.exe -c /path/to/someBoardConfig.smash "Foreach type .* Validate"
```

The last command essentially scans the I2CBus to give the result of all connected devices and includes a check on the front-panel LEDs.

2.2.4 25g Firefly Setup

The 25g bi-directional Fireflys have the possiblity of introducing CDR (clock data recovery) which is required for data transmission. However, the have two limited ranges of working:

- 25.5 - 26.0 GHz - important to set the link to within this range and also set the CDR rate to `Lo`
- 27.5 - 28.05 GHz - important to set the link to within this range and also set the CDR rate to `Hi`

Make sure he has the correct range enabled (below).

- Hard reset of the Firefly:

  ```
  Foreach name biFF.*s  "Hard Reset"
  ```

  Note that this reset can disturb the PLL lock on the FPGA transceiver so is only advisable **before** programming the FPGA.

- Enable the Firefly (both transmit and receive):

  ```
  Foreach name biFF.*s "Tx Transmit" 1-4 Enable
  Foreach name biFF.*s "Rx Output" 1-4 Enable
  ```

- Enable the CDR:

  ```
  Foreach name biFF.*s "Tx CDR" 1-4 Enable
  Foreach name biFF.*s "Rx CDR" 1-4 Enable
  ```

- Switch the range of the CDR appropriately:

  ```
  Foreach name biFF.*s "Set Tx CDR rate" 1-4 Lo
  Foreach name biFF.*s "Set Rx CDR rate" 1-4 Lo
  ```

CHAPTER

THREE

ADVANCED USAGE

[edit this page]

The following is an in-depth list of commmands that cover more advanced configuration and usage of the hardware on the SERENITY boards including clock jitter cleaners and the Fireflys transceivers. For basic usage of the SERENITY it is suggested to start here: SERENITY first steps. Some more basic SMASH are also covered within the *Useful Commands* section.

SMASH employs *regex* for taking wildcards and expansions of commands.

Note that commands which contain spaces (e.g. `Foreach type LTM.* "Store to NVM"`) must be delimited by double quotes. The tab-completion feature will auto-complete commands, but without the first double quote the command will not be executed correctly.

3.1 Set up the local environment

```
export LD_LIBRARY_PATH=/opt/cactus/lib:/opt/smash/lib:$LD_LIBRARY_PATH
export SMASH_PATH=/opt/smash
export PATH=/opt/smash/bin:$PATH
```

In the following commands a configuration/settings file is required in order to define the actual hardware board that is being interrogated/controlled via SMASH - *someBoardConfig.smash*. This file can define the board configuration (e.g. bindings between components and sockets) as well as settings (e.g. clock or firefly set-up). The directory `/opt/smash/etc/serenity` should be investigated in order to see which configurations are available and choose the appropriate one. Alternatively all the configurations could be loaded sequentially if you want the maximum options (testing purposes):

Apply a fixed configuration file (obviates the `-c filename` requirement):

```
export SMASH_DEFAULT_CONFIG=/path/to/someBoardConfig.smash
```

Run interactively:

```
smash.exe -i

or
```

```
smash.exe -c /path/to/someBoardConfig.smash -i
```

Run commands on the command-line (list of quoted commands can be extended):

```
smash.exe -c /path/to/someBoardConfig.smash "my command1" "my command2"
```

or:

```
smash.exe "my command1" "my command2"
```

if the `SMASH_DEFAULT_CONFIG` environment variable has been set.

Sometimes the ARTIX FPGA is not recognised on the PCIe bus therefore the `pcie_reconnect` script needs to be run at least once:

```
sudo pcie_reconnect_serenity.sh
```

If your path is not recognised by sudo then:

```
sudo /opt/smash/bin/pcie_reconnect_serenity.sh
```

3.2 Detailed Commands

A number of simple commands are provided here within the *Useful Commands* section. Here are described more elaborate or complicated functionality of SMASH.

3.2.1 Power On or Off both of the processing FPGAs

Note that different daughter cards require slightly different voltages for operation. Automatic reconfiguration of the power supplies depending on the daughter card type is underway, but until it is complete do not swap daughter cards between boards unless you an expert and know how to reconfigure the power supplies.

```
smash.exe -c /path/to/someBoardConfig.smash "X0:Power On"
smash.exe -c /path/to/someBoardConfig.smash "X0:Power Off"
```

To power on a given site:

- From the commandline:
  - script option:

```
smash.exe -c /opt/smash/etc/serenity/configuration/[Your board configuration]
↪"run /opt/smash/etc/serenity/examples/power_x0_on.smash"
```

  - Embedded command:

```
smash.exe -c /opt/smash/etc/serenity/configuration/[Your board configuration]␣
↪“X0:Power On”
```

- Interactive mode:

```
Run /opt/smash/etc/serenity/examples/power_x0_on.smash
```

  - Or the much simpler:

```
X0:Power On
```

To power off a given site:

- From the commandline:

```
smash.exe -c /opt/smash/etc/serenity/configuration/[Your board configuration] "Run /
↪opt/smash/etc/serenity/examples/power_x0_off.smash"
```

  - Or

```
smash.exe -c /opt/smash/etc/serenity/configuration/[Your board configuration]␣
↪“X0:Power Off”
```

- Or in interactive mode:

```
Run /opt/smash/etc/serenity/examples/power_x0_off.smash
```

  Or the much simpler:

```
X0:Power Off
```

To power on both sites:

- From the commandline:

```
smash.exe -c /opt/smash/etc/serenity/configuration/[Your board configuration]␣
↪“X0:Power On” “X1:Power On”
```

  - Or:

```
smash.exe -c /opt/smash/etc/serenity/configuration/[Your board configuration]␣
↪“Foreach name X.:Power On”
```

- In interactive mode:

```
Foreach name X.:Power On
```

To power off both sites:

- From the commandline:

```
smash.exe -c /opt/smash/etc/serenity/configuration/[Your board configuration]␣
↪“X0:Power Off” “X1:Power Off”
```

  - Or:

```
smash.exe -c /opt/smash/etc/serenity/configuration/[Your board configuration]␣
↪“Foreach name X.:Power Off”
```

- In interactive mode:

```
Foreach name X.:Power Off
```

3.2.2 Configure both the SI5345 clock jitter cleaners on an interposer

Allows configuration of the clock jitter cleaners with a text file generated the SiLabs ClockBuilder Pro software.

```
smash.exe -c /path/to/someBoardConfig.smash "Foreach type Si5345 Configure /opt/smash/
↪etc/serenity/clocks/Si5345-RevB-LHCP100F-100MHz-LHC-Precision-Clock-Registers-v2.txt
```

where the filename is just one example which maybe packaged within the smash RPM.

The Clock Pro software for configuring the clocks using I2C can be downloaded from the SiLabs-website (windows only) and the appropriate chip is a B-part.

The directory /opt/smash/etc/serenity/clocks/ contains a number of pre-built clock settings file. The correct one should be chosen for the desired clock configuration.

More information about the clock distribution network available on the SERENITY is described here.

3.2.3 Status Checks

The LTM4677 PSUs supply the FPGA transceivers (~1V), optical parts (1.8V) & general I/O (1.8V). The NDM3Zs supply the FPGA core voltage (~1V) and optical parts (3.3V)

```
smash.exe -c /path/to/someBoardConfig.smash "Foreach type LTM.* Measure .*"
smash.exe -c /path/to/someBoardConfig.smash "Foreach type NDM.* Measure .*"
smash.exe -c /path/to/someBoardConfig.smash "Foreach type .* Validate"
```

The last command essentially scans the I2CBus to give the result of all connected devices.

3.2.4 Apply multiple configuration files within one directory (alphanumeric order):

```
smash -c "[some_path]/*.smash"
```

Note that you need the quotes to prevent it being shell-expanded on the commandline. **However**, the scripts directory is deliberately configured to separate the "base board" from the "configuration". The configuration scripts explicitly instantiate the bare board in each case and then create and plug a daughtercard and any fireflys. You will find a folder called "configurions" which contain such useful scripts as "BareBlade", "RalBoard", "TomOpticalTests" and "PizzaBox". At least two of these scripts also create and configure the XVC on behalf of the end user.

3.2.5 To connect a DC to the JTAG header:

- From the commandline:

  ```
  smash.exe -c /opt/smash/etc/serenity/configuration/[Your board configuration] 'Run /
  ↪opt/smash/etc/serenity/examples/jtag_header_x0.smash'
  ```

- Or in interactive mode:

  ```
  Run /opt/smash/etc/serenity/examples/jtag_header_x0.smash
  ```

- Or explicitly:

```
JTAGmux Select JTAGheader
DC0:FPGA JtagManualConnect
DC1:FPGA JtagManualConnect
```

The preferred method of connection is to use an XVC (Xilinx Virtual Cable) rather than using a JTAG interface. Instructions for setting up the XVC are described in *XVC (Xilinx Virtual Cable)*. The configuration to setup SMASH for XVC is described below.

To connect a DC to the XVC:

- In interactive mode:

```
JTAGmux Select IPbusJTAG
DC0:FPGA Decorate XVC 5000
```

To connect the other DC replace `DC0` with `DC1`. It is no longer possible to daisy-chain the XVC. The XVC decoration should print to the screen a series of TCL instructions which can be run in Vivado to activate the XVC for programming of the FPGA. In the case of port forwarding (tunnelling) the IP address has to be replaced with localhost on the Vivado (hw_server) machine.

If you want the full JTAG chain printed out

```
XVC Print
```

3.2.6 Tunnel to run XVC through a port forward

Tunnel to run XVC through a port forward (server PC) onto a local subnet of 10.0.0.XX:

```
JTAGmux Select IPbusJTAG
DC0:FPGA Decorate XVC 5000
```

In order to open up the 5000 port through a tunnel the following command is required:

```
ssh -L 5000:localhost:5000 -p 222 cmx@pcuptracker001.cern.ch
```

Note that in the case of port forwarding then the command `open_hw_target -xvc_url 128.141.YYY.XXX:5000` given by SMASH needs to be modified such that `128.141.YYY.XXX:5000` instead says `localhost:5000`.

3.2.7 Voltage and Current Measurements

To measure voltages on the LTMs and NDMs:

- In interactive mode:

```
Foreach type LTM.* Measure Vout.*
Foreach type NDM.* Measure Vout.*
```

To measure currents on the LTMs and NDMs:

- In interactive mode:

```
Foreach type LTM.* Measure Iout.*
Foreach type NDM.* Measure Iout.*
```

To measure input voltages on the LTMs and NDMs:

- In interactive mode:

```
Foreach type LTM.* Measure Vin
Foreach type NDM.* Measure Vin
```

3.2.8 Example Configuration File:

TomOpticalTests.smash:

```
Run /opt/smash/etc/serenity/base/atca/*.smash

Run ../DaughterCards/KIT_KU15P.smash DC1
Plug I1 DC1 # Socket - Module ordering

Create FireflyBi FF1

Plug J1:19 FF1 # Socket - Module ordering

# By default we want to route all JTAG via IPbus
JTAGmux Select IPbusJTAG

DC1:FPGA Decorate XVC 5000
```

- The `Create FireflyBi FF1` adds the physical firefly to the element list.
- The `Plug J1:19 FF1` binds FF1 to the socket J1:19.

3.2.9 Locate Command:

- If you wish to see where a physical component is on the SERENITY board then the very handy `Locate` command can be run:

```
Locate FF1
Locate J1:19
```

Both should give the same answer.

3.2.10 Firefly Configuration:

The different types of Firefly: **bidirectional, passive, tx & rx** all have different funtionality within SMASH.

- For the Bi's, the functions are:

```
"Tx Transmit" [Channel expression] [Enable|Disable]
"Rx Output" [Channel expression] [Enable|Disable]
"Tx CDR" [Channel expression] [Enable|Disable]
"Rx CDR" [Channel expression] [Enable|Disable]

"Tx Squelch" [Channel expression] [Enable|Disable]
"Rx Squelch" [Channel expression] [Enable|Disable]

Or "Foreach type FireflyBi"
```

(continues on next page)

(continued from previous page)

```
"Set Tx CDR rate" [Channel expression] [Hi|Lo]
"Set Rx CDR rate" [Channel expression] [Hi|Lo]
"Set Tx Polarity" [Channel expression] [Normal|Invert]
"Set Rx Polarity" [Channel expression] [Normal|Invert]
"Set Tx Equalization" [Channel expression] [9.8dB|8.8dB|8.2dB|7.2dB|6.5dB|4.8dB|3.
↪7dB|2.7dB|1.9dB|1.3dB]
"Set Rx Pre-emphasis" [Channel expression] [7.5dB|6dB|5dB|4dB|3dB|2dB|1dB|0dB]
"Set Rx Amplitude" [Channel expression]  [300mV|450mV|600mV|900mV]
```

- For unidirectionals there is a lot less configurability
  - Tx commands are:

    ```
    "Soft Reset"
    "Channel" [Channel expression] [Enable|Disable]
    "Set Polarity" [Channel expression] [Normal|Invert]
    ```

  - Rx, the commands are:

    ```
    "Soft Reset"
    "Channel" [Channel expression] [Enable|Disable]
    "Output" [Channel expression] [Enable|Disable]
    "Set Amplitude" [Channel expression] [Off|Low|Medium|High]
    "Set De-emphasis" [Channel expression] [Off|On]
    ```

- Then it is possible to return the I2C information from the Fireflys:

```
Foreach type FireflyBi.* Validate
Foreach type FireflyRx.* Validate
Foreach type FireflyTx.* Validate

"Foreach type Firefly.* Validate"
```

Note the incolusion of a dot before the *, if you instead used `Firefly*` this would match Firefl, Firefly, Fireflyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy but not FireflyBi, FireflyRx or FireflyTx

3.2.11 Firefly Hard Reset:

```
ForEach type Firefly.* "Hard Reset"

biFF2n "Hard Reset"
```

3.2.12 Firefly High-Speed

- If you want to run the FireflyBi's at 25.5 - 26.0 GHz then CDR must be set `Lo`

  ```
  Foreach type FireflyBi Set Tx CDR rate [Channel expression] Lo
  ```

- If you want to run the FireflyBi's at 27.5 - 28.05 GHz then CDR must be set `Hi`

  ```
  Foreach type FireflyBi Set Tx CDR rate [Channel expression] Hi
  ```

Anything outside these ranges may not work unless CDR is off.

If the FPGA or whole board are power-cycled then a hard reset is requires of the Fireflys:

```
ForEach type Firefly.* "Hard Reset"
```

3.3 List Available Options in SMASH

```
[element name] List [Functions|Ports|Measurements|Attributes]
```

e.g.

```
U:6 List Functions
```

Will give the following output:

```
SetAttribute
List
Schedule
Measure
Print
Validate
```

Or for the Si5345 clock jitter cleaners:

```
U1:3 List Functions
```

will give:

```
Disable
SetAttribute
Enable
List
Schedule
Configure
```

(continues on next page)

(continued from previous page)

```
Measure
Print
Validate
```

CHAPTER

FOUR

FULL COMMAND SET FOR SMASH

SMASH employs *regex* for taking wildcards and expansions of commands.

Note that commands which contain spaces (e.g. `Foreach type LTM.* "Store to NVM"`) must be delimited by double quotes. The tab-completion feature will auto-complete commands, but without the first double quote the command will not be executed correctly.

4.1 Core – Abstract Base Elements

4.1.1 Element

The base class of all SMASH components

Functions declared in Element —

- List — List the objects known to the element.
- The object type to be listed. Valid options:

```
Functions - Ports - Measurements - Attributes
```

- Measure — Return the measured value of declared quantities.
- Any number of regular expressions for matching measurement names
- Print — Print all auxilliary attributes.
- Schedule — Schedule one or more measurements.
- A time period
- Any number of measurements to be added to the schedule [CURRENTLY UNUSED]
- SetAttribute — Set auxilliary attributes of the element.
- Any number of key=value expressions or key_a=key_b=key_c=value expressions
- Validate — Have the element validate itself.

4.1.2 I2CeepromElement

A base class for I2C eeproms

Constructor —

- Optional: Retry limit (for multimaster I2C buses). [Default value = 1]

Functions declared in I2CeepromElement —

- Read — Read the EEPROM and print it as an ASCII string.
- Write — Write an ASCII string to the EEPROM.
- A string to be written to the eeprom

Functions inherited from *Element* —

- List
- Measure
- Print
- Schedule
- SetAttribute
- Validate

4.1.3 I2Celement

A base class for I2C elements

Functions inherited from *Element* —

- List
- Measure
- Print
- Schedule
- SetAttribute
- Validate

4.1.4 JTAGelement

A base class for JTAG devices

Functions declared in JTAGelement —

- JtagManualConnect — Manually force the selection of this device in any upstream switch.

Functions inherited from *Element* —

- List
- Measure
- Print
- Schedule

- SetAttribute
- Validate

4.1.5 OpticalModuleElement

A base class for pluggable optical devices based on the SFP principle

Functions declared in OpticalModuleElement —

- Hard Reset — Strobe the module-reset line for 1ms.

Functions inherited from *Element* —

- List
- Measure
- Print
- Schedule
- SetAttribute
- Validate

4.1.6 PMbusElement

A base class for PMbus devices

Functions declared in PMbusElement —

- Off — Turn the PMbus device off.
- On — Turn the PMbus device on.
- Store to NVM — Store the current device configuration to NVM [Requires interactive running].

Functions inherited from *Element* —

- List
- Measure
- Print
- Schedule
- SetAttribute
- Validate

4.1.7 SocketElement

A base class for socket elements

Functions inherited from *Element* —

- List
- Measure
- Print

- Schedule
- SetAttribute
- Validate

4.2 Core Elements

4.2.1 Assert

Assert that a measurement has a specific value

Custom process —

- TO DO

4.2.2 Bind

A component which acts like a keyword to join the input port on a slave device to a declared port on a master device

Functions declared in Bind —

- I2C — Using I2C protocol.
- A service provider (with bracketed port index if necessary)
- A list of consumers (with bracketed port index if necessary)
- IO — Using GPIO protocol.
- A service provider (with bracketed port index if necessary)
- A list of consumers (with bracketed port index if necessary)
- IPbus — Using IPbus protocol.
- A service provider (with bracketed port index if necessary)
- A list of consumers (with bracketed port index if necessary)
- JTAG — Using JTAG protocol.
- A service provider (with bracketed port index if necessary)
- A list of consumers (with bracketed port index if necessary)
- Power — Using Power protocol.
- A service provider (with bracketed port index if necessary)
- A list of consumers (with bracketed port index if necessary)
- SPI — Using SPI protocol.
- A service provider (with bracketed port index if necessary)
- A list of consumers (with bracketed port index if necessary)

Functions inherited from *Element* —

- List
- Measure
- Print

- Schedule
- SetAttribute
- Validate

4.2.3 Create

Create an element of a specific type

Custom process —

- TO DO

4.2.4 Foreach

Iterate over all matching names or types

Custom process —

- TO DO

4.2.5 Locate

Load a GUI window to locate a component on a board

Custom process —

- TO DO

4.2.6 Plug

Connect a ‘Pluggable’ element into a ‘Socket’ element

Custom process —

- TO DO

4.2.7 Run

Run a SMASH script

Custom process —

- TO DO

4.2.8 Unbind

A component which acts like a keyword to uncouple the input port on a slave device to a declared port on a master device

Functions declared in Unbind —

- I2C — Using I2C protocol.
- A list of consumers (with bracketed port index if necessary) to unbind
- IO — Using GPIO protocol.
- A list of consumers (with bracketed port index if necessary) to unbind
- IPbus — Using IPbus protocol.
- A list of consumers (with bracketed port index if necessary) to unbind
- JTAG — Using JTAG protocol.
- A list of consumers (with bracketed port index if necessary) to unbind
- Power — Using Power protocol.
- A list of consumers (with bracketed port index if necessary) to unbind
- SPI — Using SPI protocol.
- A list of consumers (with bracketed port index if necessary) to unbind

Functions inherited from *Element* —

- List
- Measure
- Print
- Schedule
- SetAttribute
- Validate

4.3 Components

4.3.1 AT24CS02

An ATMEL I2C EEPROM with UID

Functions inherited from *Element* —

- List
- Measure
- Print
- Schedule
- SetAttribute
- Validate

Functions inherited from *I2CeepromElement* —

- Read
- Write

4.3.2 AT24MAC602

An ATMEL I2C EEPROM with UID

Functions inherited from *Element* —

- List
- Measure
- Print
- Schedule
- SetAttribute
- Validate

4.3.3 BCM53134

The Broadcomm BCM53134 6-port Gigabit Ethernet Switch

Constructor —

- Chip revision for validation

Functions declared in BCM53134 —

- Too many functions to list, please use tab-completion from an interactive console or use "<Name> List Functions"

Functions inherited from *Element* —

- List
- Measure
- Print
- Schedule
- SetAttribute
- Validate

4.3.4 DaughterCard

A Serenity form-factor Daughter-card

Functions inherited from *Element* —

- List
- Measure
- Print
- Schedule
- SetAttribute

- Validate

4.3.5 DaughterCardSocket

A socket for a Serenity DaughterCard pluggable

Functions inherited from *Element* —

- List
- Measure
- Print
- Schedule
- SetAttribute
- Validate

4.3.6 DummyComponent

A base class for I2C eeproms

Constructor —

- Initial vowel rotation for string munger
- Initial consonant rotation for string munger

Functions declared in DummyComponent —

- DocTest Dimensioned — Test Dimensioned.
- A test. Specified with units of 'Hz', Si prefixes are allowed
- A test. Specified with units of 'Hz', Si prefixes are allowed
- DocTest FilePath — Test FilePath.
- File path: A test
- File path: A test
- DocTest Optional — Test Optional.
- Optional: A test. [Default value = 42]
- Optional: A test. [Default value = 42]
- DocTest PlainDoc — Test plain documentation.
- A test
- A test
- DocTest SelectFrom1 — Test SelectFrom std::map.
- A test. Valid options:

  ```
  A - B
  ```

- A test. Valid options:

```
A - B
```

- DocTest SelectFrom2 — Test SelectFrom std::unordered_map.
- A test. Valid options:

  ```
  C - D
  ```

- A test. Valid options:

  ```
  C - D
  ```

- DocTest SelectFrom3 — Test SelectFrom std::vector.
- A test. Valid options:

  ```
  E - F
  ```

- A test. Valid options:

  ```
  E - F
  ```

- DocTest SelectFrom4 — Test SelectFrom std::list.
- A test. Valid options:

  ```
  G - H
  ```

- A test. Valid options:

  ```
  G - H
  ```

- DocTest SelectFrom5 — Test SelectFrom initializer-list.
- A test. Valid options:

  ```
  I - J
  ```

- A test. Valid options:

  ```
  I - J
  ```

- Force Exception — Force an exception to be thrown.
- Munge — Print a munged version of the arguments which follow.
- Any number of plain-text arguments to be consumed and munged

Functions inherited from *Element* —

- List
- Measure
- Print
- Schedule
- SetAttribute
- Validate

4.3.7 Firefly

SSAMTEC FireFly optical modules

Constructor —

- A full Samtec Firefly part number. For 'Y' cables, please substitute 'T' or 'R' to specify whether the site contains a Tx or Rx module.

Functions declared in Firefly —

- Amplitude — Amplitude.
- A channel expression: a comma-delimeted list of channels (1,2,3), an inequality (<4) or a range (1-3), where the indices range from 0-11 for unidirectional parts and 1-4 for bidirectional parts
- Amplitude. Valid options:

  ```
  High - Low - Medium - Off
  ```

- CDR — CDR.
- A channel expression: a comma-delimeted list of channels (1,2,3), an inequality (<4) or a range (1-3), where the indices range from 0-11 for unidirectional parts and 1-4 for bidirectional parts
- CDR. Valid options:

  ```
  Disable - Enable
  ```

- Channel — Channel.
- A channel expression: a comma-delimeted list of channels (1,2,3), an inequality (<4) or a range (1-3), where the indices range from 0-11 for unidirectional parts and 1-4 for bidirectional parts
- Channel. Valid options:

  ```
  Disable - Enable
  ```

- De-emphasis — De-emphasis.
- A channel expression: a comma-delimeted list of channels (1,2,3), an inequality (<4) or a range (1-3), where the indices range from 0-11 for unidirectional parts and 1-4 for bidirectional parts
- De-emphasis. Valid options:

  ```
  Disable - Enable
  ```

- Equalization — Equalization.
- A channel expression: a comma-delimeted list of channels (1,2,3), an inequality (<4) or a range (1-3), where the indices range from 0-11 for unidirectional parts and 1-4 for bidirectional parts
- Equalization. Valid options:

  ```
  0 - 1 - 2 - 3 - 4 - 5 - 6 - 7 - Max - Min - None
  ```

- LOS alarms — LOS alarms.
- A channel expression: a comma-delimeted list of channels (1,2,3), an inequality (<4) or a range (1-3), where the indices range from 0-11 for unidirectional parts and 1-4 for bidirectional parts
- LOS alarms. Valid options:

```
Fault - "No fault"
```

- Output — Output.
- A channel expression: a comma-delimeted list of channels (1,2,3), an inequality (<4) or a range (1-3), where the indices range from 0-11 for unidirectional parts and 1-4 for bidirectional parts
- Output. Valid options:

```
Disable - Enable
```

- Polarity — Polarity.
- A channel expression: a comma-delimeted list of channels (1,2,3), an inequality (<4) or a range (1-3), where the indices range from 0-11 for unidirectional parts and 1-4 for bidirectional parts
- Polarity. Valid options:

```
Invert - Normal
```

- Rx Amplitude — Rx Amplitude.
- A channel expression: a comma-delimeted list of channels (1,2,3), an inequality (<4) or a range (1-3), where the indices range from 0-11 for unidirectional parts and 1-4 for bidirectional parts
- Rx Amplitude. Valid options:

```
300mV - 450mV - 600mV - 900mV - Max - Min
```

- Rx CDR — Rx CDR.
- A channel expression: a comma-delimeted list of channels (1,2,3), an inequality (<4) or a range (1-3), where the indices range from 0-11 for unidirectional parts and 1-4 for bidirectional parts
- Rx CDR. Valid options:

```
Disable - Enable
```

- Rx CDR rate — Rx CDR rate.
- A channel expression: a comma-delimeted list of channels (1,2,3), an inequality (<4) or a range (1-3), where the indices range from 0-11 for unidirectional parts and 1-4 for bidirectional parts
- Rx CDR rate. Valid options:

```
Hi - Lo
```

- Rx Output — Rx Output.
- A channel expression: a comma-delimeted list of channels (1,2,3), an inequality (<4) or a range (1-3), where the indices range from 0-11 for unidirectional parts and 1-4 for bidirectional parts
- Rx Output. Valid options:

```
Disable - Enable
```

- Rx Polarity — Rx Polarity.
- A channel expression: a comma-delimeted list of channels (1,2,3), an inequality (<4) or a range (1-3), where the indices range from 0-11 for unidirectional parts and 1-4 for bidirectional parts
- Rx Polarity. Valid options:

```
Invert - Normal
```

- Rx Pre-emphasis — Rx Pre-emphasis.
- A channel expression: a comma-delimeted list of channels (1,2,3), an inequality (<4) or a range (1-3), where the indices range from 0-11 for unidirectional parts and 1-4 for bidirectional parts
- Rx Pre-emphasis. Valid options:

```
0dB - 1dB - 2dB - 3dB - 4dB - 5dB - 6dB - 7.5dB - Max - Min
```

- Rx Squelch — Rx Squelch.
- A channel expression: a comma-delimeted list of channels (1,2,3), an inequality (<4) or a range (1-3), where the indices range from 0-11 for unidirectional parts and 1-4 for bidirectional parts
- Rx Squelch. Valid options:

```
Disable - Enable
```

- Soft Reset — Write to the reset register.
- Squelch — Squelch.
- A channel expression: a comma-delimeted list of channels (1,2,3), an inequality (<4) or a range (1-3), where the indices range from 0-11 for unidirectional parts and 1-4 for bidirectional parts
- Squelch. Valid options:

```
Disable - Enable
```

- Tx CDR — Tx CDR.
- A channel expression: a comma-delimeted list of channels (1,2,3), an inequality (<4) or a range (1-3), where the indices range from 0-11 for unidirectional parts and 1-4 for bidirectional parts
- Tx CDR. Valid options:

```
Disable - Enable
```

- Tx CDR rate — Tx CDR rate.
- A channel expression: a comma-delimeted list of channels (1,2,3), an inequality (<4) or a range (1-3), where the indices range from 0-11 for unidirectional parts and 1-4 for bidirectional parts
- Tx CDR rate. Valid options:

```
Hi - Lo
```

- Tx Equalization — Tx Equalization.
- A channel expression: a comma-delimeted list of channels (1,2,3), an inequality (<4) or a range (1-3), where the indices range from 0-11 for unidirectional parts and 1-4 for bidirectional parts
- Tx Equalization. Valid options:

```
1.3dB - 1.9dB - 2.7dB - 3.7dB - 4.8dB - 6.5dB - 7.2dB - 8.2dB -␣
↪8.8dB - 9.8dB - Max - Min
```

- Tx Output — Tx Output.

- A channel expression: a comma-delimeted list of channels (1,2,3), an inequality (<4) or a range (1-3), where the indices range from 0-11 for unidirectional parts and 1-4 for bidirectional parts
- Tx Output. Valid options:

```
Disable - Enable
```

- Tx Polarity — Tx Polarity.
- A channel expression: a comma-delimeted list of channels (1,2,3), an inequality (<4) or a range (1-3), where the indices range from 0-11 for unidirectional parts and 1-4 for bidirectional parts
- Tx Polarity. Valid options:

```
Invert - Normal
```

- Tx Squelch — Tx Squelch.
- A channel expression: a comma-delimeted list of channels (1,2,3), an inequality (<4) or a range (1-3), where the indices range from 0-11 for unidirectional parts and 1-4 for bidirectional parts
- Tx Squelch. Valid options:

```
Disable - Enable
```

Functions inherited from *Element* —

- List
- Measure
- Print
- Schedule
- SetAttribute
- Validate

Functions inherited from *OpticalModuleElement* —

- Hard Reset

4.3.8 FireflyPassive

A Samtec passive Firefly cable

Functions inherited from *Element* —

- List
- Measure
- Print
- Schedule
- SetAttribute
- Validate

4.3.9 FireflySocket

A socket for a Samtec Firefly pluggable

Functions inherited from *Element* —

- List
- Measure
- Print
- Schedule
- SetAttribute
- Validate

4.3.10 Ftdi

FTDI USB-to-Any-serial converters

Constructor —

- Device type
- USB VID as decimal (1027) or hex (0x0403)
- USB PID as decimal (24592) or hex (0x6010)

Functions declared in Ftdi —

- Decorate — Specialize a port to a specific protocol.
- Protocol. Valid options:

  ```
  JTAG - I2C
  ```

- Interface. Valid options:

  ```
  A - B - C - D
  ```

- Protocol Frequency. Specified with units of 'Hz', Si prefixes are allowed
- Disable — Disable this device via the physical reset line (must be connected and available to SMASH).
- Enable — Enable this device via the physical reset line (must be connected and available to SMASH).
- Scan — Scan an I2C bus for any responding addresses.
- Interface. Valid options:

  ```
  A - B - C - D
  ```

- Set Frequency — Set the protocol clock frequency on specified interface.
- Interface. Valid options:

  ```
  A - B - C - D
  ```

- Protocol Frequency. Specified with units of 'Hz', Si prefixes are allowed

Functions inherited from *Element* —

- List
- Measure
- Print
- Schedule
- SetAttribute
- Validate

4.3.11 IPbusI2Cmaster

A firmware I2C master with an IPbus interface

Functions declared in IPbusI2Cmaster —

- Reset — Reset the firmware core.
- Scan — Scan the I2C bus for any responding addresses.

Functions inherited from *Element* —

- List
- Measure
- Print
- Schedule
- SetAttribute
- Validate

4.3.12 IPbusI2Cmux

A firmware I2C mux controlled via an IPbus interface

Functions declared in IPbusI2Cmux —

- Manual Select — Manually select an I2C output.
- Reset — Issue a 1ms reset on the reset line of the specified output.
- An output index

Functions inherited from *Element* —

- List
- Measure
- Print
- Schedule
- SetAttribute
- Validate

4.3.13 IPbusJTAGmaster

A firmware JTAG master with an IPbus interface

Functions declared in IPbusJTAGmaster —

- Set Frequency — Set the protocol clock frequency.
- Protocol Frequency. Specified with units of ‘Hz’, Si prefixes are allowed
- Set Tdo Offset — Set the phase of the TDO capture relative to TDI transmission.

Functions inherited from *Element* —

- List
- Measure
- Print
- Schedule
- SetAttribute
- Validate

4.3.14 IPbusJTAGmux

A firmware JTAG mux controlled via an IPbus interface

Functions declared in IPbusJTAGmux —

- Connect All Targets — Connect all outputs in a daisy-chain, over-riding automated target switching.
- Connect Default Target — Restore automated target switching.
- Select — Select a JTAG input.
- The name of a JTAG source component

Functions inherited from *Element* —

- List
- Measure
- Print
- Schedule
- SetAttribute
- Validate

4.3.15 IPbusSPImaster

A firmware SPI master with an IPbus interface

Functions inherited from *Element* —

- List
- Measure
- Print
- Schedule
- SetAttribute
- Validate

4.3.16 JTAGheader

A JTAG header which cannot be driven by SMASH

Functions inherited from *Element* —

- List
- Measure
- Print
- Schedule
- SetAttribute
- Validate

4.3.17 LTM4677

A Linear Technologies LTM4677 regulator

Functions declared in LTM4677 —

- Clear Fault Log — Reinitialize the fault-logging NVM.
- Clear Status Reg — Set all status register fault bits to 0.
- Configure — Configure the LTM4677 from a Linear Technology PowerPlay configuration file.
- File path: a Linear Technology PowerPlay configuration file
- Force Fault Log — Store an artificial fault to NVM.
- Freq — Set switching frequency.
- A frequency. Valid options:

  ```
  1000kHz - 250kHz - 350kHz - 425kHz - 500kHz - 575kHz - 650kHz -␣
  ↪750kHz
  ```

- Vout 1 — Set the output voltage of channel 1.
- voltage. Specified with units of ‘V’, Si prefixes are allowed
- Vout 2 — Set the output voltage of channel 2.

- voltage. Specified with units of 'V', Si prefixes are allowed

Functions inherited from *Element* —

- List
- Measure
- Print
- Schedule
- SetAttribute
- Validate

Functions inherited from *PMbusElement* —

- Off
- On
- Store to NVM

4.3.18 LTgroup

A group of regulators to be enabled or disabled together

Constructor —

- Any number of components to add to the LTgroup. Used for providing status information only.

Functions declared in LTgroup —

- Configure — Configure the LTM4677 from a Linear Technology PowerPlay configuration file.
- File path: a Linear Technology PowerPlay configuration file
- Off — Turn the LTgroup off.
- On — Turn the LTgroup on.

Functions inherited from *Element* —

- List
- Measure
- Print
- Schedule
- SetAttribute
- Validate

4.3.19 Max6639

Dual fan controller

Functions declared in Max6639 —

- Configure — A hack to set the polarity to enable the fans at 100 percent - needs fixing at some point.

Functions inherited from *Element* —

- List
- Measure
- Print
- Schedule
- SetAttribute
- Validate

4.3.20 NDM2Z

A CUI Inc. NDM2Z regulator

Functions declared in NDM2Z —

- Freq — Set switching frequency.
- A frequency. Valid options:

```
200kHz - 320kHz - 480kHz - 640kHz
```

- Vout — Set the output voltage of channel.
- voltage. Specified with units of 'V', Si prefixes are allowed

Functions inherited from *Element* —

- List
- Measure
- Print
- Schedule
- SetAttribute
- Validate

Functions inherited from *PMbusElement* —

- Off
- On
- Store to NVM

4.3.21 NDM3Z

A CUI Inc. NDM3Z regulator

Functions declared in NDM3Z —

- Freq — Set switching frequency.
- A frequency. Valid options:

```
200kHz - 320kHz - 480kHz - 640kHz
```

- Vout — Set the output voltage of channel.
- voltage. Specified with units of 'V', Si prefixes are allowed

Functions inherited from *Element* —

- List
- Measure
- Print
- Schedule
- SetAttribute
- Validate

Functions inherited from *PMbusElement* —

- Off
- On
- Store to NVM

4.3.22 PCAL6524

An NXP 24-pin I2C port expander

Constructor —

- Default pin configuration - note: not actually set until first usage!

Functions declared in PCAL6524 —

- Configure — Manually force an update of pin state.

Functions inherited from *Element* —

- List
- Measure
- Print
- Schedule
- SetAttribute
- Validate

4.3.23 PowerGroup

A group of regulators to be enabled or disabled together

Constructor —

- Any number of components to add to the power group

Functions declared in PowerGroup —

- Off — Turn the PowerGroup off.
- On — Turn the PowerGroup on.

Functions inherited from *Element* —

- List
- Measure
- Print
- Schedule
- SetAttribute
- Validate

4.3.24 QSFP

A QSFP form-factor optical module

Constructor —

- Manufacturer's name for validation

Functions inherited from *Element* —

- List
- Measure
- Print
- Schedule
- SetAttribute
- Validate

Functions inherited from *OpticalModuleElement* —

- Hard Reset

4.3.25 QSFPsocket

A socket for a QSFP pluggable

Functions inherited from *Element* —

- List
- Measure
- Print

- Schedule
- SetAttribute
- Validate

4.3.26 Si53156

PCIe clock fanout buffer

Functions inherited from *Element* —

- List
- Measure
- Print
- Schedule
- SetAttribute
- Validate

4.3.27 Si5345

SiliconLabs clock synthesizer and jitter attenuator

Functions declared in Si5345 —

- Configure — Configure the Si5345 from a SiLabs configuration file.
- File path: an Si5345 configuration file
- Disable — Disable the Si5345.
- Enable — Enable the Si5345.

Functions inherited from *Element* —

- List
- Measure
- Print
- Schedule
- SetAttribute
- Validate

4.3.28 TLC59208

RGB LED controller

Functions inherited from *Element* —

- List
- Measure
- Print

- Schedule
- SetAttribute
- Validate

4.3.29 XilinxFPGA

A Xilinx FPGA

Constructor —

- The full Xilinx device ID

Functions declared in XilinxFPGA —

- Decorate — Add optional functionality to FPGA.
- Functionality to add. Valid options:

```
PCIe - XVC - Sysmon
```

- Load Bitstream — Load a bitstream via JTAG.
- File path: Path to Bitfile
- PCIe connect — Mount the device on the linux PCIe subsystem.
- PCIe disconnect — Unmount the device on the linux PCIe subsystem.
- PCIe reconnect — Remount the device on the linux PCIe subsystem.
- PCIe reset — Strobe the physical PCIe reset line for 1ms.
- PCIe tandem boot — Perform a PCIe Tandem Boot operation.
- File path: Path to Tandem-boot Bitfile
- Reload — Initialize a reload cycle from FLASH.
- Unprogram — Hold the FPGA in an unprogrammed state.

Functions inherited from *Element* —

- List
- Measure
- Print
- Schedule
- SetAttribute
- Validate

Functions inherited from *JTAGelement* —

- JtagManualConnect

CHAPTER

FIVE

RELEASE NOTES

5.1 Version 0.6.1

Created on 23th January 2022.

- Serenity Z1.2: Add clock and crosspoint switch configs required to use the TCDS2 links to/from the DTH
- PCIe udev rules: Use sysfs rather than dmesg

5.2 Version 0.6.0

Created on 14th November 2021.

Backward-incompatible changes

- Given that the software now supports Z1.2 and Z1.2, the existing Z1.1 bare board configs in `/opt/smash/etc/serenity/configuration/` have been renamed as follows:

```
BareBoard.smash -> BareZ1p1.smash
BareBoard_a35.smash -> BareZ1p1_a35.smash
```

Main improvements:

- LTMs: Added mechanism to disable the global/rail addressing (so that the LTMs can still be brought up/down as a group, but the 3v3 on Serenity 1.2 may be excluded)
- Fireflys: Added support for setting pre-emphasis on > 20G parts, and de-emphasis on <20G parts
  - Also reorganised implementation of Firefly component more widely.

5.3 Version 0.5.2

Created on

- Add support for Serenity Z1.2
  - Bare board config: `/opt/smash/etc/serenity/configuration/BareZ1p2.smash`
- NDM: Fixed bugs in measurements (“I2C no acknowledge received” exceptions)
- Firefly: Fix temperature measurement bug (alternating valid and unphyiscal values), limited transfer size to 4 bytes, and more generally adhere to TWI spec rather than I2C.

5.4 Version 0.5.1

Created on 26th June 2021.

- Updated to uHAL v2.8
- Switch to pybind11 for Python bindings

5.5 Version 0.5.0

Created on 24th March 2021.

Backward-incompatible changes

- Updated `Pluggable` base class to issue warning if socket and plug names do not match. As a result, on Serenity v1.1:
  - Daughter card components must be named `DC0` or `DC1`
  - Firefly modules must be named `Jn:mm` (with the values `n` and `mm` matching the corresponding site)

Minor improvements

- Reorganised implementations of I2C page handling to avoid duplication across different components
- Updated Makefiles and code to support clang-based builds
- Fixed some errors in inline documentation
- Assorted cleanup, including: moving some constants out of global scope; removing dependence of core library on uHAL

5.6 Version 0.4.0

Created on 28th January 2021.

- Fixed bug in udev 'remove' rule
  - Previously this rule was not run when a device was powered off, and so corresponding device file symlinks were not deleted.
- Reorganised Firefly components
  - Merged `FireflyBi`, `FireflyRx` and `FireflyTx` classes into single `Firefly` class
  - Added support for newer parts
- Updated method for connecting to multiple FPGAs in daisy chain
  - For daisy chain: `JTAGmux "Connect All Targets"`
  - For single target: `JTAGmux "Connect Default Target"`
- Added measurements for Si5345 clock synthesisers
- Updated documentation (auto-generated by new tooling)
- Updated internals to use C++11 features and libraries

5.7 Version 0.3.8

Created on 28th January 2021.

- Extended deprecation date for old Serenity Artix JTAG interface (used in v0.2.x of Artix FW) to 1st July 2021.

5.8 Version 0.3.7

Created on 18th September 2020.

- Added FTDI and SEMA components.

  Note: This involved adding the Artix FPGA to the base board config files, and as a result of this the following line will no longer work:

  ```
  Run /opt/smash/etc/serenity/base/atca/*.smash
  ```

  Instead, that should be replaced by one of the following config files:

  - For almost all boards: `Run /opt/smash/etc/serenity/configuration/BareBoard.smash`
  - For those with an A35 Artix (only the first few boards): `Run /opt/smash/etc/serenity/configuration/BareBoard_a35.smash`

- Added config for Serenity VU7P daughter cards: `serenity/DaughterCards/VU7P.smash`
- Shortened names of existing Serenity daughter card config files:
  - `Imperial_KU115.smash` becomes `KU115.smash`
  - `KIT_KU15P.smash` becomes `KU15P.smash`

5.9 Version 0.3.6

Created on 9th July 2020.

- Extended deprecation date for old Serenity Artix JTAG interface to 1st October.

5.10 Version 0.3.5

Created on 13th June 2020.

- Updated code & makefiles to add support for CentOS8
- `XilinxFPGA` component: Added measurements for System Monitor (can be read via either I2C or JTAG)

5.11 Version 0.3.4

Created on 14th May 2020.

- Added option to specify timeout for locking shared mutex
- Fixed bug in shared mutex permissions, so that writeable for all users
- Updated implementation of makefiles and packaging files

5.12 Version 0.3.3

Created on 9th April 2020.

- Updated `XilinxFPGA` and 'JTAG master' code to support new JTAG master interface in v0.3.0 of Serenity Artix FW. Note: This release of SMASH still supports the previous JTAG master interface as well.

5.13 Version 0.3.2

Created on 23rd March 2020.

- Console: Fix bug encountered in pasting (only 1 in 16 characters were read).

5.14 Version 0.3.1

Created on 19th March 2020.

- Updated Serenity Artix address tables for v0.2.2 of Artix FW.

5.15 Version 0.3.0

Created on 7th March 2020.

- Added protection against conflicts between multiple simultaneous SMASH sessions in different processes
- Updated to uHAL version 2.7
- Updated `Smash` class to no longer be a singleton
- Fixed memory leaks in `Smash` and `Element` classes

5.16 Version 0.2.9

Created on 9th February 2020.

- Updated XVC server and console to boost ASIO-based implementation using dedicated worker thread
- Python: Added `Element attributes` method
- Cleaned up makefiles

5.17 Version 0.2.8

Created on 9th February 2020.

- Python: Added `Element` `measurements` method

5.18 Version 0.2.6

Created on 16th December 2019.

- Added ability to directly reset the I2C master firmware block
- Daughter card config files: Fixed I2C address for EEPROM
- Fixed bug causing direct JTAG loading to fail on 2nd attempt in single SMASH session

5.19 Version 0.2.5

Created on 30th November 2019.

- Added ability to reset I2C port expanders
- Fixed FPGA speed and temperature grade in config file for Serenity KU15P daughter-card

5.20 Version 0.2.4

Created on 27th November 2019.

- Added ability to directly program FPGAs (i.e. bypassing an XVC server)
- Fixed temperature and speed grade fields in config file for Serenity KU115 daughter-card
- Created Python bindings for core SMASH functions

5.21 Version 0.2.3

Created on 24th October 2019.

- Renamed some of the example configuration files (`TomOpticalTests.smash` becomes `OpticalTests.smash`)
- Added ‘PCIe utilities’ package: Includes new set of `udev` rules that created the `/dev/serenity_pcie` symlinks
- Added Serenity butler (`serenitybutler`), to replace previous Artix butler (`abutler`) script

5.22 Version 0.2.2

Created on 7th October 2019.

- Add missing file required for configuring the clock synthesisers.
- Main executable: Add `--quiet` and `--verbose` flags.
- Updated core and component classes to use same logging library.

5.23 Version 0.2.1

Created on 20th September 2019.

- Firefly: Corrected units of optical power measurements (issue #17)
- Firefly: Added LOS, LOL and Laser fault alarms as measurements (issue #18)
- Firefly: Removed measurements for registers whose values don't conform to specs (`Part Number`, `Revision Number`, `Serial Number` and `Date Code`) (issue #19)
- LTM4677: Added status & fault log measurements, and functions for clearing those registers (merge request #33)

5.24 Version 0.2.0

Created on 20th August 2019.

- Serenity: Added address tables for v0.2.0 of the Artix FW
- Serenity: Updated base-board SMASH scripts to use address tables for v0.2.0 of Artix FW
- FPGA element: New functions - `PCIe reset`, `Unprogram` and `Reload` - and new measurement - `Config state`
- FPGA `Validate` function: Added checks on power configuration (i.e. voltage/current within Xilinx-specified ranges)

N.B. v0.2.x of the Artix FW should be loaded on boards before updating to this release

---

5.25 Version 0.1.4

Created on 13th July 2019.

- Added `Assert` command
- `PMbus` element: Fixed paging bug
- Added `DummyComponent` class for test purposes
- EEPROMs: Implemented read and write functionality

5.26 Version 0.1.3

Created on 11th July 2019.

- Serenity PCIe disconnect script: Updated so that runs *lspci* multiple times in case of error

5.27 Version 0.1.2

Created on 8th July 2019.

- Added `abutler` script for controlling Serenity Artix TTC FW
- Improved `Print` command output for power-related elements

5.28 Version 0.1.1

Created on 25th June 2019.

- XVC server: Fixed bug in host IP address resolution

5.29 Version 0.1.0

First tag; created 23rd June 2019.

CHAPTER

SIX

VERY ADVANCED USAGE

The following instructions indicate how to reprogram the SERENITY Carrier Card for different FPGA Daughter Card configurations. It is imperative that this is done reponsibly as choosing the wrong voltage configurations could cause **catastrophic** failure of the FPGA. *SMASH* is used to control and configure the hardware. Basic commands are covered within *Useful Commands* and *Advanced Usage*.

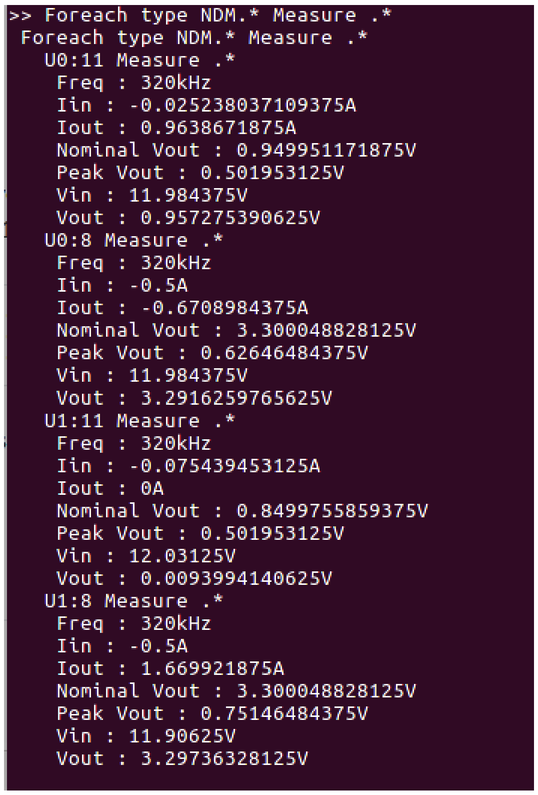
In the following the SMASH commands are given for interactive usage (`smash -i`) then can be included in the command-line with the addition of double quotes - see the introductory SMASH documentation for more details.

6.1 Set the Core (NDM) Voltages:

First determine what the voltages are from the core NDM regulators:

```
Foreach type NDM.* Measure .*
```

The `U0:8` and `U1:8` are for the Fireflys and should be set to 3.3V. The `U0:11` and `U1:11` correspond to the Daughter Card sites `X0` and `X1`. For the KU115 the voltages should look like:



```
>> Foreach type NDM.* Measure .*
 Foreach type NDM.* Measure .*
   U0:11 Measure .*
    Freq : 320kHz
    Iin : -0.025238037109375A
    Iout : 0.9638671875A
    Nominal Vout : 0.949951171875V
    Peak Vout : 0.501953125V
    Vin : 11.984375V
    Vout : 0.957275390625V
   U0:8 Measure .*
    Freq : 320kHz
    Iin : -0.5A
    Iout : -0.6708984375A
    Nominal Vout : 3.300048828125V
    Peak Vout : 0.62646484375V
    Vin : 11.984375V
    Vout : 3.2916259765625V
   U1:11 Measure .*
    Freq : 320kHz
    Iin : -0.075439453125A
    Iout : 0A
    Nominal Vout : 0.8499755859375V
    Peak Vout : 0.501953125V
    Vin : 12.03125V
    Vout : 0.0093994140625V
   U1:8 Measure .*
    Freq : 320kHz
    Iin : -0.5A
    Iout : 1.669921875A
    Nominal Vout : 3.300048828125V
    Peak Vout : 0.75146484375V
    Vin : 11.90625V
    Vout : 3.29736328125V
```

These correspond to the recommended values from the Xilinx data-sheet:

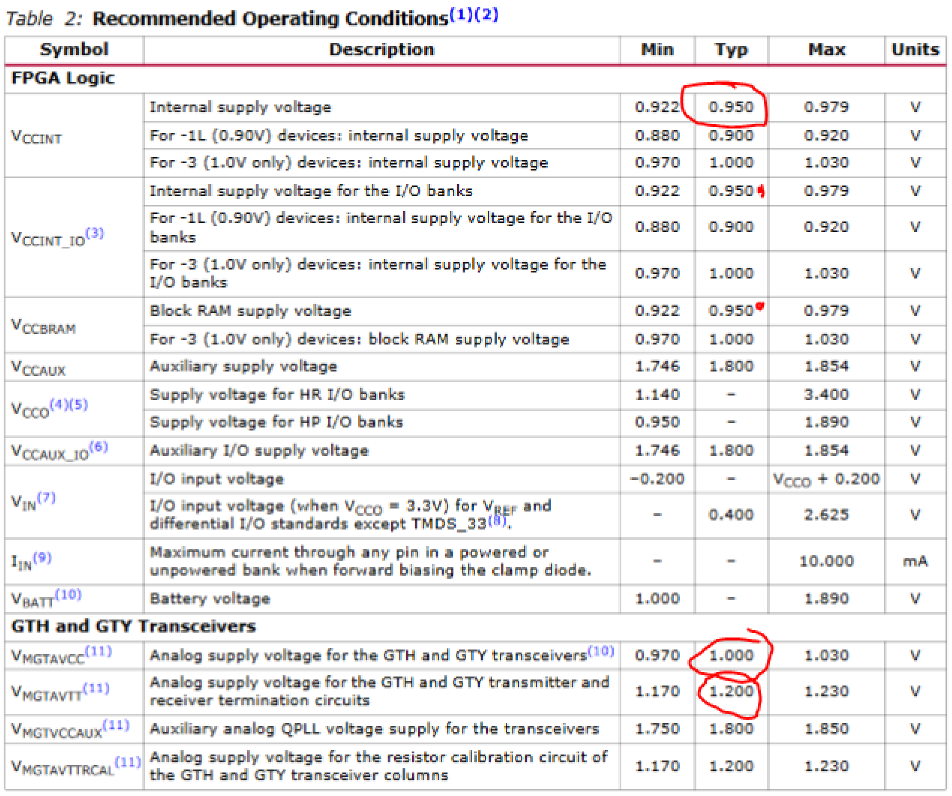
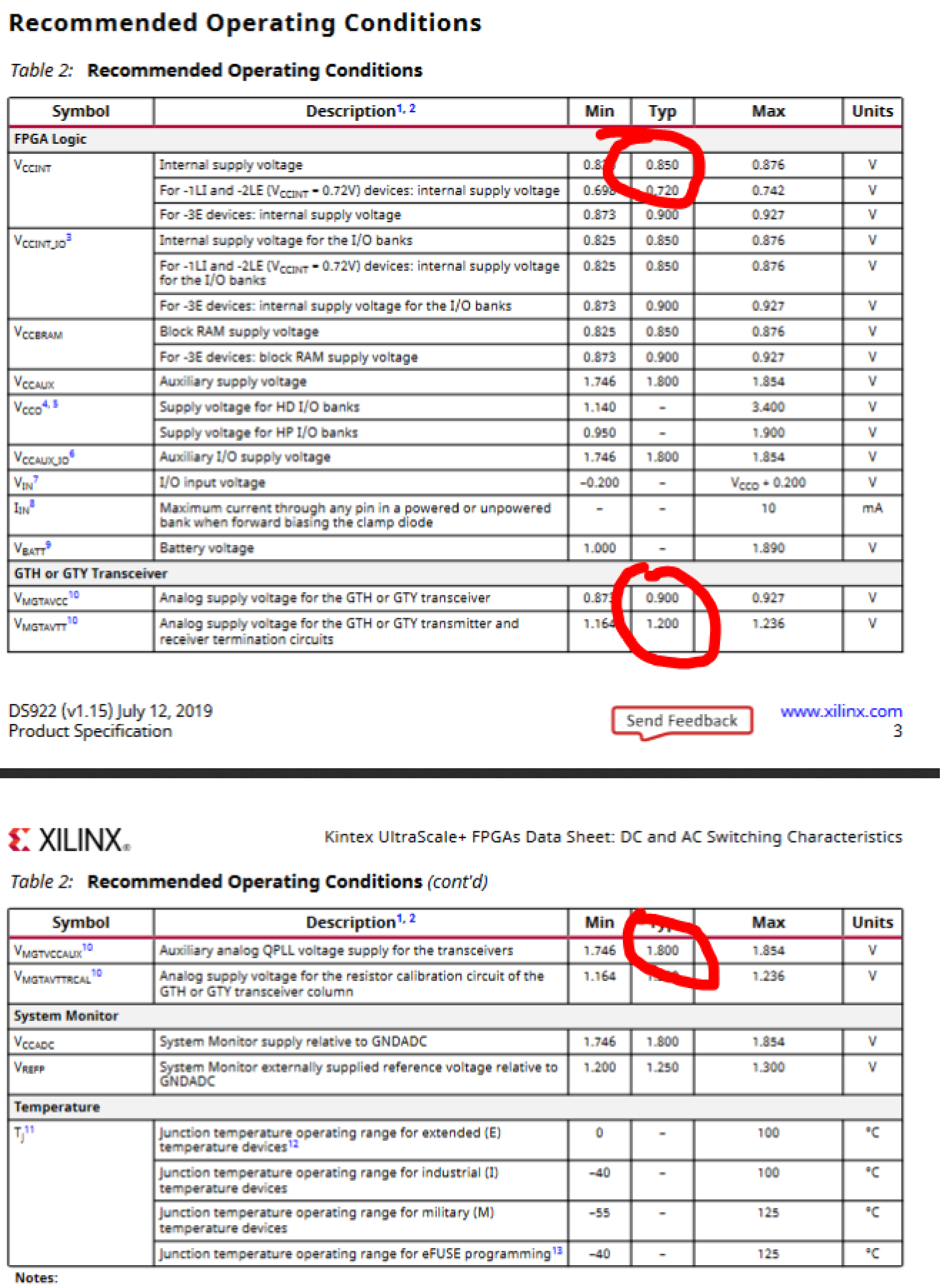


Table 2: **Recommended Operating Conditions**[1][2]

Symbol	Description	Min	Typ	Max	Units
FPGA Logic					
V_{CCINT}	Internal supply voltage	0.922	0.950	0.979	V
	For -1L (0.90V) devices: internal supply voltage	0.880	0.900	0.920	V
	For -3 (1.0V only) devices: internal supply voltage	0.970	1.000	1.030	V
V_{CCINT_IO}[3]	Internal supply voltage for the I/O banks	0.922	0.950	0.979	V
	For -1L (0.90V) devices: internal supply voltage for the I/O banks	0.880	0.900	0.920	V
	For -3 (1.0V only) devices: internal supply voltage for the I/O banks	0.970	1.000	1.030	V
V_{CCBRAM}	Block RAM supply voltage	0.922	0.950	0.979	V
	For -3 (1.0V only) devices: block RAM supply voltage	0.970	1.000	1.030	V
V_{CCAUX}	Auxiliary supply voltage	1.746	1.800	1.854	V
V_{CCO}[4][5]	Supply voltage for HR I/O banks	1.140	–	3.400	V
	Supply voltage for HP I/O banks	0.950	–	1.890	V
V_{CCAUX_IO}[6]	Auxiliary I/O supply voltage	1.746	1.800	1.854	V
V_{IN}[7]	I/O input voltage	–0.200	–	V_{CCO} + 0.200	V
	I/O input voltage (when V_{CCO} = 3.3V) for V_{REF} and differential I/O standards except TMDS_33[8].	–	0.400	2.625	V
I_{IN}[9]	Maximum current through any pin in a powered or unpowered bank when forward biasing the clamp diode.	–	–	10.000	mA
V_{BATT}[10]	Battery voltage	1.000	–	1.890	V
GTH and GTY Transceivers					
$V_{MGTAVCC}$[11]	Analog supply voltage for the GTH and GTY transceivers[10]	0.970	1.000	1.030	V
$V_{MGTAVTT}$[11]	Analog supply voltage for the GTH and GTY transmitter and receiver termination circuits	1.170	1.200	1.230	V
$V_{MGTVCCAUX}$[11]	Auxiliary analog QPLL voltage supply for the transceivers	1.750	1.800	1.850	V
$V_{MGTAVTTRCAL}$[11]	Analog supply voltage for the resistor calibration circuit of the GTH and GTY transceiver columns	1.170	1.200	1.230	V

For the KU15P the main FPGA voltage should be 0.85V, as stated here:



Recommended Operating Conditions

Table 2: **Recommended Operating Conditions**

Symbol	Description[1, 2]	Min	Typ	Max	Units
FPGA Logic					
V_{CCINT}	Internal supply voltage	[illegible]	0.850	0.876	V
	For -1LI and -2LE (V_{CCINT} = 0.72V) devices: internal supply voltage	[illegible]	0.720	0.742	V
	For -3E devices: internal supply voltage	0.873	0.900	0.927	V
V_{CCINT_IO}[3]	Internal supply voltage for the I/O banks	0.825	0.850	0.876	V
	For -1LI and -2LE (V_{CCINT} = 0.72V) devices: internal supply voltage for the I/O banks	0.825	0.850	0.876	V
	For -3E devices: internal supply voltage for the I/O banks	0.873	0.900	0.927	V
V_{CCBRAM}	Block RAM supply voltage	0.825	0.850	0.876	V
	For -3E devices: block RAM supply voltage	0.873	0.900	0.927	V
V_{CCAUX}	Auxiliary supply voltage	1.746	1.800	1.854	V
V_{CCO}[4, 5]	Supply voltage for HD I/O banks	1.140	–	3.400	V
	Supply voltage for HP I/O banks	0.950	–	1.900	V
V_{CCAUX_IO}[6]	Auxiliary I/O supply voltage	1.746	1.800	1.854	V
V_{IN}[7]	I/O input voltage	–0.200	–	V_{CCO} + 0.200	V
I_{IN}[8]	Maximum current through any pin in a powered or unpowered bank when forward biasing the clamp diode	–	–	10	mA
V_{BATT}[9]	Battery voltage	1.000	–	1.890	V
GTH or GTY Transceiver					
$V_{MGTAVCC}$[10]	Analog supply voltage for the GTH or GTY transceiver	[illegible]	0.900	0.927	V
$V_{MGTAVTT}$[10]	Analog supply voltage for the GTH or GTY transmitter and receiver termination circuits	[illegible]	1.200	1.236	V

DS922 (v1.15) July 12, 2019
Product Specification

Send Feedback

www.xilinx.com
3

XILINX. Kintex UltraScale+ FPGAs Data Sheet: DC and AC Switching Characteristics

Table 2: **Recommended Operating Conditions** *(cont'd)*

Symbol	Description[1, 2]	Min	Typ	Max	Units
$V_{MGTVCCAUX}$[10]	Auxiliary analog QPLL voltage supply for the transceivers	1.746	1.800	1.854	V
$V_{MGTAVTTRCAL}$[10]	Analog supply voltage for the resistor calibration circuit of the GTH or GTY transceiver column	1.164	[illegible]	1.236	V
System Monitor					
V_{CCADC}	System Monitor supply relative to GNDADC	1.746	1.800	1.854	V
V_{REFP}	System Monitor externally supplied reference voltage relative to GNDADC	1.200	1.250	1.300	V
Temperature					
T_j[11]	Junction temperature operating range for extended (E) temperature devices[12]	0	–	100	°C
	Junction temperature operating range for industrial (I) temperature devices	–40	–	100	°C
	Junction temperature operating range for military (M) temperature devices	–55	–	125	°C
	Junction temperature operating range for eFUSE programming[13]	–40	–	125	°C

Notes:

If the voltage is incorrect then these can be set (e.g. for the X1 site) and then stored:

```
U1:11 Vout 0.95V
U1:11 "Store to NVM"
```

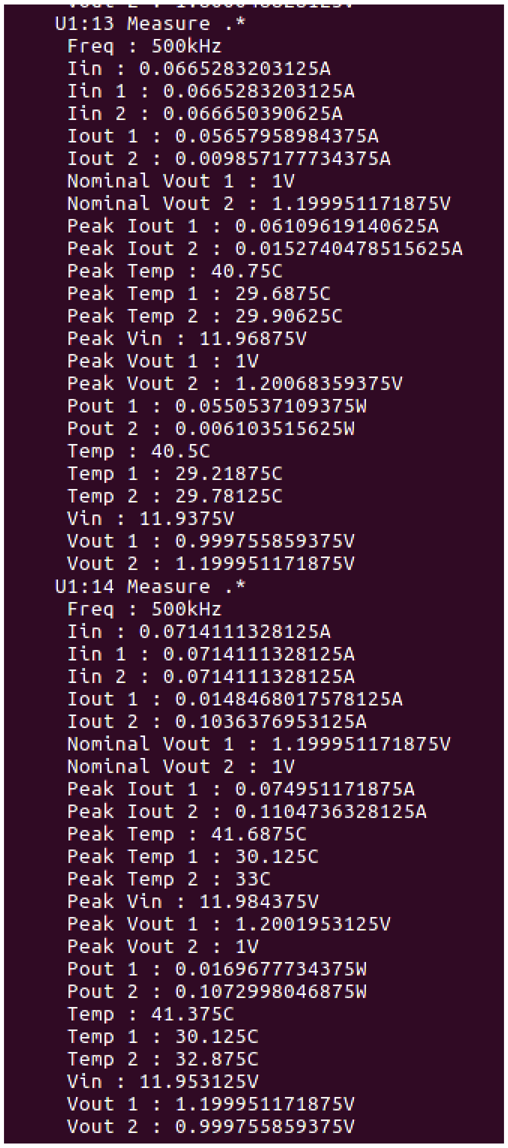
Failure to execute the last command will lose the adjusted value if the board is power cycled (independent of DC power cycle).

6.2 Set the LTM Voltages:

First again determine what the voltages are currently set to:

```
Foreach type LTM.* Measure .*
```

The correct voltage configuration for a KU115 FPGA should look like the following (if the X1 site is powered on):



```
U1:13 Measure .*
 Freq : 500kHz
 Iin : 0.0665283203125A
 Iin 1 : 0.0665283203125A
 Iin 2 : 0.066650390625A
 Iout 1 : 0.05657958984375A
 Iout 2 : 0.009857177734375A
 Nominal Vout 1 : 1V
 Nominal Vout 2 : 1.199951171875V
 Peak Iout 1 : 0.06109619140625A
 Peak Iout 2 : 0.0152740478515625A
 Peak Temp : 40.75C
 Peak Temp 1 : 29.6875C
 Peak Temp 2 : 29.90625C
 Peak Vin : 11.96875V
 Peak Vout 1 : 1V
 Peak Vout 2 : 1.20068359375V
 Pout 1 : 0.0550537109375W
 Pout 2 : 0.006103515625W
 Temp : 40.5C
 Temp 1 : 29.21875C
 Temp 2 : 29.78125C
 Vin : 11.9375V
 Vout 1 : 0.999755859375V
 Vout 2 : 1.199951171875V
U1:14 Measure .*
 Freq : 500kHz
 Iin : 0.0714111328125A
 Iin 1 : 0.0714111328125A
 Iin 2 : 0.0714111328125A
 Iout 1 : 0.0148468017578125A
 Iout 2 : 0.1036376953125A
 Nominal Vout 1 : 1.199951171875V
 Nominal Vout 2 : 1V
 Peak Iout 1 : 0.074951171875A
 Peak Iout 2 : 0.1104736328125A
 Peak Temp : 41.6875C
 Peak Temp 1 : 30.125C
 Peak Temp 2 : 33C
 Peak Vin : 11.984375V
 Peak Vout 1 : 1.2001953125V
 Peak Vout 2 : 1V
 Pout 1 : 0.0169677734375W
 Pout 2 : 0.1072998046875W
 Temp : 41.375C
 Temp 1 : 30.125C
 Temp 2 : 32.875C
 Vin : 11.953125V
 Vout 1 : 1.199951171875V
 Vout 2 : 0.999755859375V
```

Note that these values should not be set manually if it is different to the above (and you have a KU115). Instead a full configuration file should be used.

The LTM configuration files can be created using analog's *ltpower-play* software (windows only):

- LTM Configuration (external site)

Already created files can be found below:

Configuration files for the KU115:

- Project configuration file: KU115 Raw Configuration File

- Exported text: KU115 ASCII Export

And configuration files for the KU15P:

- Project configuration file: KU15P Raw Configuration File
- Exported text: KU15P ASCII Export

The configuration file can be loaded for a single daughter site (`X0`, DC0, IC0, South) with the following SMASH interactive command:

For the KU115 FPGA:

```
X0:LTgroup Configure /path/to/LTM4677-KU115-V1.txt
```

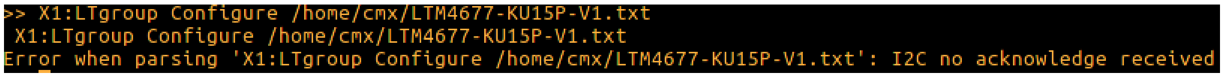
Or for the KU15P:

```
X0:LTgroup Configure /path/to/LTM4677-KU15P-V1.txt
```

Equally both daughter card sites can be programmed together:

```
Foreach type LTgroup Configure [filename]
```

If the wrong configuration file type is being loaded (either a proj file or an incompatible text file) then the error will indicate that I2C has failed and no acknowledge is received with something like:



```
>> X1:LTgroup Configure /home/cmx/LTM4677-KU15P-V1.txt
 X1:LTgroup Configure /home/cmx/LTM4677-KU15P-V1.txt
Error when parsing 'X1:LTgroup Configure /home/cmx/LTM4677-KU15P-V1.txt': I2C no acknowledge received
```

Finally the LTM values should be stored to NVM in order to make them persistent during power cycling of the carrier card (independent of the DC site):

```
Foreach type LTM.* "Store to NVM"
```

CHAPTER

SEVEN

STATUS OF THIS DOCUMENT

First steps to using the SERENITY prototype system.

- Rev 1.0: First version - Alex Howard, 9th April 2019.
- Rev 2.0: First Commands - Greg Iles, June 2019.
- Rev 3.0: Advanced Commands - Alex Howard, 2nd July 2019.