D-TACQ Solutions Ltd

D-TACQ 4G User Guide

Prepared By: Peter Milne

User Guide Covers modular appliance range:

ACQ1001

ACQ1002

ACQ1014

ACQ2106

KMCU

Fitted with analog modules including

ACQ420FMC, ACQ430FMC, ACQ480FMC,

ACQ423ELF, ACQ424ELF, ACQ425ELF, ACQ427ELF, ACQ435ELF, ACQ482ELF

AO420FMC, AO424ELF, DIO432, DIO482

Please DO NOT Print this Guide!

It's best viewed on a larger screen. Some diagrams have considerable detail that can be seen at higher zoom.

Navigation: the document has a full set of PDF bookmarks. Use the “Bookmarks” feature of Adobe Reader to navigate.

Rev	***Date***	***Description***
1	Feb 1 2014	Intial.
2	Mar 24 2014	Add site and css logs
3	Apr 6 2014	Describe ACQ1001/CELF2
4	May 6 2014	Add MDSshell
5	July 18 2014	Package definition. Describe Transients
7	Sept 21 2014	Fault monitor
8	November 3 2014	ACQ425 Transient, Streaming, CSS Install.
9	February 28 2015	Add ACQ2106
10	May 25 2015	Show boot time customisation options
11	June 19 2015	Document AWG, SYNC.
12,13		Minor changes.
14		ACQ480
15		Minor
16		Add transient capture detail
17/18		LIVETOP, AWG, server sockets ref updated
21		DROP segments. Describe gpg, delay_trigger
23		Voltage coding
24		Overview of operating modes
29		play0 command defined
31		trim ancient stuff. Add new .ovl package def.
32		TIGA
33		Updated SPAD controls.
34	15 Dec 2020	Simpler MGTDRAM

Table of Contents

Copyright and Attribution.

This document and D-TACQ Software comprising platform Linux port, Linux kernel modules and most applications are released under GNU GPL/FDL:

Document:

Software:

1 Introduction

1.1 What is D-TACQ 4G?

D-TACQ 4G is a range of intelligent carriers and modular data acquisition cards, configured together to make a networked data acquisition appliance.

Generically, an “ACQ400 System” comprises a CARRIER and one or more MODULES. We refer to “ACQ400” as a shorthand for such a system.

The carriers include

ACQ1001 – single site intelligent “shortbread box” unit.

ACQ1002R and ACQ1002S – dual site “Rack” and “Stack” variants

ACQ2106 – 2 U x 19” computer unit with 6 sites and MGT comms links.

KMCU-Z30 : AMC for physics module, 2 x ELF sites at rear.

and the module range includes

ACQ420FMC – 4 x 2MSPS simultaneous ADC module.

ACQ425ELF – 16 x 2MSPS simultaneous SAR ADC module.

ACQ424ELF-32 -32 x 1 MSPS simultaneous SAR ADC module.

ACQ435ELF – 32 x 128kSPS, 24 bit simultaneous delta-sigma ADC module.

ACQ480FMC – 8 x 80MSPS

AO424ELF-32 – 32 x 500kSPS, 16 bit DAC module.

The networked system is a highly configurable computer supporting from 4 to 192 channels per chassis.

This document describes how to configure and use the system.

1.2 Intended Audience

- End Users
- End User developers.

** This Document has 136 Pages. Yes, it's far too long. But, control of your ACQ400 system can be VERY SIMPLE. Most of the time, D-TACQ can supply a customised script to make it do what you want. Please don't hesitate to ask us. Don't Panic! **

1.3 Scope

- Describes software system operation.
- For hardware specifications, see data sheets 1.5.3
- For hardware installation, see hardware installation guides. 1.5.4

1.4 Glossary

- *Carrier* : a motherboard with embedded computer capable of supporting a payload of one or more modules.
- *Modules*: analog input module with specific analog functionality.
- FMC – VITA57 standard for FPGA-compatible modules
- ELF – D-TACQ extension to FMC standard.
- SOC – System on Chip
- ZYNQ – Xilinx(r) SOC device comprising a hard ARM core and FPGA logic.
- MGT – Multi-Gigabit Transceiver.
- STL - State Transition List: ascii list of time,state values, forming a compact definition of a digital pattern.

1.5 References

1.5.1 EPICS:

Experimental Physics and Industrial Control System
http://www.aps.anl.gov/epics/

1.5.2 CSS

Control System Studio : EPICS GUI Client CSS

ACQ400 OPI Set: https://github.com/D-TACQ/ACQ400CSS

1.5.3 D-TACQ Data Sheets

Data Sheets

1.5.4 Installation Guide

ACQ1001_Installation_Guide

ACQ2106_Installation_Guide

1.5.5 HAPI : Host Application Python Interface

https://github.com/D-TACQ/acq400_hapi : library

https://github.com/D-TACQ/acq400_hapi_tests : example client applications.

1.5.6 MDSplus

Experimental data system: http://www.mdsplus.org

1.6 Notation

- **`command`** : indicates name of a program (command)
- `preformatted text` : literal input or output from terminal session.
- *Defined Term* : some term or acronym specific to this domain (perhaps referenced in the glossary)

2 Quickstart Guide

This is very much dependent on the combination of modules that is delivered.

The Easy Way – Ask D-TACQ!

Each system will be delivered with a custom configuration script, run the script and see data. If you can describe your requirement to D-TACQ, we can provide a setup that meets the requirement with minimal extra configuration.

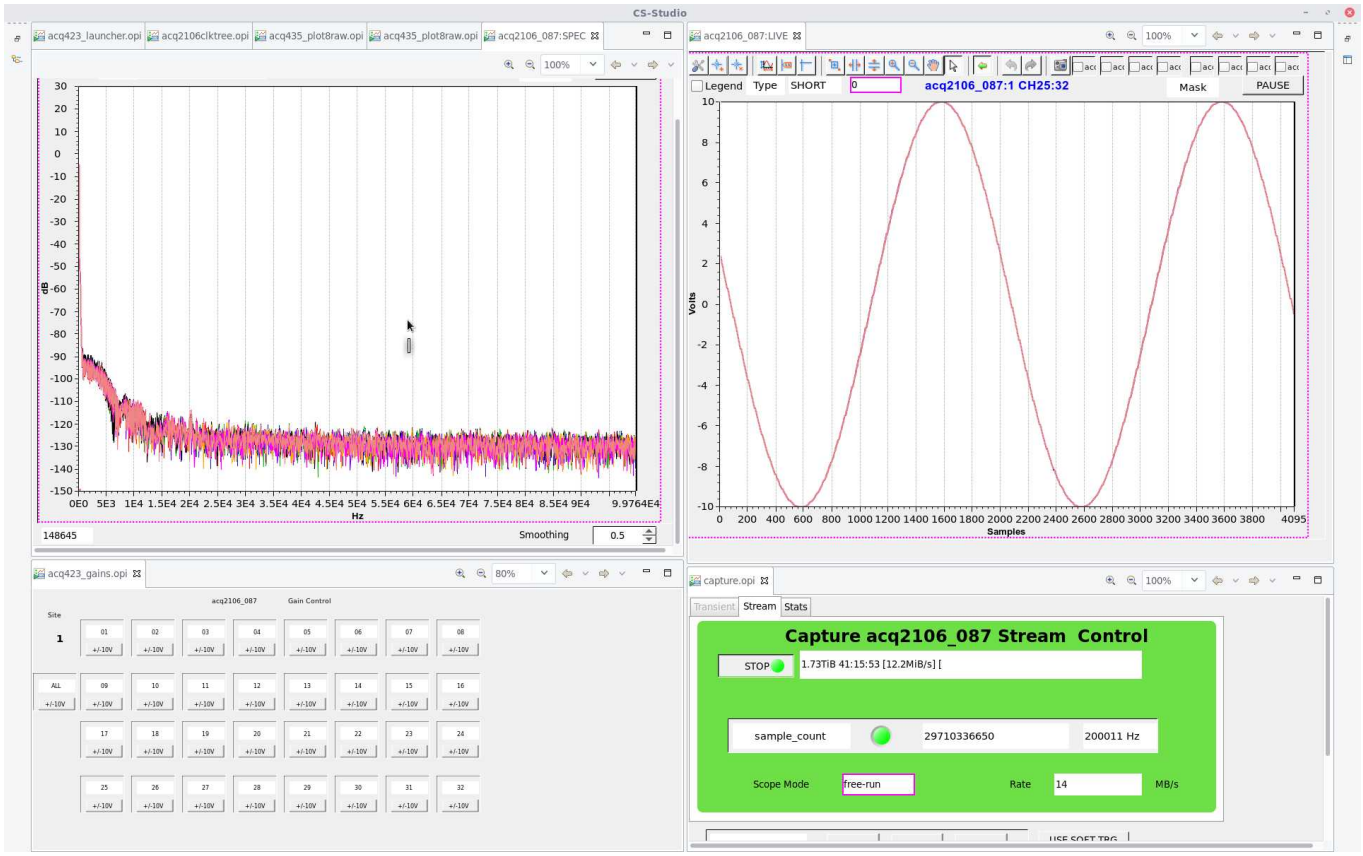
2.1 Turnkey capture with live data display

Use the CSS GUI 1.5.2 . Please be sure to follow the steps in README.md,

and also look at acq1001_acq430_quickstart.pdf, steps are relevant regardless of carrier or payload.

First run the LAUNCHER, then pop the CAPTURE.opi to start a streaming capture, then use a Site 1 LIVE PLOT to view the data immediately.

The data is “streaming to nowhere”, but we can show snapshot waveforms and view scalar values.



2.2 What Operating Mode ?.

2.2.1 First, List the Operating Parameters:

(assuming an ADC application)

INPUT:

- Number of Channels : NCHAN 4..192, word-size WS (2 or 4 byte)
- Sample Rate: SR (10k .. 80M)
- Duration Of Capture, finite: TSHOT or infinity
- Time between Captures : TINTERVAL
- Burst or Continuous Data : burst length BLEN

OUTPUT DATA calculation:

- Data Rate in Mbytes/Sec MBPS : DATA_RATE = NCHAN*WS*SR
- Data Volume in MB : DATA_VOL = TSHOT * DATA_RATE

Then, decide : ***SHOT*** or ***STREAM***

2.2.2 SHOT

Capture data at high rate for a period TSHOT, stop, and offload.

- Check: does DATA_VOL fit available DRAM?.
  - If not, on ACQ2106, consider fitting a bigger RAM #27
  - If not, consider streaming to network or fiber-optic #2.2.3
- Check: does TINTERVAL allow time for postprocessing, including offload.
  - If not, consider streaming in Burst Mode.
- If the data is not timewise continuous, enable Burst Mode. #10.5
- Is this is a PRE/POST capture : system may run for a long time capturing data in PRE buffer before a trigger causes transition to POST, stop, and upload.
  - “Fault Monitor” Application #20, low subrate rate scalar values are published to the network as EPICS AI records.
- A typical SHOT system captures 512MB of data in about 8s, with TINTERVAL > 30s.
- ONE SHOT TRANSIENT #10.4 #21

2.2.3 STREAM

Streaming: data is acquired continuously with no preset limit.

- *Stream to nowhere*: Data is DISCARDED, however low rate scalar point values are published to the network, eg as EPICS AI records, as well as possibly snapshot waveforms
  - The snapshot waveforms may be synchronized with an external event-trigger, to form a live Scope Display. Each trigger updates a set of EPICS WF records, that are published on the network and may be archived, limits of length and rate (4K, 5Hz, typical)
  - Live spectrum display also supported. #2.1
- Stream to Ethernet: (all units) #10.3
  - This is extremely simple to configure
  - DATA_RATE limited to 30MBPS.
  - Burst Mode is an effective way to resynchronize with a repeating external trigger, and can reduce the data rate.
  - External timing edges, configured as “EVENT_TRIGGERS” may be used to embed timing markers in the data stream
- Stream to Fiber-Optic (ACQ2106) #23
  - Requires a suitable HOSTPC system, with fiber-optic host bus adapter.
  - High throughput, uses large buffers:
    - DATA_RATE to 400MBPS/link, up to 4 links per adapter.
  - Low Latency, transfers single samples, control optimised.

2.2.4 START TRIGGER

A single digital edge starts the capture. This could be

- Soft Trigger
- External trigger, from front panel TRG input or SYNC-IN
- A delayed trigger from GPG #12.3.1

2.2.5 EVENT TRIGGER

One or more digital edges input after the shot has started. These could be used to trigger BURST captures, to trigger the transition from PRE to POST or simply to embed a timing point ES in the data stream.

- Input from front panel TRG or SYNC
- Generated by GPG #12.3.3
- Generated by a DSP function. #13

3 Overview

3.1 ZYNQ SOC

The ACQ2006 carrier features the Xilinx ZYNQ System on Chip. This provides unprecedented system integration, combining

- Dual-Core ARM Cortex A9 processor
- FPGA fabric
- Dual Ethernet and DDRAM controller.

3.2 ACQ1001/ACQ1002/ACQ1014 Appliance

The Appliance combines the Zynq SOC in a system that auto-configures the set of modules currently in system, and powers up ready to run. The system may be configured to run turnkey (captures data from power up), or it can be configured using standard TCP/IP networking. All control is via simple text based, script-able commands, no device drivers required. Data is transferred efficiently in binary format, and a large number of transfer mechanisms are available. The Appliance features an embedded 1.5.1EPICS IOC, this presents all the Process Variables, for control, monitoring and data on the network. The IOC is responsible for a lot of system logic, for example computing a 64 bit sample count and outputing a sample frequency. These PV's can be picked up by a remote EPICS client application such as 1.5.2 CSS. D-TACQ provides a full function CSS GUI interface to the appliance. Even users with no interest in EPICS or the embedded IOC can still make use of IOC generated values through the unified command interface.

3.3 ACQ2106 Appliance

This appliance has the same functionality as ACQ1001, but with a larger Zynq SOC, 6 ELF sites and the addition of up to 4 MGT links. The links are physically connected by a “comms mezzanine” CM, typically MGT482, with 4 x SFP+ sites. Another “comms mezzanine” is MGTDRAM-8, with 8 GB of capture DRAM, but no external links.

3.4 Module Sites

The appliances offer from 1 (ACQ1001) to 6 (ACQ2106) FMC/ELF module sites. The sites are numbered 1..6. In addition, there is a System Controller SC function nominally at “Site 0”, and possibly other system-specific optional functions at logical sites 7-20.

3.5 Unified Command Interface

The Appliance uses the concept of a “Knob” or controllable parameter. In the software, a Knob might be a read-only file (constant or variable value), a writeable file or even an executable command, including a query on an EPICS PV. The system presents a consistent “getter”, “setter” interface that completely hides the implementation detail.

3.6 Unified Data Interface

The FPGA hardware includes a powerful “aggregator” function, that can combine all the data from any set of modules into a single stream. The data is presented to remote (and local) clients on a single TCP socket from “Site 0” at PORT 4210

```
nc acq2006_006 4210 > big-raw-file
```

3.7 Remote Command interface.

Each site is supported by its own TCP server socket at PORT 422+SITE.

- PORT 4220 : Site0 Commands
- PORT 4221 : Site1 Commands ...
- PORT 4226 : Site6 Commands.

It's simple to connect from a remote host and execute sets and queries on the parameters. All that your HOST OS and programming language has to do is support making a TCP/IP client socket connection. In the examples below, we use a Linux Host and the **`netcat (nc)`** program to make the connection, then we run the sets/queries by typing them. An actual remote program would use any programming language to automate the sequence. D-TACQ recommends using **`expect`** (TCL). D-TACQ provides a comprehensive set of remote control client scripts HAPI.

The remote command interface includes

- **`help`** : lists the available knobs
- supports wildcards for queries
- **`prompt on`** : enables a prompt. This makes dialogs more clear for the human, maybe more difficult for a computer interaction.
- Set- values are range checked. Parameters may be read-only, and attempts to set them are rejected with an error.

So a typical remote control sequence would be :

- connect to each module site in turn, configure parameters

- connect to the system controller site and start a capture.

Sounds complicated?. It isn't really, the commands are high level. And, in a

lot of cases, we can supply a single command to cover an entire configuration.

3.7.1 Ease of Use Vs Security.

Connecting a socket to a well-known port, send a string to execute a command – this is very easy to do, but is it secure?. Well, firstly, it is relatively secure because the server program only allows access to the knobs in the directory. This should be sufficient for a basic level of security.

If a higher level of security is required, clients can be forced to use secure shell and authentication as follows:

- use **`tcp-wrappers`** to force **`inetd`** to listen to `localhost` only.

- external clients can route information to `localhost` via an **`ssh`** tunnel.

3.8 Scripting Local Commands.

All the knobs available to the site servers are available to local scripts using the **`set.site`** command. There are two styles of usage, pick the one that is more convenient. The “Stream of command style” may be a little faster to execute.

3.8.1 Single command Style

```
set.site 1 trg 1,1,1
set.site 1 clkdiv 100
set.site 1 hi_res_mode 1

set.site 0 run0 1
set.site 0 soft_trigger
```

3.8.2 Stream of commands style

```
set.site 1 << EOF
trg 1,1,1
clkdiv 100
hi_res_mode 1
EOF

set.site 0 <<EOF
run 0 1
soft_trigger
EOF
```

4 System software structure

4.1 Bootloader in Flash

The system boots the Xilinx First Stage Bootloader FSBL and regular boot-loader u-boot from soldered QSPI flash. The u-boot environment has important information including baseboard model type, serial number and Ethernet MAC address is also stored in the QSPI. The QSPI is considered to be ROM from the user perspective.

4.2 Embedded Linux system loaded from SD card

The bootloader then selects Linux kernel initial ramdisk and device-tree images from an internal SD card. The images are unzipped into RAM. The device tree enables model specific hardware customisation.

4.3 User Space boot

Once the kernel has booted, it hands over to a boot program contained in the initial ramdisk as normal. First, the boot program does standard things like load the network – the initial ramdisk is a fixed, limited size system. Finally, the boot loaded hands over to the package system. At this stage, the SD card is available as a mounted file system under /mnt.

4.4 System customization by packages.

D-TACQ uses a very simply package mechanism to complete the system boot. Packages are tarballs that are loaded from /mnt/packages/ onto ramdisk. The tarballs have a numeric prefix, to enforce sequence, and after the tarball has been unpacked, an optional init script is run. Certain standard packages are selected by baseboard model. Early in this phase, a system specific package `5-acq1001*tgz` (in the case of ACQ1001), for example will perform FMC module enumaration. A common package `10-acq420*tgz` uses the enumeration to select an FPGA personality, load an appropriate FPGA image and to load the core FMC device driver. Suqsequent packages provide higher level functionality (eg web server, EPICS IOC). It's common to bring in application customization packages towards the end of the sequence.

A new type of package based on file system images and overlays has been introduced to address the issue of very large packages that would exhaust the RAM.

Packages are described in detail in Package Reference.

Finally, the boot script calls the local boot script:

4.5 Local boot script : /mnt/local/rc.user

This script runs last. It's intended to contain only site-specific customization.

Please note, the entire `/mnt/local/` directory is available for site customization.

4.6 SD Card runtime policy.

D-TACQ considers the ACQ400 system to be a hard real-time system that runs from RAM. The system achieves maximum reliability by utilising the large SD card as read-only memory. Ideally, the system unpacks all the code and data it needs from the SD card at boot time, and doesn't touch it again. That is the most conservative case. It is of course entirely possible to store code and data on the SD card, even during a data capture. The SD card can be used as a simple logging store for low rate variables. However, we ask users to bear in mind that the system will have maximum reliability when operating at high speed, and maximum longevity if the system minimises write activity to the SD card. For intensive data logging, we recommend first and foremost using the Ethernet port, or for isolated systems, fitting a second disk to the USB port.

5 Power Up Guide

5.1 Serial Console

System console access is provided on a micro-usb port. The usb port appears as a console port in the host OS. Connect to the console port using a terminal emulator, 115200 baud, 8 bit No parity. D-TACQ recommends Kermit as the terminal emulator.

D-TACQ makes extensive use of low-power, low cost “Raspberry PI” Linux computers as terminal severs. The server is configured to set the port for each ACQ4xx devices by name, and typing the name opens a direct Kermit connection to the correct device. Contact D-TACQ for details.

5.2 Account

The root password is provided on a printed sheet with your shipment. Please do NOT publish the root password.

5.3 Network IP address

5.3.1 Boots DHCP as default

At boot, the default behaviour of Acq4xx is to request an ip-address on ETH0 using DHCP. This is suitable for large installations as it allows central control, and for small installations perhaps booting in a network with a DHCP server on a router. If you do the latter, please make sure the router has gigabit ethernet ports!.

The MAC address is printed on the case of the unit.

5.3.2 Set Static IP address

If DHCP is not available in your configuration, a static IP address is required.

Specify a static IP address from the root console as follows:

`set.static_ip` PORT IP-ADDRESS # PORT : 0|1 IP-ADDRESS: dotted quad

```
acq1001_004> set.static_ip 0 192.168.1.4
```

This command writes to the file `/mnt/local/network`. The setting is persistent through power off. More complex network scenarios may be configured by editing this file directly.

5.3.3 Failover to a fixed static address

If the unit is set to boot DHCP by default, but no DHCP address is obtained on boot, after a ten second timeout, the unit will set a fixed static IP address.

The fixed address is approximately :

192.168.0. {SERIAL_NUMBER %200 + 1}.

The exact fixed address is can be viewed with this command:

```
acq2106_189>  /usr/local/CARE/show-fallback-ip
MAC ADDRESS 00:21:54:13:00:bd
+++ fallback static ip 192.168.0.189
```

5.4 Time of Day

ACQ2006 / ACQ1001 does NOT include a battery backed clock. The system is likely to be installed for a very long time, and the battery is a point of failure. It is recommended to set the time of day automatically on boot using ntp.

At boot, it's assumed that the ntp server is the same as the dhcp server. If this is not the case, or if you're using static ip, the ntp server can be specified like this:

```
acq1001_004> set.ntpd 2.pool.ntp.org
```

This setting is stored in /mnt/local/ntpd.conf, and persists through power off.

If a more complex ntp scenario is required, exit /mnt/local/ntpd.conf directly

5.5 Embedded Web Pages

When the system is powered up and networked, point a web-browser to the device, and the embedded web pages 7 will be visible.

The web server uses the standard port (Port 80).

5.6 ssh Server

ACQ4xx acts as an ssh server, and it's possible to log in and run commands using ssh.

(on Windows, use WSL, git-bash, Cygwin or PUTTY).

ssh works more smoothly with key-exchange. To set up custom keys, make a copy of `/mnt/packages.opt/15-custom_sshkeys-YYMMDD.tgz` to your local computer. Unpack it, add your public keys, tar it back up with your own site name eg

`15-SITE_sshkeys-YYMMDD.tgz` and deploy to `/mnt/packages/` on every UUT on-site.

Now ssh logins are automatic, no password required. This saves time, especially with automated remote update scripts.

5.7 Site and Data Ports

Are enabled by default. If this is considered a security hazard, use TCPWRAPPERS to control access. One secure mechanism is to set the server data ports to listen to localhost only, then to access them via ssh port-forwarding.

5.8 Emergency Rescue

Break into u-boot at power up (press space-bar inside first 3s).

Create EBREAK environment variable and SAVE it.

```
acq2006-uboot> setenv EBREAK yes
acq2006-uboot> saveenv
acq2006-uboot> boot
```

boot up to emergency prompt.

```
BREAK to emergency shell ^D when done
/bin/sh: can't access tty; job control turned off
/ #  EBREAK / # [   43.766283] random: nonblocking pool is
initialized

/ #  EBREAK / #
/ #  EBREAK / #
/ #  EBREAK / # ls
acq400          home            mnt             sbin            usr
bin             lib             opt             sys             var
dev             linuxrc         proc            tmp
etc             lost+found      root            update_qspi.sh
/ #  EBREAK / #
```

Fix problem. on reboot, you MUST clear the environment:

```
setenv EBREAK
saveenv
boot
```

Procedure valid from release 568 onwards.

5.9 Configuration Backup

Each unit is provided with a custom configuration held in //mnt/local.

We recommend that this should be backed up when the unit is first received, and backup again if any configuration changes are made. A suitable backup command would be:

```
cd BACKUPS
mkdir acq2106_123
scp -r root@acq2106_123:/mnt/local acq2106_123
```

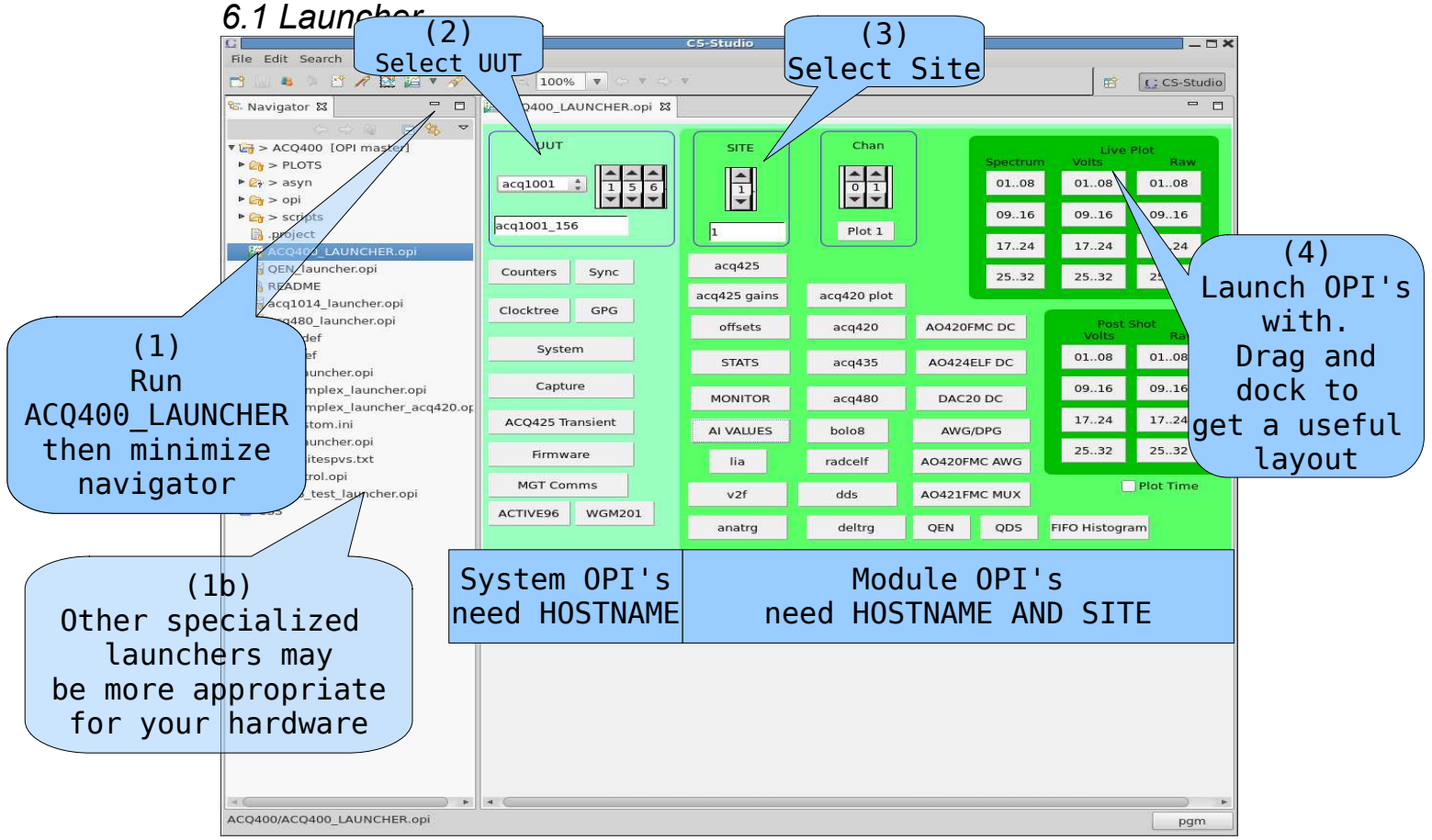
Units are backed up on shipment from D-TACQ, so please contact us should you lose your backup. This instruction is really so that you don't lose any customizations you may make. //mnt/local is NOT disturbed by firmware update.

6 Demonstration GUI

ACQ4xx includes an embedded EPICS IOC. Even if you have no plans to use EPICS, combined with the CSS GUI, it makes a very convenient way to control and use data, and it's recommended at least for initial evaluation.

- Install CSS 1.5.2
- Download D-TACQ CSS Support: ACQ400CSS
  - This is a github page, select “Clone or Download”, download Zip.
- Run CSS and import the project from the Zip.
- **IMPORTANT** : always run everything from the Launcher!
- In edit mode, select acq420_launcher.opi, right-click, open with OPI RUNTIME

6.1 Launcher

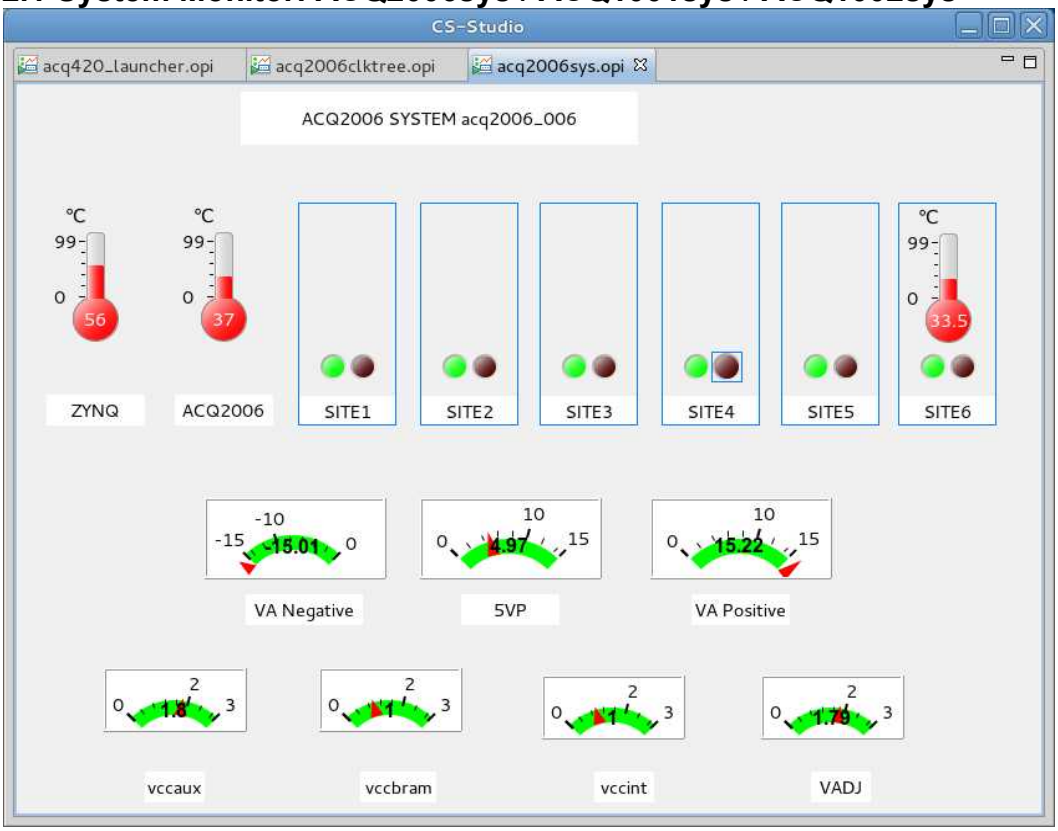


Please note: always refer to the UUT using the hostname, this is independent of any DNS facility on the network.

Please note: OPI's are in continuous development and appearance may vary.

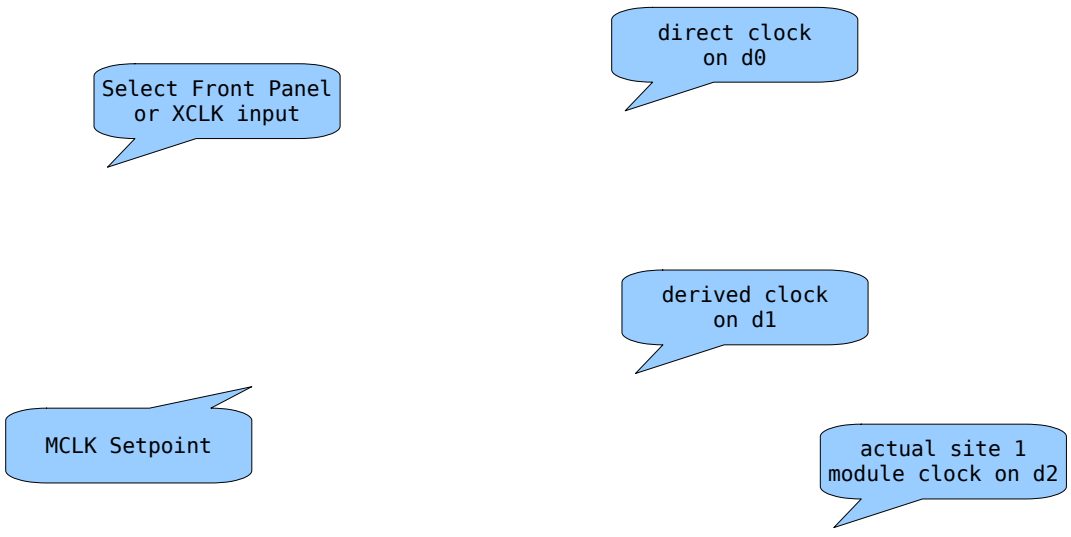
6.2 System OPI's

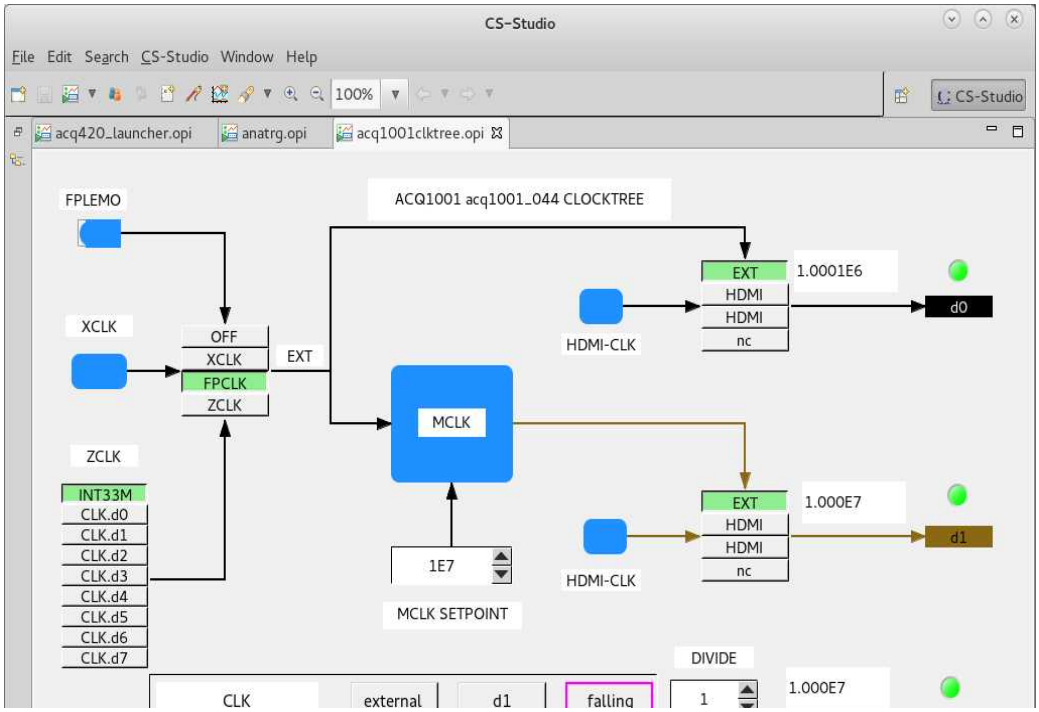
6.2.1 System Monitor: ACQ2006sys / ACQ1001sys / ACQ1002sys



6.2.2 Clocktree

Helps to understand the sample clock setup. eg:





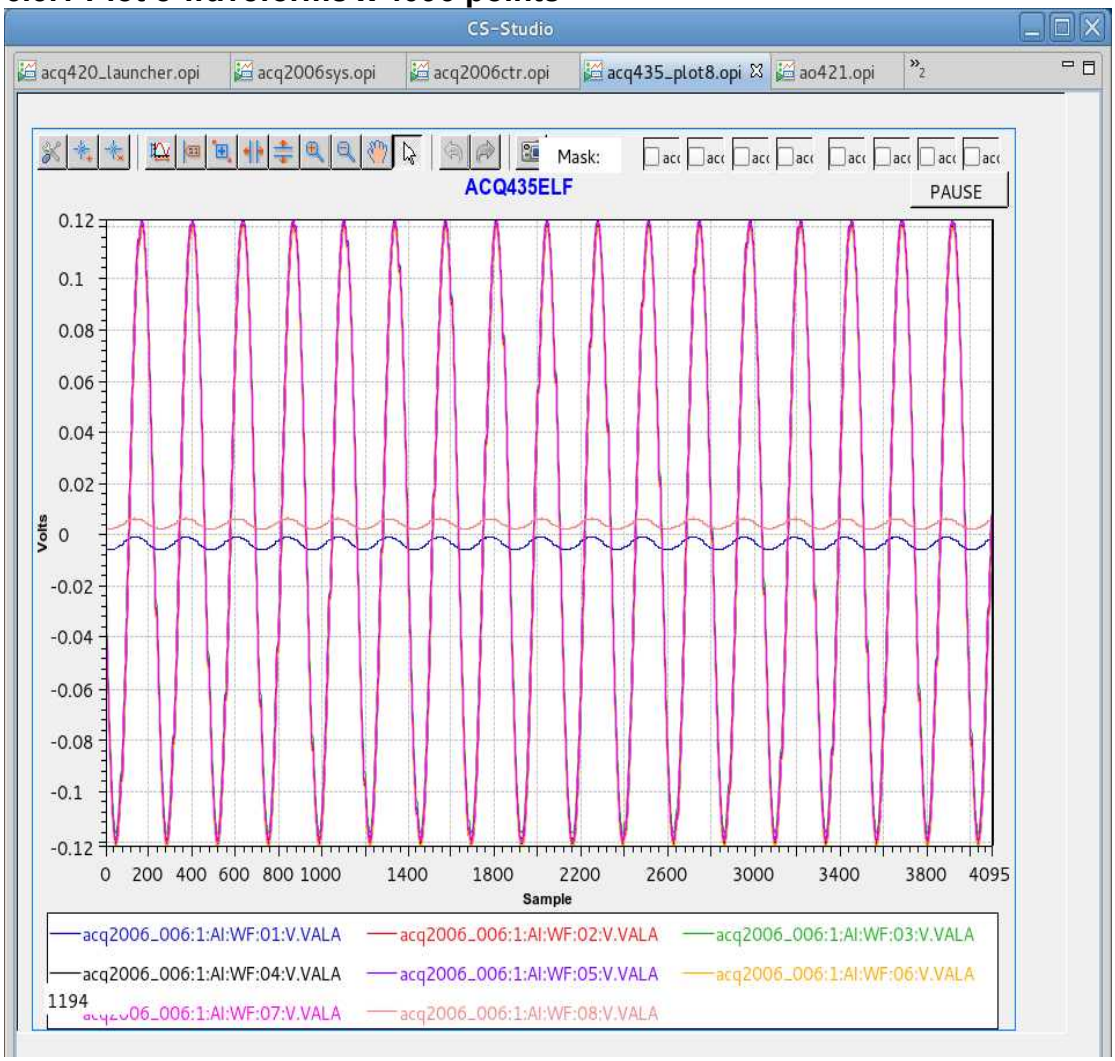
6.2.3 ACQ2006 Counters

A comprehensive set of Clock, Trigger, Event and Sync counts, activity and frequency:


CS-Studio
acq420_launcher.opi
acq2006clktree.opi
acq2006sys.opi
acq2006ctr.opi
acq2006_006 COUNTERS
acq2006_006 CLK
47743389 Hz
10311015 Hz
24613678 Hz
0 Hz
0 Hz
0 Hz
0 Hz
0 Hz
12932169496
10299080413
6696303457
0
0
0
0
0
EXT
MB
S1
S2
S3
S4
S5
S6
acq2006_006 TRG
0 Hz
0
acq2006_006 EVT
acq2006_006 SYN
48094 Hz
48070 Hz
48065 Hz
48064 Hz
13111535
16404061
16415401
16431865

6.3 Module OPI's

6.3.1 Plot 8 waveforms x 4096 points



- Live plot feature, limited to 4096 points.
- cs-studio can plot post shot data to 100K points.
- for longer captures, we recommend more powerful tools such as:
  - MDSplus jScope
  - KST plot.

6.3.2 ACQ435 Control

6.3.2.1 Signals

In the above dialog, please note the 4 rows of “control triplets”. This applies repeatedly to signal selections throughout the ACQ400 product.

The “control triplet” refers to 3 controls:

- Mode: Usually 0 | 1, Enable | Disable, can be more.
- DX : Signal Line, usually selection d0..d7
- Edge: Rising | Falling

This concept appears again at: in general at 8.2.1 and specifically at 12.1, among others.

7 Diagnostic Web Pages

Note on reporting: At times D-TACQ will request a diagnostic report of a page.

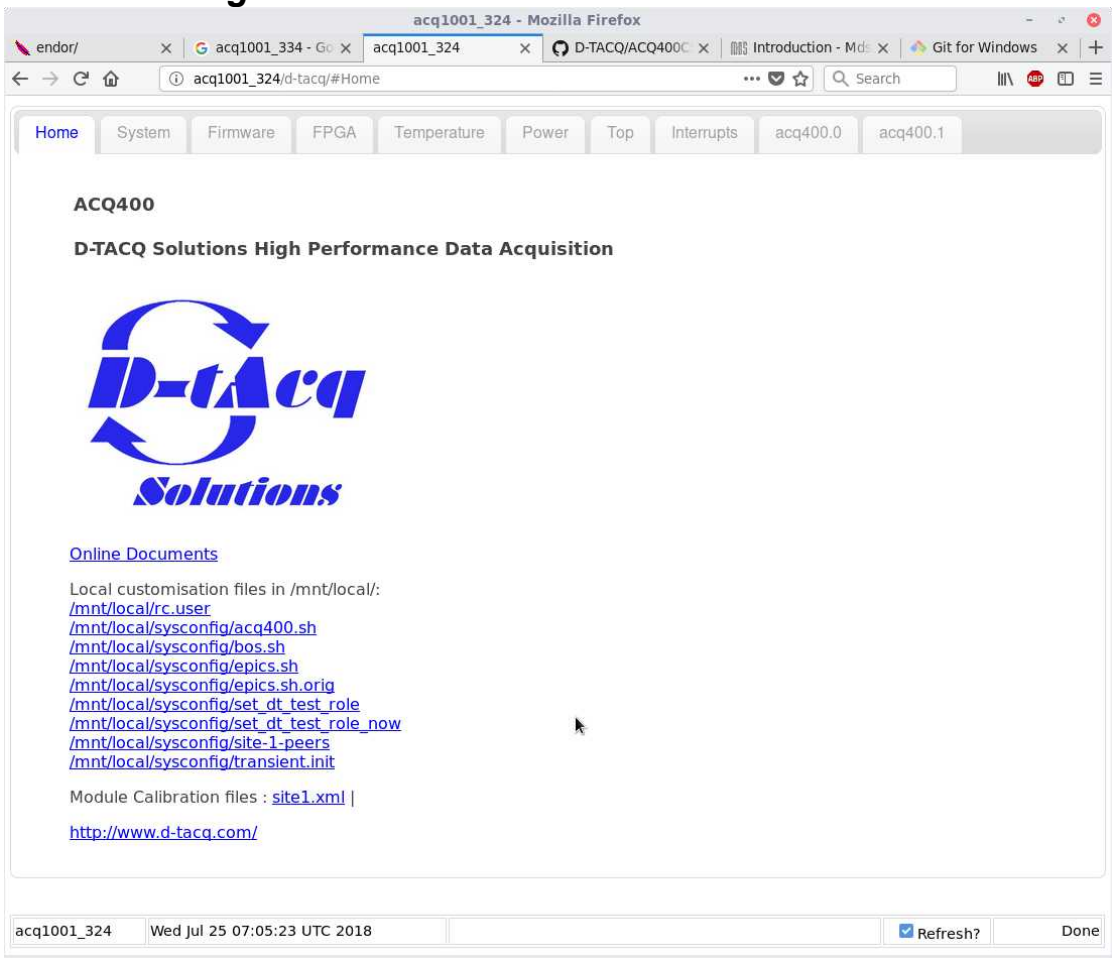
This can be in the form of a screen shot, but we do prefer plain text please.

Copy/Paste plain text from the pages can be difficult because they are dynamic; to do this, please uncheck the [ ] Refresh? Checkbox at the bottom right. It should then be possible to highlight, copy, paste text in the normal way.

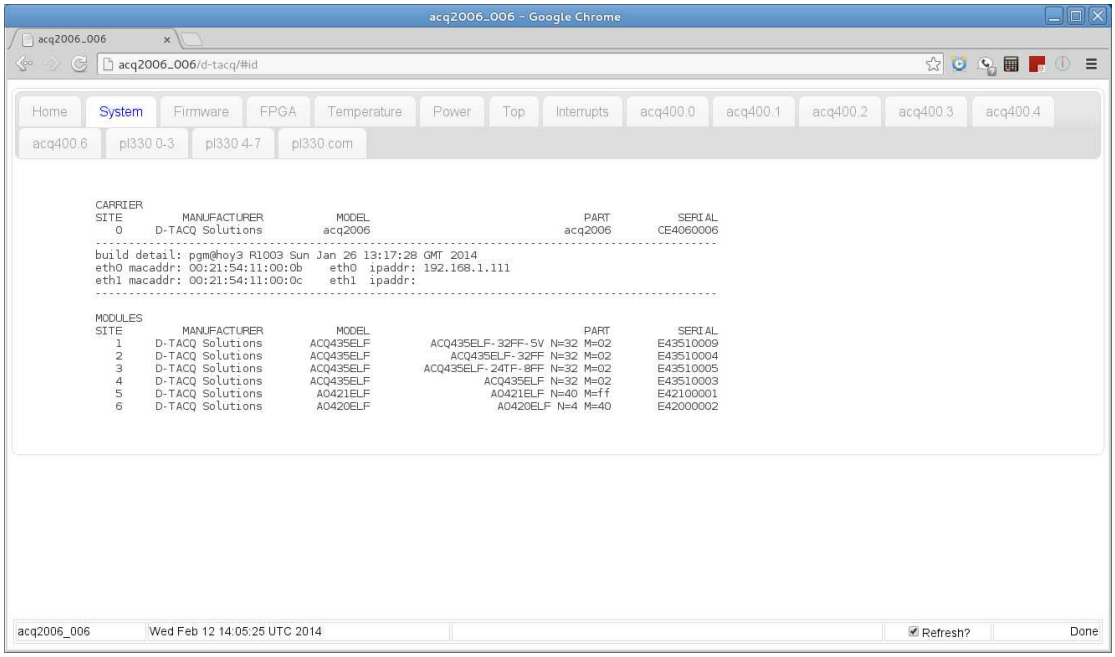
Text is preferred for support because it's easy to search and annotate; however below we show screenshots to give a better feel for the appearance of the pages.

An example is provided at 7.9

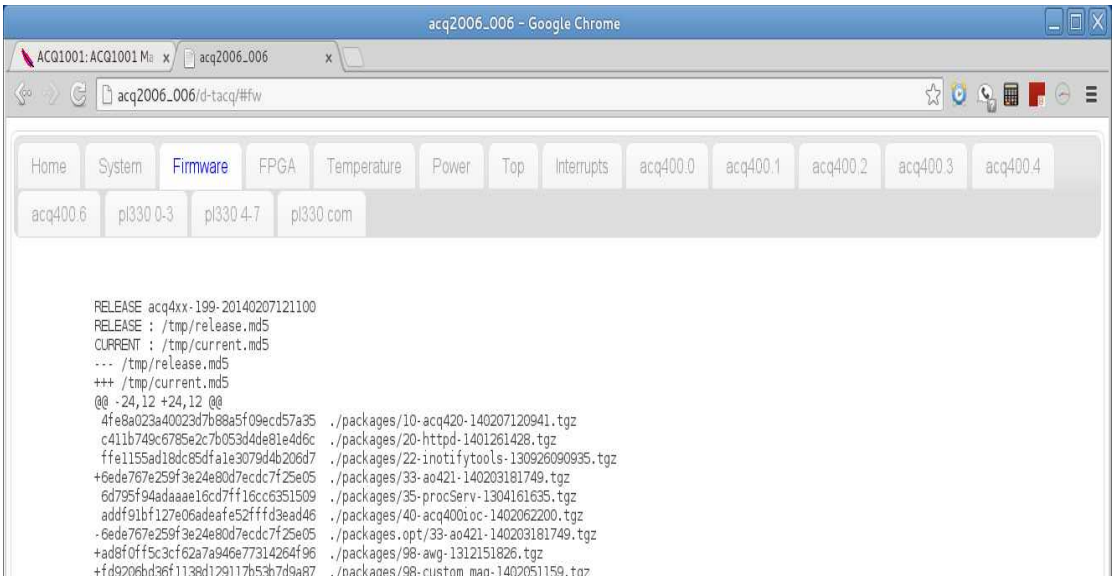
7.1 Home Page



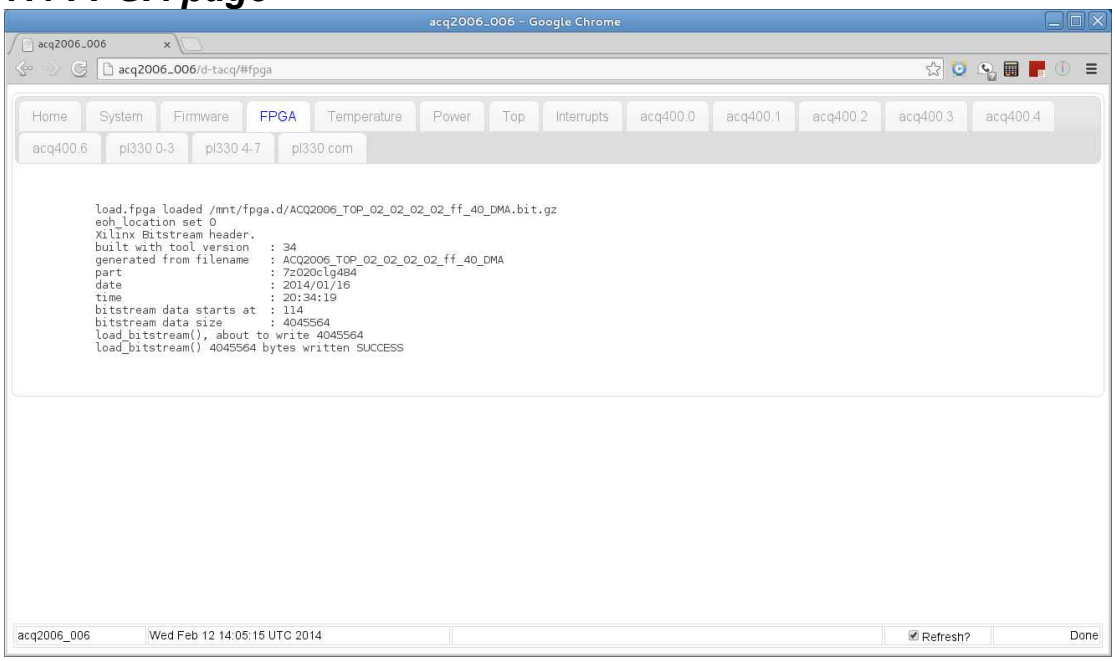
7.2 System Page



7.3 Firmware Page



7.4 FPGA page

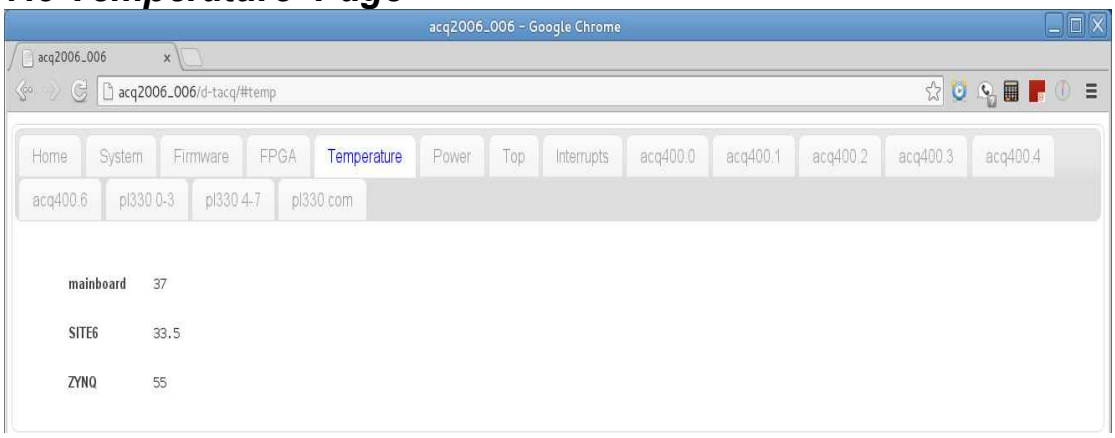


7.4.1 Example: FPGA as text (preferred format for email) :

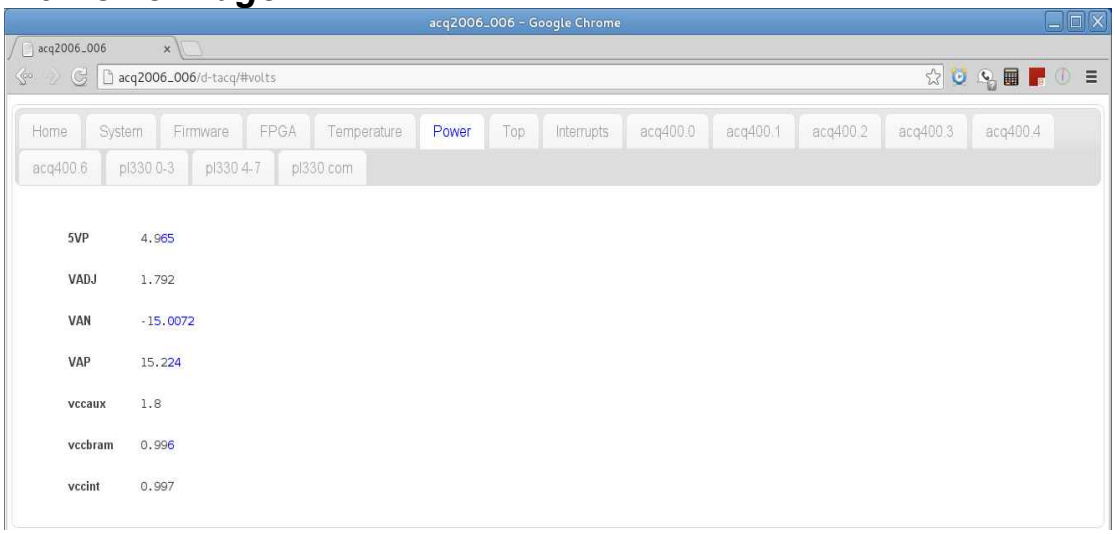
Uncheck the “Refresh?” check box, cut and paste text in the normal way:

```
load.fpga loaded
/mnt/fpga.d/ACQ2006_TOP_02_02_02_02_ff_40_DMA.bit.gz
eoh_location set 0
Xilinx Bitstream header.
built with tool version    : 34
generated from filename    : ACQ2006_TOP_02_02_02_02_ff_40_DMA
part                       : 7z020clg484
date                       : 2014/01/16
time                       : 20:34:19
bitstream data starts at   : 114
bitstream data size        : 4045564
load_bitstream(), about to write 4045564
load_bitstream() 4045564 bytes written SUCCESS
```

7.5 Temperature Page



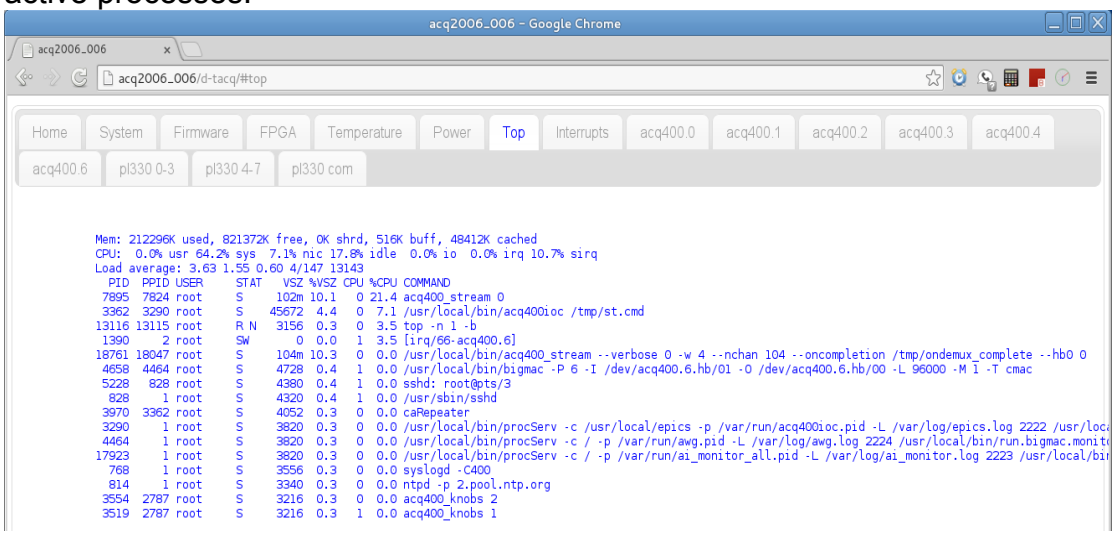
7.6 Power Page



Page contents are dynamic – updates shown in blue .

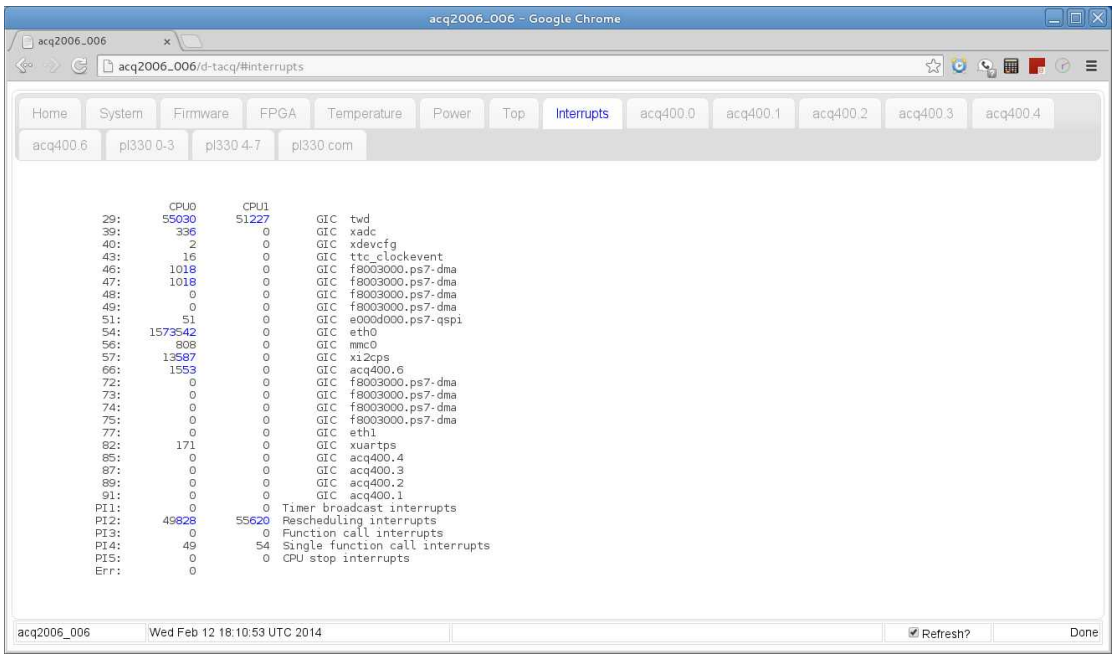
7.7 Top Page

Top runs the unix top(1) command, shows an interactive display of the most active processes:



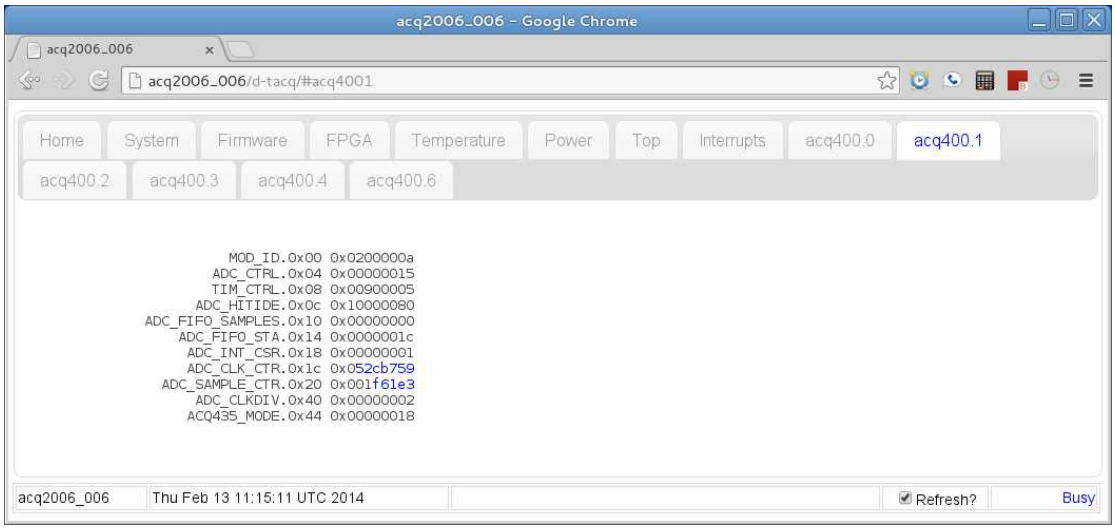
7.8 Interrupts Page

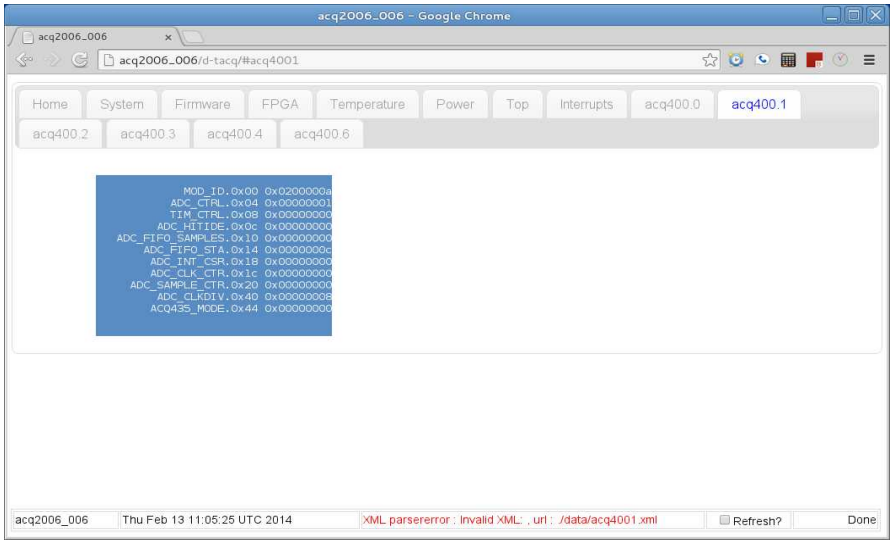
Provides a live view of the interrupt system



7.9 Site Specific Pages

The webserver provides a dynamic view of the register set for each installed site in the system. This information may not be too useful to the user, but is critical for D-TACQ diagnosis. Please be sure to use the text cut and past technique if asked:






```
              MOD_ID.0x00 0x0200000a
         ADC_CTRL.0x04 0x00000001
         TIM_CTRL.0x08 0x00000000
       ADC_HITIDE.0x0c 0x00000000
ADC_FIFO_SAMPLES.0x10 0x00000000
    ADC_FIFO_STA.0x14 0x0000000c
     ADC_INT_CSR.0x18 0x00000000
     ADC_CLK_CTR.0x1c 0x00000000
  ADC_SAMPLE_CTR.0x20 0x00000000
      ADC_CLKDIV.0x40 0x00000008
```

8 Remote Reference

The ACQ4xx appliance is divided into sites {1,2 .. 6} with module-specific functions, and site 0, the motherboard. Each site presents a set of “knobs” or controls with simple key=value settings. The “knobs” are designed to be easily scriptable, and are accessible from both locally on the ACQ4xx and remotely using a dedicated socket connection.

8.1 Host API HAPI

A comprehensive remote Host API is available

8.2 Common Features

- Connect to a site control socket 4220+Site number (0= System Controller)

```
nc acq2006_006 4221
```

- This interface is meant to be easy to control by a computer script. For Humans, it's easier if there is a prompt. The prompt includes the site number

```
prompt on

acq400.1 0 >
```

- Execute a query – we ask for the mode and it responds on the next line

```
acq400.1 0 >hi_res_mode
1
acq400.1 0 >
```

- Execute a command

```
acq400.1 0 >hi_res_mode=1
acq400.1 0 >
```

- Execute a query again

```
acq400.1 0 >hi_res_mode
0
acq400.1 0 >
```

- help : how to tell what options there are on this service

```
acq400.1 0 >help
MANUFACTURER
MODEL
...
hi_res_mode
```

- help2 : how to tell what the options do and what are valid parameters. You can get (query) any parameter (r) and set parameters denoted (w)

```
acq400.1 0 >help2
MANUFACTURER         : r
      manufacturer
```

```
hi_res_mode                 : rw
      [0|1]
```

- wild card queries are supported, eg:

```
acq400.1 0 >SIG*FREQ
SIG:clk_count:FREQ 4.91623e+07
SIG:sample_count:FREQ 48003.5
acq400.1 0 >
```

- HIGH LEVEL and low level commands.
  In general, knobs in CAPS are considered to be “high level” commands (they are frequently EPICS PV's), and where a HIGH LEVEL and a low level knob coexist, it's recommended to use the HIGH LEVEL command, so that any controller logic is utilised, and to guarantee a consistent view to external clients.
- The exception to the above recommendation is for signal setting:

8.2.1 Signal Setting “Triplet”:

- where the low level command has a convenient “triplet” structure, mapped to 3 PV's. All signal triplet PV's are “round tripped” so that the EPICS PV's track the low-level setting automatically. Although script users may prefer the self-description of the 3-PV's method.

```
Eg
set.site 1 trg=1,1,0
equivalent to
set.site 1 TRG=EXT
set.site 1 TRG:DX=d1
set.site 1 TRG:SENSE=FALLING
```

For typical UI representation, see 6.3.2

Sample listings are given below, but please note, the lists are not complete; for accurate listing, run the help command on your system.

D-TACQ can also supply a full configuration-specific command reference for any system as it leaves the factory, if required.

8.3 System Controller

```
acq400.0 0 >help2
NCHAN                : rw
      number of channels
SIG:CLK_EXT:COUNT    : rwx
      external clock count
SIG:CLK_EXT:FREQ     : rwx
      external clock frequency
SIG:CLK_MB:COUNT     : rwx
      MB clock count
SIG:CLK_MB:FREQ      : rwx
      MB clock frequency
SIG:CLK_S1:COUNT     : rwx
      site 1 clock count
...
...
SIG:TRG_MB:FREQ      : rwx
      not used
SIG:TRG_S1:COUNT     : rwx
      site 1 trig count
SIG:TRG_S1:FREQ      : rwx
      site 1 trig frequency
...
aggregator           : rw
      displays aggregator state, do not set
autocap              : rwx
      [on|off] control autocapture
data32               : rw
      [0|1] data size 16bit/2byte or 32bit/4byte
fpmux                : rwx
      [fpclk|xclk] control clock source
gpg_clk              : rw
      configure Gate Pulse Generator clock source
gpg_enable           : rw
      enable GPG
gpg_mode             : rw
      set GPG mode One Shot | Loop | Loop/Wait
gpg_sync             : rw
      /usr/share/doc/acq400_help0:gpg_sync EXT,dx,RISING  configure
GPG sync with (sample) clock source
gpg_top              : rw
      query gpg top address
gpg_trg              : rw
      configure GPG start trigger source
mb_clk               : rwx
      [FINKHz FOUTKHz]
run0                 : rwx
      aggregate from sites
soft_trigger         : rwx

spad                 : rw
      [0|1] scratchpad enable
spad0                : rw
      [0x12345678] scratchpad 0 entry (sample count: do not set)
```

9 Package Reference

9.1 What is a package.

A package is code and data that customizes the system.

Active packages are located in `/mnt/packages`.

Packages have a canonical naming structure:

SEQ-NAME-REV.tgz

Where *SEQ* is a two-digit number that indicates the position in the unpacking sequence (starting from zero)

NAME is the unique package name

REV is a YYMMDDhhmm revision code

.tgz : the package is a zipped tarball.

9.1.1 Package Structure

It's a tarball that will be unpacked into the file system at the root '/'

Optionally it includes a named init file, to be executed after the package has been unpacked.

example:

```
tar tvzf /mnt/packages/05-acq1001-140306181234.tgz  | cut -c 52-
./usr/local/bin/acq1001_init_gpio
./usr/local/acq1001.map
./usr/local/init/
./usr/local/init/acq1001.init
```

- *SEQ* is 05 : this executes early
- *NAME* : is acq1001
- *REV* : is 140306181234 (2014, March 6)
- The init file is acq1001.init

9.2 Summary of Current Standard Packages

Packages stored in /mnt/packages are install to ramdisk at boot time

All packages NN-package-REV.tgz

Packages install in sequence number NN.

SEQ	Name	Description
02	tcl	TCL / Expect scripting language
03	acq400_common	common utils for board boot
05	acq1001	acq1001 board boot
05	acq1002	acq1001 board boot
05	acq2006	acq2006 board boot
05	acq2006b	acq2006b board boot
05	acq2106	acq2106 board boot
06	procServ	procServ process server
10	acq420	module enumeration and device driver
20	httpd	web server
39	transient	transient/faultmonitor capture control.
40	acq400ioc	EPICS IOC

9.3 Summary of Current Optional Packages

Packages stored in /mnt/packages.opt are held in reserve.

To enable a package, mv it to /mnt/packages and reboot.

SEQ	Name	Description
04	custom_pmod	support for PMOD
15	custom_sshkeys	customize ssh keys
21	custom_wdt	watchdog timer
33	ao421	support AO421FMC hardware
38	custom_8pps	Gate Pulse Generator+8pps example
39	kmux	support Keithley Mux
70	mdsshell	mdsplus thin client
80	custom-cifs	cifs (Windows) file share client
98	awg	special Arbitrary Waveform Generator, block load, with gain and offset control

SEQ	Name	Description
98	custom_mag	customer specific
98	custom_sos	Simple One Shot : transient capture with local demux. ***DEPRECATED***
99	custom_sync	Configures multi-box sync feature
99	autocapture	autocapture: turnkey capture on start
99	custom_awg	standard Arbitrary Waveform generator, with load-by channel capability.
99	custom_bolo	support for BOLO8 hardware.
99	custom_hil	hardware in the loop support
99	custom_mb	customer specific
99	custom_sjo	customer specific

9.4 Including an Optional Package in the boot

mv /mnt/packages.opt/PACKAGE /mnt/packages

9.5 Modify and Re-package a package

You might want to add some customization?.

Untar in a safe place, tar it back up, give it a new name.

Stash the old package in /mnt/packages.opt.

Remember, the NAME must be unique in /mnt/packages.

9.6 New Filesystem image / overlay concept

A new packaging style is introduced in 2020, this runs alongside the existing packages. The original packages are unpacked into RAM, the new packages are read only file system images, that allow for much larger standard distributions (eg Python, EPICS4) without consuming excess RAM.

The filesystems are optimised for size using squashfs. Overlay file systems are also supported, where the overlay is a small ramdisk image that is intended for local customization, eg setup files.

9.6.1 Packages in standard release

./ko/packageko-4.14.0-acq400-xilinx-200605091137.img

kernel modules. Only required modules need to be in RAM

./ko/fpga-218-20200605095138.img :: 79.5M

FPGA personalities. No FPGA personalities need to be in RAM

9.6.2 Custom Package examples:

39-epics7-2006050840.ovl

- Overlay packages for EPICS7 / PVA functionality. These are squashfs images, with a local RAM overlay.
- Handled in sequence from:
  - /mnt/packages/39-epics7-2006050840.ovl :: 26.5M
- Mounted at:
  - /usr/local/epics7/
- Runs this init file on boot:
  - /usr/local/epics7/epics7.init
  - Typically the init file will create links from standard locations

```
acq2106_180> cat /usr/local/epics7/epics7.init
#!/bin/sh
echo +++ epics7.init
(cd /usr/local/lib; for so in /usr/local/epics7/lib/*.so; do ln -s $so; done)
(cd /usr/local/bin; for file in /usr/local/epics7/bin/*; do ln -s $file; done)
```

- Result:

```
acq2106_180> ls -l /usr/local/lib | grep epics7 | cut -c 58-
libCom.so -> /usr/local/epics7/lib/libCom.so
libasyn.so -> /usr/local/epics7/lib/libasyn.so
libca.so -> /usr/local/epics7/lib/libca.so
libcmdButtonsSupport.so -> /usr/local/epics7/lib/libcmdButtonsSupport.so
libdbCore.so -> /usr/local/epics7/lib/libdbCore.so
libdbRecStd.so -> /usr/local/epics7/lib/libdbRecStd.so
libdevTestGpib.so -> /usr/local/epics7/lib/libdevTestGpib.so
libnt.so -> /usr/local/epics7/lib/libnt.so
libpv.so -> /usr/local/epics7/lib/libpv.so
libpvAccess.so -> /usr/local/epics7/lib/libpvAccess.so
libpvAccessCA.so -> /usr/local/epics7/lib/libpvAccessCA.so
```

/mnt/packages/91-pvaPy-2006050841.ovl

- Overlay package for python PV access
- Handled in sequence from:
  - /mnt/packages/91-pvaPy-2006050841.ovl :: 7.9M
- Mounted at:
  - /usr/local/pvaPy/
- Runs this init script
  - /usr/local/pvaPy/pvaPy.init

10 Data Capture

10.1 Concept

The ACQ400 FPGA implements an AGGREGATOR function that fetches data from each participating module in turn and makes it available for DMA. The software implements a streaming dma function. To capture data, it's always necessary to “set up a stream”. The streamed data could go to the network .. a “Streaming Data” or to local DRAM “Transient Data”. The full rate data can even be discarded, using the EPICS waveform display as a diagnostic indicator.

Raw data is streamed continuously to Ethernet. This is valid at rates up to low 20's Mbytes/second

10.2 Preparation

10.2.1 Define capture conditions

Clock and Trigger definition. It's not necessary to set this up since sensible defaults are provided.

- **`set.site`** 1 clk=CLK
- **`set.site`** 1 trg=TRG

10.2.2 Define the AGGREGATOR set

The **`run0`** command configures the set of modules that are participating in the capture, eg to include modules in sites 1,2,3,4:

- **`run0`** 1,2,3,4

The system is now primed and ready to go. All we have to do is “start the stream”.

10.2.3 Port 4210 : Aggregator data port

To start the stream, connect a socket to port 4210 and read data. In the simplest case, run streamtonowhere

```
nc localhost 4210 > /dev/null
```

Or, with a rate indicator:

```
nc localhost 4210 | pv > /dev/null
```

10.3 Network Streaming data capture

Stream data over Ethernet. Effective up to 30 MB/s (suitable for ACQ435, and ACQ420/ACQ425 in low channel count or low rates) :

```
# from another host
nc UUT 4210 | pv > mybigdatafile
```

Of course, all **nc** does is open a TCP socket and read data. Your host side software may not be **nc**, but it will do the same thing – open a socket and read. For users with no access to **nc** or **pv** , it's possible to run this demonstration on the UUT itself:

```
# self hosted. send to null to avoid disk overrun.
nc localhot 4210 | pv > /dev/null
```

This is rate limited by the ZYNQ TCP/IP capability. For full rate streaming, we recommend using ACQ2106 and PCI-Express or SFP fiber optic.

10.3.1 Strategies for reducing Wire Rate

What to do if the average input date rate exceeds the average stream data rate?. One answer is to reduce the Sample Rate until the two rates are matched, however, obviously this can conflict with user need for higher rate sampling..

10.3.1.1 Reduce channel count

Certain models of hardware eg ACQ425, ACQ435 offer a hardware means to reduce the number of channels and therefore the data rate. Hardware channel masking is in banks ie NOT discrete channels.

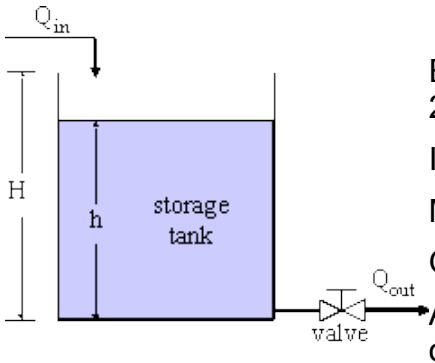
In addition, there's a software control to reduce the output rate by discarding higher channel counts. Select at boot time as follows:

```
/mnt/local/rc.user:
echo 'STREAM_OPTS="--subset=8"' > /etc/sysconfig/acq400_streamd.conf
```

Reduces output data to channels to 1..8

10.3.1.2 Dump Buffers.

The streaming data flow can be likened to a tank filling with concurrent drainage:



Schematic diagram of a filling and draining a storage tank.

Example case: ACQ1001+ACQ480, 8 channels, 2.5MSPS/channel

Input Rate (Qin) : 8 x 2 x 2.5 = 40MB/s

Memory buffer size: 512MB

Output Rate (Qout) : 32MB/s.

Average fill rate of the buffer: 8MB/s, buffer will overflow in 64s.

This isn't a bad result, compared to a standard transient capture, 40MB/s to 512MB buffer, the buffer fills in 12s.

=> If the rates are nearly in balance, streaming data gives a "long transient".

When the buffer finally fills, the ACQ400 software will discard all buffer data and start again. This is fatal for some applications that really do require continuous data. However, the data does continue to flow, and it's possible that the user application could tolerate the loss of data, if it's informed when the data break occurred.

If the system is streaming data with included time information, maybe one of the formats from 11. For 32 bit data, the embedded frame is certainly an option, but for 16 bit data, embedding time information adds to the size of the data, not helpful when we are trying to optimise the data rate. Instead we offer:

"Stream Start of Buffer Signature" :

- The ACQ400 memory buffer is blocked in large blocks, either 1MB or 4MB

(eg 4MB for ACQ480).

- At the start of every block, the software will transmit a "Start of Buffer Signature" (**`sob-sig`**), a one-sample wide data pattern marking the start of buffer and showng the buffer index.

- Client software will have to scan for **`sob-sig`**, check the buffer index for continuity and discard the signature.

- The buffer index will rotate in a regular pattern 0..MAX, repeating. If the sequence is broken, that means data has been lost. The client application can deduce how much data was discarded by the difference from the previous index.

To enable this mode:

```
/mnt/local/rc.user:
echo 'STREAM_OPTS="--stream-sob-sig=1 --subset=8"' \
      > /etc/sysconfig/acq400_streamd.conf
```

The signature appears as a row on N u32 values, to the size of one sample:

```
0xaa55fbff 0xaa55fbff ... 0xaa55fbff <half> BUFFERNUM.. BUFFERNUM
```

10.3.1.3 Burst Mode eg Repeating Gate Mode

With *RGM*, the capture proceeds indefinately, but is broken into bursts by a hardware *GATE* signal.

Repeating Gate Mode *RGM* is frequently superimposed on streaming data.

For details, see 10.5

Provided the average capture rate is less than the average stream rate, then this system will run “forever”. It could also be run concurrently with Buffer Dumping as in 10.3.1.2

10.4 Transient Data Capture

This is “SHOT BASED” data acquisition. The unit allocates 512MB memory for shot data. Capture starts on trigger and ends after a pre-programmed number of samples. Data is then available for upload.

Transient capture uses a ZERO-COPY mechanism to store the data in a series of 512, 1MB buffers. After the shot, the data may be be access raw (fastest, but most complex for applications) or de-muxed in place. The demux data is presented as a set of virtual files:

/dev/acq400/data/$site/??

The simplest way to access these files is to install the “channel server” :

make-ch-server

Do this AFTER running “**run0**”. Now each channel's data is presented per channel on sockets 53000+*CC*. Simply connect to the required port and read the data.

10.4.1 Configure a Transient Capture.

Transient capture may be configured programmatically as follows, where PRE, POST are pre-trigger, post-trigger capture lengths in samples

```
transient [PRE=N] [POST=N] [OSAM=1] [SOFT_TRIGGER=1]
Set trigger condition.
If PRE=0:
set.site 1 trg=1,0,1          # external TRG, rising
set.site 1 event0=0,0,0       # no event
If PRE > 0:
set.site 1 trg=1,1,1          # local (SOFT) TRG
set.site 1 event0=1,0,1       # external rising edge causes PRE→POST
```

Then run

set_arm

With SOFT_TRIGGER=1 set, the capture will start immediately. If it's not set, run the **soft_trigger** command separately

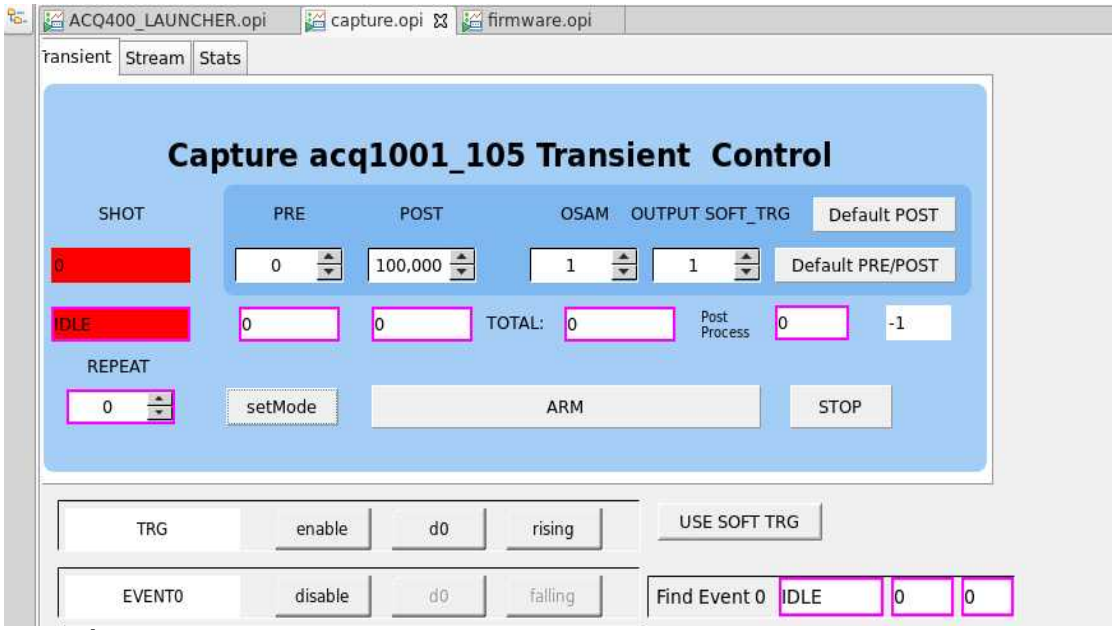
Monitor the capture using **acq4xx-transient-console**.

Note that a CSS OPI provides the same functionality as a remote-able GUI.

The transient and set_arm commands are available on the command line to local scripts, they are also available as knobs on site 0 :

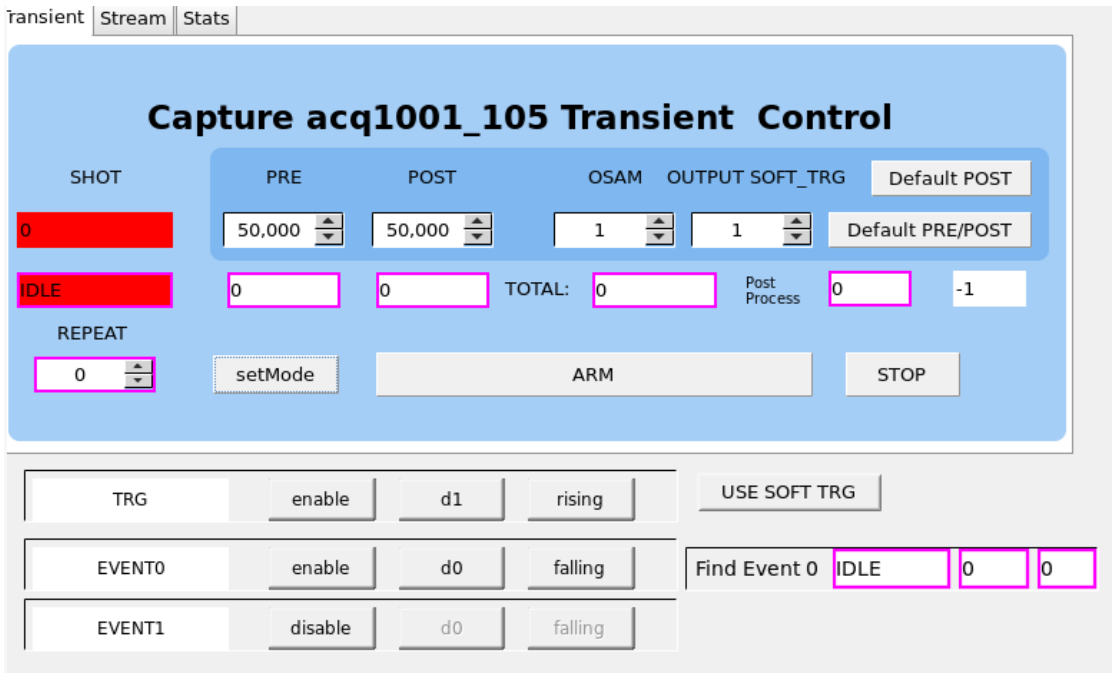
```
set.site 0
transient POST=1000 SOFT_TRIGGER=1
set_arm
transient_state
0 0 0 0
```

CS-STUDIO OPI, Default POST selected.




```
set.site 1 trg=1,0,1
```

CS-Studio OPI, Default PRE/POST selected




```
set.site 1 trg=1,1,1          # local (SOFT) TRG
set.site 1 event0=1,0,1
```

10.4.2 Transient Upload Methods

There are many possible ways to access the data, here are some examples.

- Load from well known port e.g. 53000+CC
- use **`scp`** (slow).
- The multi-purpose **`curl`** client is included.
- The files could be saved to an NFS mount.
- The files can be saved to a CIFS (Windows) file share eg perhaps from a NAS box.
- Direct upload to MDSplus is supported, please see 19.
- Save files to localUSB disk.

10.5 Burst Mode Capture

With *RGM*, the capture proceeds indefinately, but is broken into bursts by a hardware *GATE* signal. This keeps the data stream in sync with some external event, and it may also reduce the data rate on the wire. There are 3 ways to operate:

- *RGM*: *GATE* ACTIVE starts data flow, *GATE* INACTIVE stops flow.
- *RTM* : Repeating Transient mode: Trigger to start, captures a programmable burst length and stops.
- *SRTM* : As *RTM*, but resyncs the sample clock to the Trigger every trigger edge. This feature is only available on oversampling digitizers like ACQ435ELF.

10.6 Voltage Calibration

ACQ400 units are supplied with calibration data. The calibration data allows client applications to convert raw binary data from the ADC to calibrated volts. Calibration values automatically include the current gain setting on variable gain modules.

10.6.1 Automation

From an EPICS client, eg cs-studio, chose “Plot Volts” to see calibrated data automatically. The MDSPlus thin client function includes calibration by default. Users of the HOST API HAPI have access to calibrated data through the method `acq400_hapi.Acq400.chan2volts.`

10.6.2 ESLO/EOFF : Slope and offset

ACQ400 uses the EPICS AI record conventions of ESLO and EOFF to provide calibration data.

```
get.site 1 AI:CAL:ESLO
AI:CAL:ESLO 9 0 3.83565e-05 0.000306892 0.000306888 0.000306841
0.00030691 0.000306819 0.000306808 0.000306834

# NAME LENGTH [0] [1] .. [N]
# Channels values indexed from 1

acq1001_324> get.site 1 AI:CAL:EOFF
AI:CAL:EOFF 9 0 5.83948e-05 2.01251e-05 0.000772002 8.91821e-05
9.68462e-05 0.000426686 0.000165888 0.000741205
```

ValueInVolts = RawValue * ESLO + EOFF

Where RawValue is a signed 16 bit number for 16 bit ADC, signed 32 bit number for higher resolution. For 24 bit ADC, 11.1, normalise the raw value first by dividing by 256 (>> 8).

10.6.3 Calibration File

Calibration data for each module is held in xml format, loaded on boot an available from the front web page as a hyperlink, eg

http://acq1001_324/d-tacq/cal/site.1.xml

11 Streaming Data format

By default, raw data is streamed to the streaming output port. ACQ400 also offers a variety of hardware-generated frame structures to aid in decoding this data over a long period.

11.1 ACQ435, default coding of spare bits

ACQ435 data, with 24 data bits in a 32 bit field, imposes a module/channel encoding on the unused bits as an aide to checking channel alignment:

- d7-d5 : SITE SSS {1..6}
- d4-d0 : CHANNEl {0..31}

d31-d08	*d7-d5*	*d4-d0*
CH00 data 0xaabbcc	SSS	00000
CH01 data 0xaabbcc	SSS	00001
...	...	...
CH31 data 0xaabbcc	SSS	11111

When converting ADC data to counts or voltage, treat the 32 bit number as signed and divide by 256 to remove the channel coding.

11.2 ACQ435, Bitslice Frame, Embedded bits

Enable as follows:

```
# set.site N bitslice_frame=1
```

With Embedded bits enabled, bits d7.d5 become dynamic, with values inserted by the FPGA. With a 32 channel sample, a 3 x 32 bit words of signaling information are available.

- d7 : SC : Sample Count, inserted by hardware
- d6 : SEW0 : Software Embedded Word 0, inserted by software. A typical use of this is PPS: a latched sample count on external PPS signal.
- d5 : SEW1 : Software Embedded Word 1, inserted by software. A typical use of this is to embed an NMEA string from an external GPS, 3 chars at a time. The 4th char is typically an index value; this is necessary because the SEW is updated asynchronously at a slow rate relative to the sample rate, and a decoder needs to see the index value change to detect a new data word.

11.2.1 Example Package

The “CUSTOM_TUNA” package shows how to use the bitslice frame to embed both a timestamp (relative to a GPS ONEPPS edge) and low rate GPS NMEA data in the data stream.

11.3 Scratchpad

Valid for all data type (16 or 32 bit).

Extended 8 x 32 bit tag, inserted every sample.

The scratchpad is encoded as follows.

- SC : Sample Count inserted by hardware
- SEW[1-7] : Status words inserted by software.

Word	***Function***
0	SC Sample Count
1	SEW1 / USECS / DI4
2	SEW2
3	SEW3
4	SEW4
5	SEW5
6	SEW6
7	SEW7

11.3.1 SPAD1 set SEW1

set.site 0 spad=1,N,0 # N >= 2

set.site 0 spad1=number

11.3.2 USECS Count in SPAD1

set.site 0 spad=1,N,0 # N >= 2

set.site 0 usecs=1,0,1 # source USEC counter from d0

eg

set.site 0 SIG:SRC:CLK:0 = INT01M

11.3.3 DI4 in SPAD1

DI4 are the 4 digital inputs on the [HDMI] SYNC_IN port

set.site 0 spad=1,N,1 # N >= 2

12 Clock and Synchronisation

12.1 Sync cases

12.1.1 Between modules on a carrier

Carriers include a synchronization bus to allow multiple modules to share clock, trigger and index signals if required.

The carrier has 4 x 8 bit sync busses:

- CLK : clocking controllable
- TRG : trigger (capture start control)
- EVT : Event (pre/post, multiple events)
- SYNC : Sample rate output from site.

On each bus:

- d0 is the front panel signal (CLK/TRG only)
- d1 is a local generated signal (eg MB_CLK, SOFT_TRG)
- d2 is the SITE1 output, d3..d7 are outputs from SITES2..6 if fitted.

Examples:

- set.site 1 trg=1,0,1 : Site 1: enable front panel trigger, rising
- set.site 1 trg=1,1,1 : Site 1: enable local (soft) trigger, rising
- set.site 1 clk=1,0,1 : Site 1: enable front panel clock, rising

Tip

Use the counters OPI to monitor signal state.

TRG: Is a one-shot. Any capture needs ONE TRG transition to start.

EVT: Transition occurs during the capture. “event0” can cause the transition from PRE to POST phases in a PRE/POST capture. “event1” is typically used to mark external signals eg ONEPPS.

12.1.2 Between carriers

Carriers implement a simple timing daisy-chain based on standard, low cost HDMI cables. Each carrier has an IN and an OUT HDMI connector.

With multiple units, the units are connected by a daisy chain of HDMI cables, the first unit in the chain is defined as the MASTER and all downstream units are SLAVES. Definition of MASTER and SLAVE is automatic. The MASTER has the plant TRG signal on the front panel. The MASTER enables the TRG to itself and all the SLAVEs only after it has been armed. So, to ensure simultaneous capture on all boxes, user HOST software should first connect the data path on all the SLAVE boxes, lastly, connect to the MASTER, and

data flows on the next TRG edge.

12.2 Universal Clock command : sync_role

A single common command is provided to work with all carriers, all modules, including delta-sigma. The command is sync_role, provided as a site 0 service.

12.2.1 Roles

- **`solo`** : a stand-alone box, local clock
- **`master`** : local clock, local trigger, masters other boxes
- **`fpmaster`** : front panel clock, front panel trigger, masters other boxes
- **`rpmaster`** : plant clock and trigger from rear SYNC IN, masters other boxes
- **`slave`** : clock and trigger are provided on SYNC IN by another MASTER box.

Notes:

- Slave boxes are in fact Masters with respec to any other boxes further down the SYNC daisy-chain.
- The command supports modifiers to enable sub-roles
  - eg Fpmaster clock, front panel clock, local (soft) trigger.
  - eg Master clock, local clock, front panel trigger.
- The CLKHZ argument refers the ADC sample rate; this is the local clock for SAR ADC's, for Delta-Sigma ADC's, the local clock will be the modulator clock, and this is calculated automatically.

12.2.2 Command Reference: sync_role

```
acq2106_119> get.site 0 sync_role help
USAGE sync_role {fpmaster|master|slave|solo} [CLKHZ] [FIN]
modifiers [CLK|TRG:SENSE=falling|rising] [CLK|TRG:DX=d0|d1]
modifiers [CLKDIV=div]
```

- *role* : one of {fpmaster|master|slave|solo}
- CLKHZ : sample clock frequency in Hz (not needed for slave)
- FIN: Input sample frequency, recommended for fpmaster.
- modifiers:
  - CLK|TRG:SENSE : set edge sensitivity
  - TRG:DX=d0 : override trigger setting to front panel (external)
  - TRG:DX=d1 : override trigger setting to local (soft).
- CLKDIV : set local clock divider. This is normally done automatically and should be AVOIDED in normal use.
- If *role* is “help”, prints help
- If *role* is omitted, queries current setting.
- CLKHZ, FIN may be specified as any of:
  - Integer eg 1000000
  - Float eg 1e6
  - kilo eg 1000k
  - Mega eg 1M

12.2.3 Use Case: Solo box, 200kHz SR

```
set.site 0 sync_role solo 200000
```

12.2.4 Use Case: Front panel Clock, 200kHz SR

```
set.site 0 sync_role fpmaster 200k
```

12.2.5 Use Case: Front panel Clock 1MHz, 200kHz SR, local trigger override

```
set.site 0 sync_role fpmaster 200000 1000000 TRG:DX=d1
```

12.2.6 Use Case: Front panel Clock, 12MHz, 80MSPS SR

```
set.site 0 sync_role fpmaster 80M 12M
```

12.2.7 Use Case: Slave Module

```
set.site 0 sync_role slave
```

12.2.8 Use Case: Local clock 2MHz, FP trigger, falling edge

```
set.site 0 sync_role master 2000000 TRG:DX=d0 TRG:SENSE=falling
```

12.3 Gate Pulse Generator GPG

A hardware based STL sequencer that generates a series of pulses on trigger. The GPG has 8 possible outputs, intended for use as internal triggers. Some of the outputs may be output from the UUT.

There are 3 separate wrappers for GPG

12.3.1 delay_trigger : runs as a one-shot delay

Command available on site 0

```
delay_trigger USECS

set.site 1 trg=1,1,1
```

By default, the GPG triggers off the front panel TRG.d0

After USECS microseconds, the GPG will generate a pulse on TRG.d1

delay_trigger configures TRG.d1 to source from GPG, so it's no longer "soft_trigger". Select the site 1 trigger to be d1.

12.3.2 STL : load stl to generate an arbitrary sequence

UUT provides an service that accepts STL and loads the GPG with the user sequence.

- GPGSTL= 4541 : loads user STL sequence
- GPGDUMP = 4543 : client can view loaded sequence

Note that GPGDUMP shows raw hardware values from the GPG and will not match the STL one for one.

Loading STL is a complex subject, we recommend that users use the load utility provided by HAPI:

- run_gpg.py

12.3.3 Pulse Train.

Include optional package 38-custom_8pps, use command load.pulse_def to configure a pulse train. Load.pulse def configures a sensible default, many other options are possible. It's a TCL script, intended for site localisation.

Multi-box sync is enabled with the GPG via this command:

```
set.master-slave-hdmi.role {AI64M, AI64S} PPS SR TRANSLEN
```

Where AI64M is the master box, AI64S each slave box. PPS is pulses per second, triggered by an external ONEPPS (eg from a GPS unit) and TRANSLEN is the pulse capture length in samples

13 DSP Features

13.1 Oversampling filter

Accumulate/Decimate filter. Accumulate NACC samples, then output one value. Allows for data reduction with improved SNR and aliasing reduction due to oversampling.

set.site N nacc=NACC[,SHR[,START]]

- NACC=1,2,3,4..32,[64] : number of samples to accumulate over
- SHR=1,2,3,4,5,[6] : data is right shifted by this amount to maintain scaling
- START=N : sample to start accumulating on

In general, only the one argument NACC need be supplied, the software interface will calculate the best fit for the requirement.

eg

	Command	Action	Settings		
			NACC	SHR	START
1	set.site 1 nacc=4	exact scale	4	2	0
2	set.site 1 nacc=64	exact scale	64	6	0
3	set.site 1 nacc=50	decimate by 50, accumulate over 32, shift for exact scale	50	5	12
4	set.site 1 nacc=50,5,0	decimate by 50, accumulate over 50, shift 5, scaling 50/32	50	5	0

User has control of trade-off between #3 and #4 :

- #3 achieves the “decimal divide SR” (eg 1000k/50 = 20kHz), with exact voltage scaling, but part of the sample region is omitted.

- #4 gives the desired sample rate 20kHz, but the data is scaled up by 1.6. This approach works provided signals of interest do not saturate the arithmetic. NB: result for the case of arithmetic overflow is UNDEFINED.

13.2 FIR Filters

Ask D-TACQ for details

13.3 Custom DSP Functions.

eg Analog Threshold Detect.

14 AWG Feature / Wavegen

14.1 Raw data waveform

Package: 11-custom_awg-yymmddHHMM.tgz

Buffer partitioning. By default AWG loads at buffer 0.

That's a reasonable choice, but there are two reasons to optimise by starting at the highest possible buffer number:

1/ Load time is proportional to the number of buffers – starting at a higher buffer number can reduce load time.

2/ In a mixed AI, AO system, loading the AO high above the AI reduces the possibility of overwrite in-shot, and may allow repeat shots without reload.

Key parameters

sys/module/acq420fmc/parameters/nbuffers	512
/sys/module/acq420fmc/parameters/bufferlen	1048576
/sys/module/acq420fmc/parameters/distributor_first_buffer	470

Here we set distributor_first_buffer to 470, allowing a maximum AWG length 32 MB (1s for 1xAO424 at max 500kSPS), with a reasonably fast load time.

Recommendation:

Set on load from /mnt/local/acq420_custom

14.2 AWG Modes

14.2.1 AWG Definitions

- ONE SHOT:: load raw data, play once on trigger.
- ONE SHOT, AUTO REARM:: play once, then run again on next trigger
- ONE SHOT, CONCURRENT::
  - For a very long ONE SHOT, save time by allowing capture to start while load is in progress.
- CONTINUOUS :: load raw data once, play it continually.
  - NB: the SAME data plays continually.
- STREAM :: load and play data without limit.
  - Drop the connection to stop the stream.

14.2.2 AWG Network Interface

PORT	Mode	Command
54201	ONE SHOT	bb load --mode 1
54202	ONE SHOT AUTO REARM	bb load --mode 2
54203	ONE SHOT CONCURRENT LOAD	bb load --mode 1 --concurrent 1
54205	CONTINUOUS	bb load --mode 0
54206	CONTINUOUS, CONCURRENT LOAD	bb load --mode 0 --concurrent 1
54207	STREAM	bb_stream
54200	SHA1SUM	shows sha1sum of loaded data
54024	DUMP	outputs loaded data

* CONCURRENT LOAD: allow play to start during LONG load.ssh roo

14.3 Raw data waveform examples

All data for one of more sites is stored in raw (multiplexed) format to DRAM.

The waveform is played through an FPGA “DISTRIBUTOR” mechanism associated with Site 0; the FPGA is responsible for farming the data to target sites.

The distributor is configured as follows:

```
set.site 0 play0 5,6      # distributor set includes site 5, site 6
```

The system uses high performance, target-mediated DMA capable of running at 200MB/s.

Up to 512MB of DRAM may be assigned for AWG pattern memory.

14.3.1 Example load from file on local disk

```
nc localhost 54201 </mnt/local/raw400000_32
```

14.3.2 Example load/verify from a remote host

```
nc < bigawgfile UUT 54201
nc UUT 54200
# shows sha1sum of loaded file.
# compare with sha1sum of local file to validate..
sha1sum  bigawgfile
# sha1sums will match provided “bigawgfile” is padded out to next
1MB boundary
```

14.3.3 Trigger the Waveform

Waveform will trigger on the trigger condition for the Master site.

eg : AO424 in site 1, Master site=1,

set.site 1 trg=1,0,1 # external trigger

eg : AO424 in site 2, Master site = 2,

set.site 2 trg=1,0,1 # external trigger.

14.3.4 Warnings for use

1. Once the AWG is loaded, it MUST run to completion.
2. AWG and AI interact. In a system with concurrent AO and AI, the AI process must end AFTER the AO process. Ideally the AI capture should be at least 20% longer than the AWG.
3. In a mixed AI, AO system, partition the AO from the AI by setting:

```
/sys/module/acq420fmc/parameters/distributor_first_buffer
```

15 ACQ480 Special Features

ACQ480 runs significantly faster than the other ACQ400 series modules. It uses a different bus structure and a different DMA engine. The ADS6294 device also has a number of DSP features, and there may be switchable 50R termination.

15.1 Switched 50R termination

Control per channel

```
T50R_1
T50R_2
T50R_3
T50R_4
T50R_5
T50R_6
T50R_7
T50R_8
```

Global control

```
T50R

T50R 0      # high impedance, default
T50R 1      # 50 ohm set.
```

15.2 Source Synchronous Clocking

The ADS5294 provides data on LVDS lanes that need to be individually trained. The ACQ400 software autodetects when training is required and runs a training cycle (about 1s per site) on ARM. The firmware will detect and error out if the training is lost during the capture.

- Training is lost when the clock is interrupted or modified.

ACQ480FMC for use on ACQ1001 also features a “jitter cleaner” chip, this is also auto configured at start of capture. ACQ2106 has a more advanced clock and so does not need the jitter cleaner; however if the jitter cleaner is present, it will be set correctly.

15.3 Valid Clock Rates.

Minimum clock rate for ADS5294 : 10MHz. Maximum clock rate on ACQ2106 : 50MHz; ACQ1001FMC will be able to support an 80 MHz clock rate. Note that the wire clock rate is the sample clock divided by any decimation enabled on the ADS5294.

15.4 DSP Features

15.4.1 FIR Filter

ADS5294 has a number of FIR filter settings.

Please note ALL channels in a system should be set to have the SAME FIR filter (or at least, the same DECIMATION). By convention, FIR:01 sets all the channels in the site.

```
ACQ480:FIR:01 VALUE

where VALUE is one of
DISABLE
LP_ODD_D2
HP_ODD_D2
LP_EVEN_D4
BP1_EVEN_D4
BP2_EVEN_D4
HP_ODD_D4
CUSTOM_D2
CUSTOM_D4
CUSTOM_D8
CUSTOM_D1
```

15.4.2 Other Filters

(probably not useful for the general user).

```
acq480_setHiPassFilter
acq480_setLFNS
```

15.4.3 Gain Controls

```
ACQ480:GAIN:01
ACQ480:GAIN:02
ACQ480:GAIN:03
ACQ480:GAIN:04
ACQ480:GAIN:05
ACQ480:GAIN:06
ACQ480:GAIN:07
ACQ480:GAIN:08

acq480_setInvert
```

16 DIO432 Special features

- Byte programmable directions

```
set.site N byte_is_output X1,X2,X3,X4

where N is the location of the module and
X is 1 for output and 0 for input
X1 represents bits 0..7, X2: 8..15, X3: 16..23, X4: 24..31

example:
set.site 5 byte_is_output 1,1,0,0
# set two output, two input bytes on module in site 5
```

- Select Clocked or immediate mode

```
set.site N mode M

where N is the location of the module and
M is 1 for IMMEDIATE and 2 for CLOCKED.

example
set.site 5 mode 2
# set module in site 5 to CLOCKED mode.
```

- In immediate mode, write to DO32, read from DI32

```
set.site N DO32 hexval
get.site N DI32 hexval
```

16.1 Digital Pattern Generator DPG

The DIO432 may be configured as a digital pattern generator

- Clocked output word, 32 bits per module.
- Clock rate max 1MHz
- Pattern memory: max 512MB, ie 128s at 1MHz
- State Transition List STL for efficient pattern generation
- Supported configurations:
  - ACQ1001+1xDIO432 : 32 outputs.
  - ACQ1002+2xDIO432 : 64 outputs.
  - ACQ2106+NxDIO432 : up to 192 outputs.

16.1.1 DPG Theory

The DIO432 uses the ACQ400 Arbitrary Waveform Generator capability:

- Group 1 or more DIO432 in a “Distributor Set”
- Play out data from pattern memory, one sample per clock
- The data is the raw pattern, so for 32 bits at 1MHz, a 4MB buffer lasts

one second. The Raw pattern memory can be expanded to all of the data capture DRAM (512MB).

- A compact State Transition List STL notation is used. This is to make it easy for clients to generate huge raw pattern memory sets from a compact textual definition.

16.1.2 STL

The STL is a simple text description:

COUNT-UNTIL, NEWSTATE

The DO function is clocked at a fixed rate, when the counter reaches COUNT-UNTIL, NEWSTATE is output.

COUNT-UNTIL : is a clock count 0..4e9

NEWSTATE is a 32 bit hex value, suitable for a single DO32.

Multiple word outputs are possible.

eg for 64 bit:

0,0x00000000,0x00000000

1234,0xabcdef12,0x3456789a

16.1.3 DPG Services

The DPG feature offers a number of TCP port services:

Port	Description
4501	Loads RAW data to DPG
4511	Downloads RAW data for verification
4521	Load DPG from STL
4531	Provides SHA1SUM of expanded STL for verification.

Example script accesses the services:

```
acq1001_124> cat /usr/local/CARE/load.DO.test
STL=${1:-/mnt/local/stl-10s-p1}
nc localhost 4521 < $STL
nc localhost 4531
```

16.1.4 custom-dpg package

The DPG function is enabled by the custom-dpg packages

```
README for package: custom_dpg

Implements a DIGITAL PATTERN GENERATOR on a DIO432 in site 1
Possibility for additional sites

DPG may be programmed in any of 3 ways:

1. RAW data pattern : 4 bytes per clock, 4MB/s at 1MHz update
2. STL State Transition List : COUNT_UNTIL, NEW_STATE
3. DSD Digital Signal Definition : per-line waveform definition
[TBD]

Control by connecting to sockets at fixed ports:

4501 : write raw DPG data for site 1 here
4511 : read back the raw DPG data
4521 : write STL definition for site 1 here
4531 : read back the STL definition for site 1 here
4541 : write the DSD for site 1 here [future]
4551 : read back the DSD for site one here [future]

Boot time customisation for 1xDIO432:

cat /mnt/local/acq420_custom
DRVR_CUSTOM="default_dma_direction=1 xo_use_bigbuf=1
dio432_rowback=0"

cat /mnt/local/rc.user
set.site 0 SYS:CLK:FPMUX=FPCLK
set.site 1 clk 1,0,1
set.site 1 trg 1,0,1
set.site 1 byte_is_output 1,1,1,1
echo 1 > /sys/module/acq420fmc/parameters/xo_use_distributor
play0 1
```

16.1.5 dpg_abort

A DPG may be cancelled as follows

```
set.site 1 dpg_abort 1
# poll for run=0
get.site 1 run
0
set.site 1 dpg_abort 0
```

17 DIO482 Special Feature: LIVE_TOP

Live TOP : Live Time Of Pulse: capture at high speed recording times of first transition on each input bit.

Custom Package: 99-custom_livetop

17.1 Configuration

Example: assume 10MHz ext clk on HDMI, ext trg on HDMI:

```
/mnt/local/rc.user
# livetop test: clk and trg on HDMI
(/usr/local/epics/scripts/wait_ioc_ready; sleep 1
set.site 0 SIG:SRC:CLK:0 HDMI
set.site 0 SIG:SRC:TRG:0 HDMI
set.site 1 clk=1,0,1
set.site 1 clkdiv=1
set.site 1 trg=1,0,1
) &
```

17.2 Customisation:

- FINISHED=mask : drop out when the bits in mask have been set.
  - for example, unit is monitoring CH05..CH08
- TIMEOUT=secs : maximum time to run
  - eg 120s

```
cat /mnt/local/sysconfig/livetop.conf
export FINISHED=0x000000f0
export TIMEOUT=120
```

17.3 Operation:

nb: using nc directly is probably unwise, since it will not terminate.

```
nc acq1001_294 53998
0,0,0,0,20000044,40000044,60000044,80000044,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0
```

D-TACQ recommends run_livetop.py from acq400_hapi_tests.

18 Server Port Reference

Commonly used ports in **BOLD**.

18.1 Peers and Groups

With a multiple board set, it may be tedious to control the same knobs on each site in turn. Systems that ship with multiple modules of the same type, include a PEER group where commonly used knobs are set on site 1 and the change is automatically copied across all sites.

Similarly, modules like ACQ435 have a large number of similar knobs in the same site (Gain control for each channel). Collecting all the channel gain controls into a GROUP allows a single group knob to make the same setting on all the channel knobs.

18.2 Standard Server Ports

2222/tcp	acq4xx-epics-console	
2223/tcp	acq4xx-aimonitor-console	
2224/tcp	acq4xx-mdsshell-console	
2225/tcp	acq4xx-transient-console	
2226/tcp	acq4xx-nowhere-console	
2235/tcp	acq4xx-transient-log-console	Transient status
4210/tcp	**stream from aggregator**	
4220/tcp	**site 0 knobs**	
4221/tcp	**site 1 knobs**	
4222/tcp	**site 2 knobs**	
4223/tcp	**site 3 knobs**	
4224/tcp	**site 4 knobs**	
4225/tcp	**site 5 knobs**	
4226/tcp	**site 6 knobs**	
...		
42211/tcp	White Rabbit	
42212/tcp	mgtB	
42213/tcp	mgtA	
42214/tcp	dsp	
4236/tcp	site 16 knobs (special case)	
4240/tcp	**bos** : Big One Shot : high speed raw transient	***DEPRECATED***
4241/tcp	**crb** : Concatenate Raw Buffers: output all raw data	DEPRECATED
4501/tcp	load raw data to AWG	
4511/tcp	read AWG memory back	
4521/tcp	load STL to memory	
4531/tcp	read to get sha1sum of expanded RAW memory	

4541/tcp	GPG STL	Load GPG
4543/tcp	GPG_DUMP	View GPG
450[123456]1	Site 1..6 WRPG GPG _STL	Load S1 GPG
450[123456]1	Site 1..6GPG STL recap	Review S1 STL
450[123456]1	Site 1..6 WRPG GPG_DUMP	Dump S1 GPG
45072/tcp	BOLO8_CAL	Calibrate Bolo8
53000/tcp	DATA0 : transient all raw data	
53001/tcp	DATA1: CH01 data for transient	
..		
53196/tcp	DATA196: CH196 data for transient	
53990	MGT ACTION	MGT server.
53998	LIVETOP	
53999	ONESHOT (HIL Control)	
54200	AWG checksum	
54201	AWG one shot	
54202	AWG one shot, auto-rearm	
54203	ONE SHOT CONCURRENT LOAD	
54205	CONTINUOUS	
54206	CONTINUOUS, CONCURRENT LOAD	
54200	SHA1SUM	
54204	DUMP	

18.2.1 Transient Log Console Format

The transient log console is an efficient way for an external client to track state changes. The status information is provided as a set of 5 numbers on a single line. Client programs can filter it like this:

```
[pgm@hoy4 ~]$ nc acq1001_127 2235 | grep '^[0-9] '
0 0 0 0 0
1 0 0 0 0
3 0 0 0 0
3 0 100000 524288 0
4 0 100000 1310720 0
4 0 100000 1310720 1
5 0 100000 1310720 0
0 0 100000 1310720 0
```

The numeric fields are defined like this:

Field	Description	Values
0	STATE	0: "IDLE" 1: "ARM" 2: "RUN_PRE" 3: "RUN_POST" 4: "POST_PROCESS" 5: "CLEANUP"
1	PRECOUNT	Samples in pre buffer
2	POSTCOUNT	Samples in post buffer
3	TOTALCOUNT	Total samples so far
4	DEMUXSTATUS	Post process status indicator.

18.3 Channel Data Server Ports

Read channelized capture data direct from 53000+CH, example 32 channels.

53000/tcp	run.transient.service/gash (deprecated)
53001/tcp	cat dev/shm/transient/ch/01
53002/tcp	cat dev/shm/transient/ch/02
53003/tcp	cat dev/shm/transient/ch/03
53004/tcp	cat dev/shm/transient/ch/04
53005/tcp	cat dev/shm/transient/ch/05
...	
53032/tcp	cat dev/shm/transient/ch/28

18.4 Spy services

53666	slowmon (subrate raw data)
53667	spy (full rate raw data)

18.5 Custom_awg Server Ports

18.5.1 Raw multiplexed data to limit of DRAM

Port	Function
54000/tcp	load raw multiplexed data here to limit of DRAM
54200/tcp	read back a sha1sum of the raw data

18.5.2 Channelized data (limited length)

Send channelized raw AWG data direct to these ports:

54001/tcp	stdin2file /usr/local/awgdata/ch/ch01
54002/tcp	stdin2file /usr/local/awgdata/ch/ch02
54003/tcp	stdin2file /usr/local/awgdata/ch/ch03
54004/tcp	stdin2file /usr/local/awgdata/ch/ch04

19 MDSplus

D-TACQ devices interface with the MDSplus data system.

19.1 MDSplus devices

D-TACQ provides a range of MDSplus Devices for conventional control, monitoring and data archive from an MDSplus host.

19.2 Thin Client

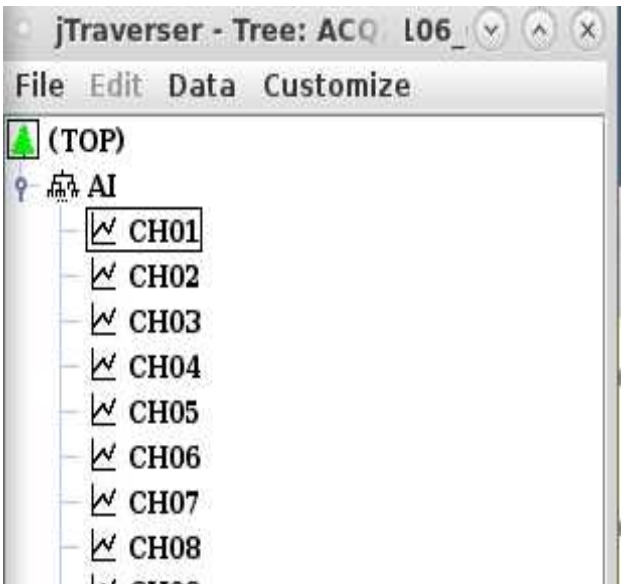
The MDSplus and thin client is an aleternative to send data direct to an MDSplus server.

19.2.1 Pre-requisites

19.2.1.1 Configure MDSplus tree on host

Tree creates using make_acqtree. Example structure for tree name “acq2106_085” :

```
make_acqtree --aichan=16 acq2106_085
```



19.2.1.2 Customise firmware

```
mv /mnt/packages.opt/70-mdsshell* /mnt/packages
```

19.2.1.3 Review post-shot script

Copy template at /usr/local/CARE/mdsputch-postshot-example to /mnt/local/postshot

```
#!/bin/sh
MDSHOST=andros
HN=$(hostname)
mdsConnect $MDSHOST
mdsOpen ${HN}
mdsPutCh -b 1 --site=0  --field=AI.CH%02d --expr %calsig :
# use next line if NCHAN > 99
#mdsPutCh -b 1 --site=0  --field=AI.CH%03d --expr %calsig :
mdsValue setEvent\(\'${HN}_done\',42ub\)
mdsClose
mdsDisconnect
```

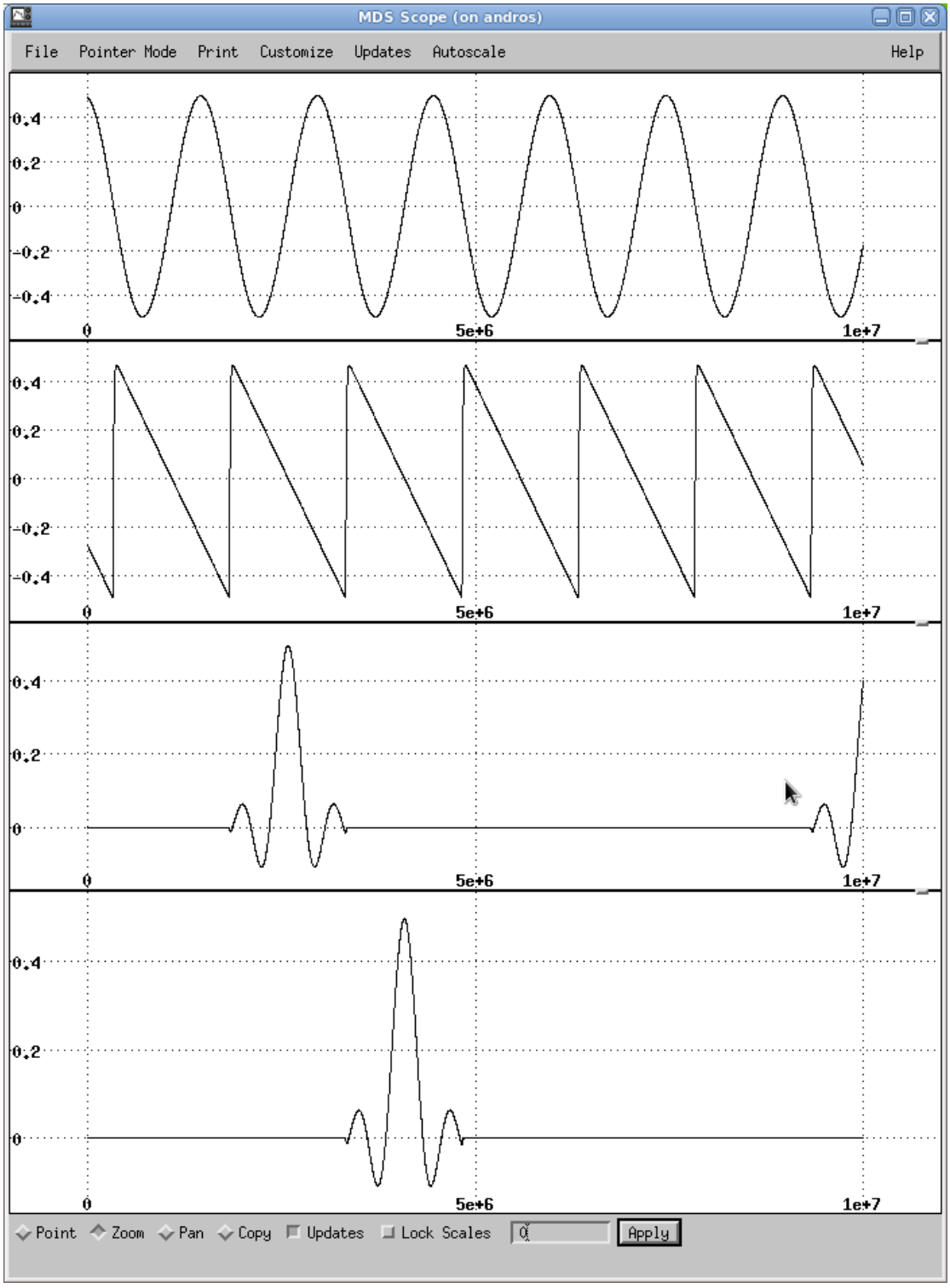
nb: CHnnn for channel count > 99, else CHnn

19.2.2 Run The Shot

For example, we configure an AO420FMC AWG in SITE2 to loopback to an ACQ420FMC digitizer in SITE1. Add this to `/mnt/local/rc.user` to make a turnkey system:

```
acq1001_017> cat /mnt/local/rc.user
cp /usr/local/CARE/acq400_streamd.0.conf-soft_trigger
/etc/sysconfig/acq400_streamd.0.conf
# mdsplus demo

# site 2 has an AO420FMC : else use a signal generator
## set the AWG to trigger on soft trigger, use default clocking
set.site 2 trg=1,1,1
## specify some waveforms
set.site 2 wavegen --loop 1 1=sin.dat 2=saw.dat 3=pulse1.dat
4=pulse2.dat

# site 1 has an ACQ420FMC digitizer
## set to trigger off soft trigger, use default clocking
set.site 1 trg=1,1,1

# configure the capture globally
set.site 0 run0=1
set.site 0 soft_transient 20000

# repeat the final command to run another shot.
```

19.2.3 Plot the data



20 Fault-monitor example

Typical example:

1. ACQ1002R + 2 x ACQ435ELF-32 : 64 channels, 128kSPS, 24 bit
2. ACQ1001Q + 1 x ACQ430ELF-8 : 8 channels, 128ksSPS, 24 bit.

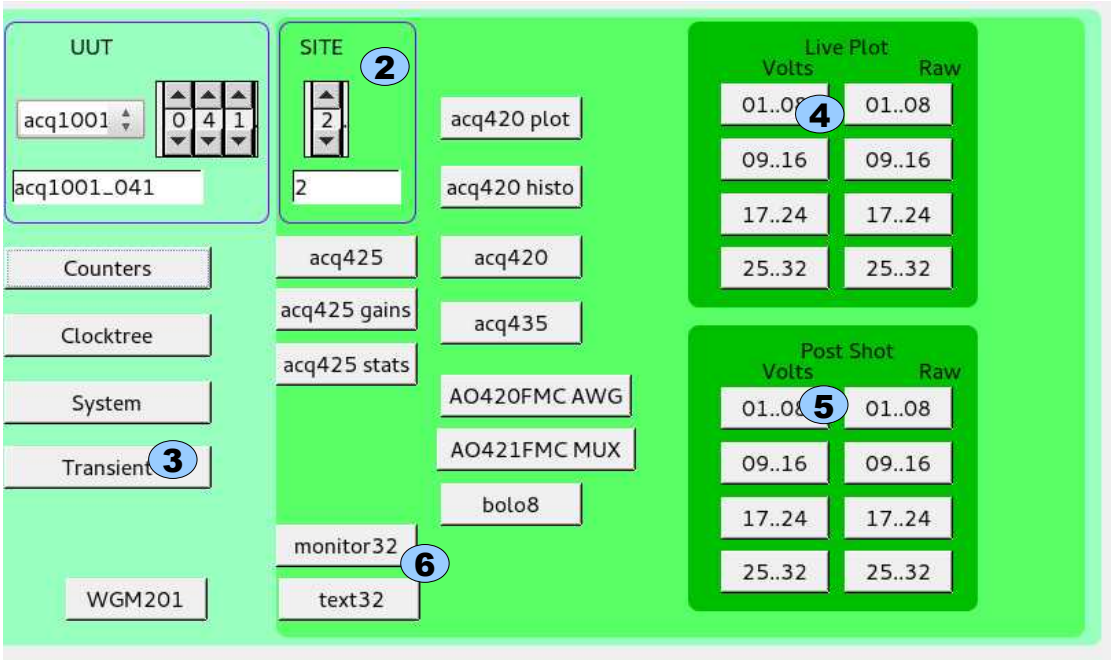
20.1 Features

1. Unit is a networked EPICS IOC
2. Capture runs continuously at full rate to local DRAM configured as a circular buffer.
3. During the capture, subrate data is published as an AI record per channel, typically at 10Hz.
4. During the capture, snapshots of full rate data are published as waveform (WF) records.
5. The use can specify a trigger condition (currently, this is a front panel digital input), PRE samples and POST samples.
6. The system continuously monitors the trigger input, and once the trigger occurs, continues for POST samples, then stops, publishing the PRE+POST data as a full-rate WF records.

20.2 Example CSS GUI

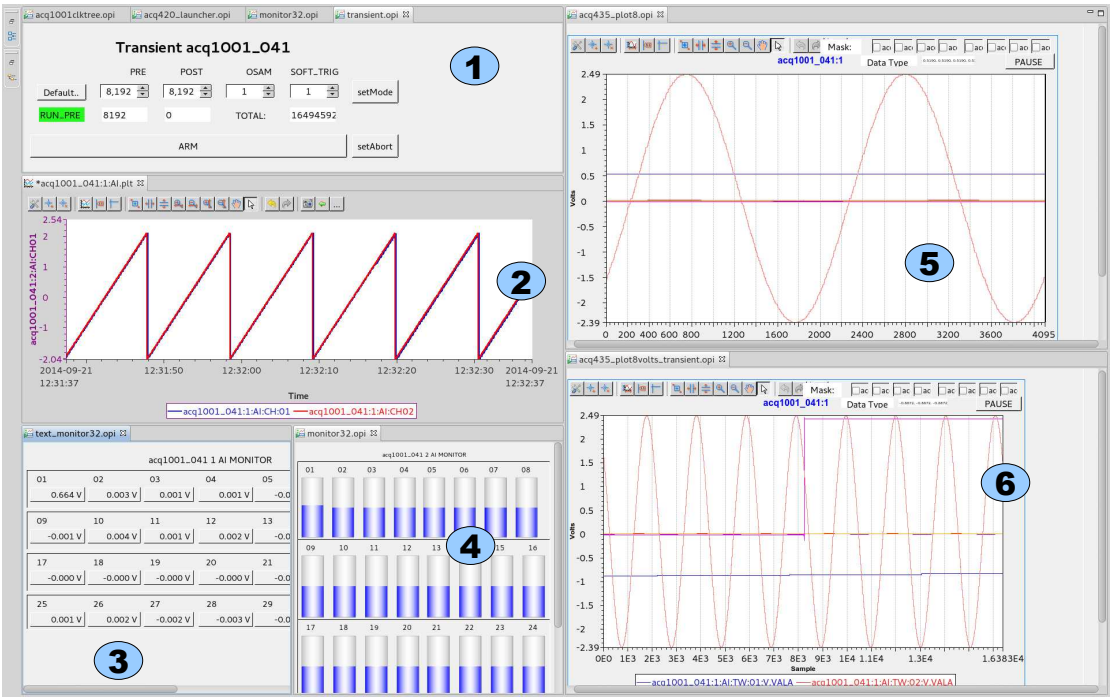
20.2.1 Launcher

- Always use the Launcher to run other .opi's. The Launcher allows you to specify the UUT (eg acq1001_041) and, where required, the module SITE number.



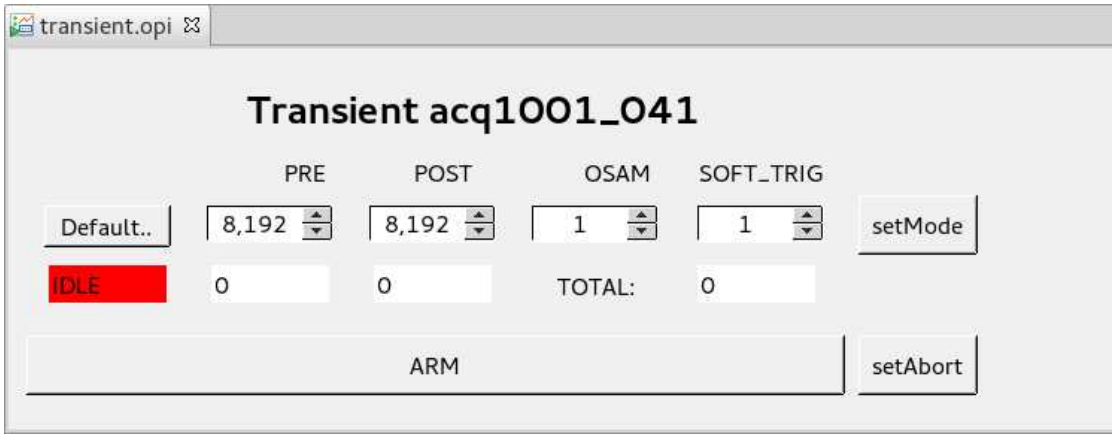
- Select UUT : model and serial to create unique PV prefix (1)
- Select SITE to use with module-specific OPI's in dark green (2)
- Control Fault Monitor from the Transient OPI (3)
- Live full-rate snapshot/scope plot (4)
- Post shot full-rate pre/post scope display. (5)
- Live monitors for subrate data. (6)
- Subrate data best shown on a stripchart.

20.2.2 Typical Multi-opi display

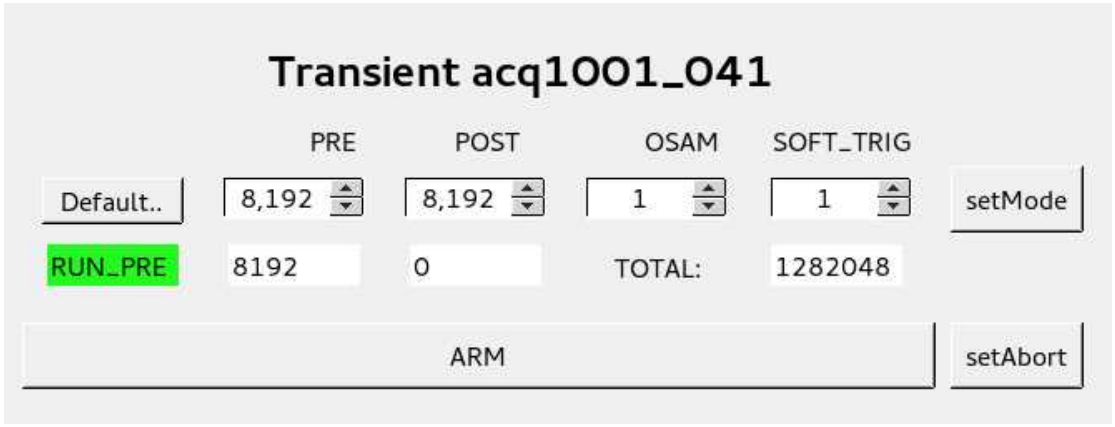


1. Transient controller showing status
2. Subrate Strip-chart
3. Subrate text display
4. Subrate tank display (good for wiring check)
5. Full rate live-scope
6. Full rate pre-post post-mortem (from previous shot).

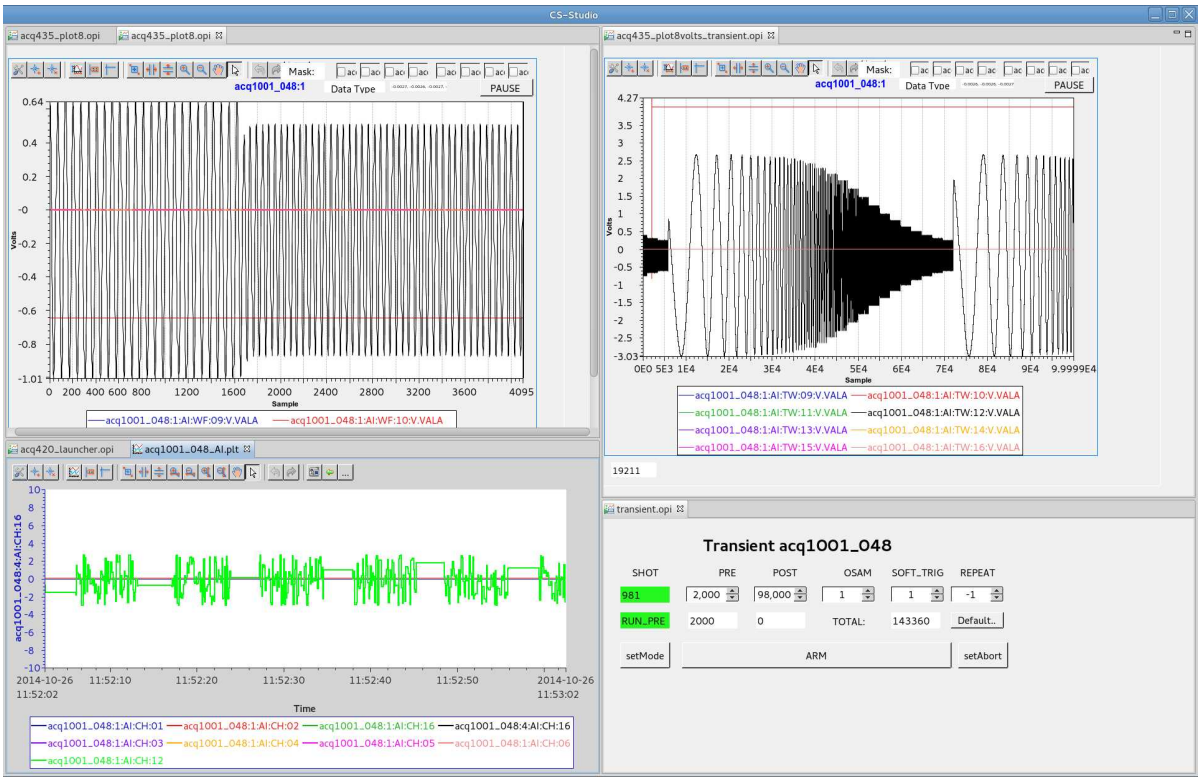
20.2.3 Transient GUI



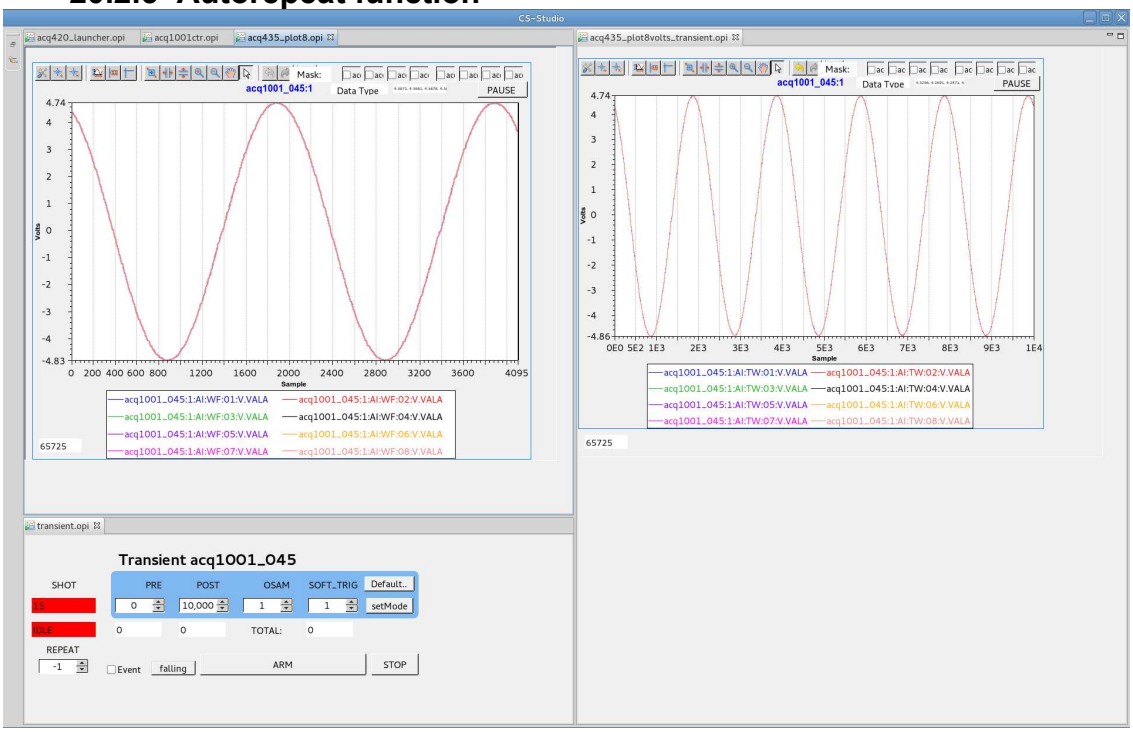
- Press “Default” to set sensible values.
- Optionally, modify PRE, POST (PRE+POST <= 16384) and SOFT_TRIG.
- Press “setMode” to commit to the unit
- Press ARM to start, setAbort to abort.
- Live status update during capture:



20.2.4 Slow rate strip plot, with scope and post-shot display



20.2.5 Autorepeat function



20.3 EPICS record reference

Abbreviated. For full reference, ask D-TACQ.

PV NAME	Type	FUNCTION
${UUT}:${SITE}:AI:CH:${CH} acq1001_041:2:AI:CH:01	AI	Single channel Subrate data, scalar value updated at 10Hz
${UUT}:${SITE}:AI:WF:${CH}:V acq1001_041:1:AI:WF:01:V	WF	Single Channel Live Snapshot waveform in volts
${UUT}:${SITE}:AI:TW:${CH}:V acq1001_041:1:AI:TW:01:V	WF	Single Channel Transient Waveform in VOLTS
${UUT}:MODE:TRANSIENT:PRE	longout	Specify PRE samples
${UUT}:MODE:TRANSIENT:POST	longout	Specify POST samples
${UUT}:MODE:TRANSIENT:OSAM	longout	Set !=0 for subrate data
${UUT}: MODE:TRANSIENT:SOFT_TRIGGER	longout	Set ==1 for soft trigger
${UUT}:MODE:TRANSIENT	longout	Set to commit above values
${UUT}:MODE:TRANSIENT:SET_ARM	longout	Start a capture
${UUT}:MODE:TRANSIENT:SET_ABORT	longout	Abort a capture
${UUT}:MODE:TRANS_ACT:PRE	longin	Monitor PRE sample actual
${UUT}:MODE:TRANS_ACT:POST	longin	Monitor POST sample actual
${UUT}:MODE:TRANS_ACT:TOTSAM	longin	Monitor elapsec sample actual
${UUT}:MODE:TRANS_ACT:STATE	mbbi	Monitor state actual
${UUT}:0:SIG:CLK_S1:FREQ	longin	Site 1 source clock Hz
${UUT}:0:SIG:SYN_S1:FREQ	longin	Site 1 sample clock Hz

20.4 Data Still Available outside EPICS.

20.4.1 Subrate

Binary raw channel vector update at subrate:

```
ls -l /dev/shm/subrate
..           256 Sep 21 11:38 /dev/shm/subrate
```

20.4.2 Snapshot

Binary channelized data, presented one file per channel, at about 1Hz:

```
acq1001_041> ls /dev/shm/AI.1.wf
CH01  CH04  CH07  CH10  CH13  CH16  CH19  CH22  CH25  CH28  CH31
CH02  CH05  CH08  CH11  CH14  CH17  CH20  CH23  CH26  CH29  CH32
CH03  CH06  CH09  CH12  CH15  CH18  CH21  CH24  CH27  CH30
```

This could be in “Dirfile Format” for direct plot by kst. eg from an NFS or SAMBA mount.

20.4.3 Post Shot

Post shot data is held at /dev/acq400/data/.

NB: this is a virtual file system mapping onto kernel buffers. This allows large files (because it doesn't take additional space eg on a RAMDISK), but it also means the data is erased immediately on the next shot.

```
acq1001_041> ls -l /dev/acq400/data/
total 0
drw-r--r--    1 root     root             0 Sep 21 10:35 0
drw-r--r--    1 root     root             0 Sep 21 10:35 1
drw-r--r--    1 root     root             0 Sep 21 10:35 2
drw-r--r--    1 root     root             0 Sep 21 10:35 raw
acq1001_041> ls -l /dev/acq400/data/1/
total 0
-rw-r--r--    1 root     root         65536 Sep 21 11:30 01
-rw-r--r--    1 root     root         65536 Sep 21 11:30 02
-rw-r--r--    1 root     root         65536 Sep 21 11:30 03
-rw-r--r--    1 root     root         65536 Sep 21 11:30 04
-rw-r--r--    1 root     root         65536 Sep 21 11:30 05
```

- /dev/acq400/data/0 : channelised mappings for all data in set (not used)
- /dev/acq400/data/[12] : channelised mappings per channel.
- /dev/acq400/data/raw : view of raw data (not valid after channelise process)

21 Big One Shot example

Example for 4xACQ425 System

```
cat /mnt/local/rc.user
# tweak up the analog rails to +/-13V
set.sys /dev/acq2006/vap 18
set.sys /dev/acq2006/van 18
#
cat - >/etc/acq400/1/peers <<EOF
PEERS=1,2,3,4
KNOBS=clkdiv,trg,rgm
EOF

set.site 1 trg=1,1,1
set.site 1 clkdiv=100
run0 1,2,3,4
transient POST=1000000 SOFT_TRIGGER=1
```

- clkdiv=100 :: 100MHz clock / 100 = 1MSPS sample clock
- trg=1,1,1 :: SOFT TRIGGER, PEER mechanism copies all sites.
- run0 1,2,3,4:: aggregate data from all 4 modules
- transient POST=1000000 SOFT_TRIGGER=1
  - equivalent to "acqcmd setMode SOFT_TRANSIENT 1000000".

We don't have an explicit “setArm”, instead connect a socket to port 4240 and read status:

```
nc acq2006_013 4240
0 0 0 0
1 0 0 0
3 0 0 0
3 0 8192 8192
3 0 16384 16384
3 0 32768 32768
...
3 0 999424 999424
4 0 1000000 1007616
0 0 1000000 1007616
```

nc connects to a TCP socket at port 4240 and listens
This is a combined "setArm" and "getNumSamples".
Because we set SOFT_TRIGGER=1, the system triggers automatically.
The fields are:
3 0 8192 8192
STATE PRE POST ELAPSED

Where

STATE: 0:STOP, 1:ARM, 2:RUN_PRE, 3:RUN_POST, 4:POSTPROCESS
PRE : only PRE=0 is supported at this time
POST : number of POST trigger samples so far
ELAPSED : total number of samples so far

After the shot, pull raw data from the host as follows:

```
nc acq2006_013 4241 | pv > bigfoot1M4
```

What happens here?
nc connects a TCP socket to port 4241 and reads data
pv is a convenience program to show upload state/speed

The raw data on disk can be demuxed and stored or displayed.

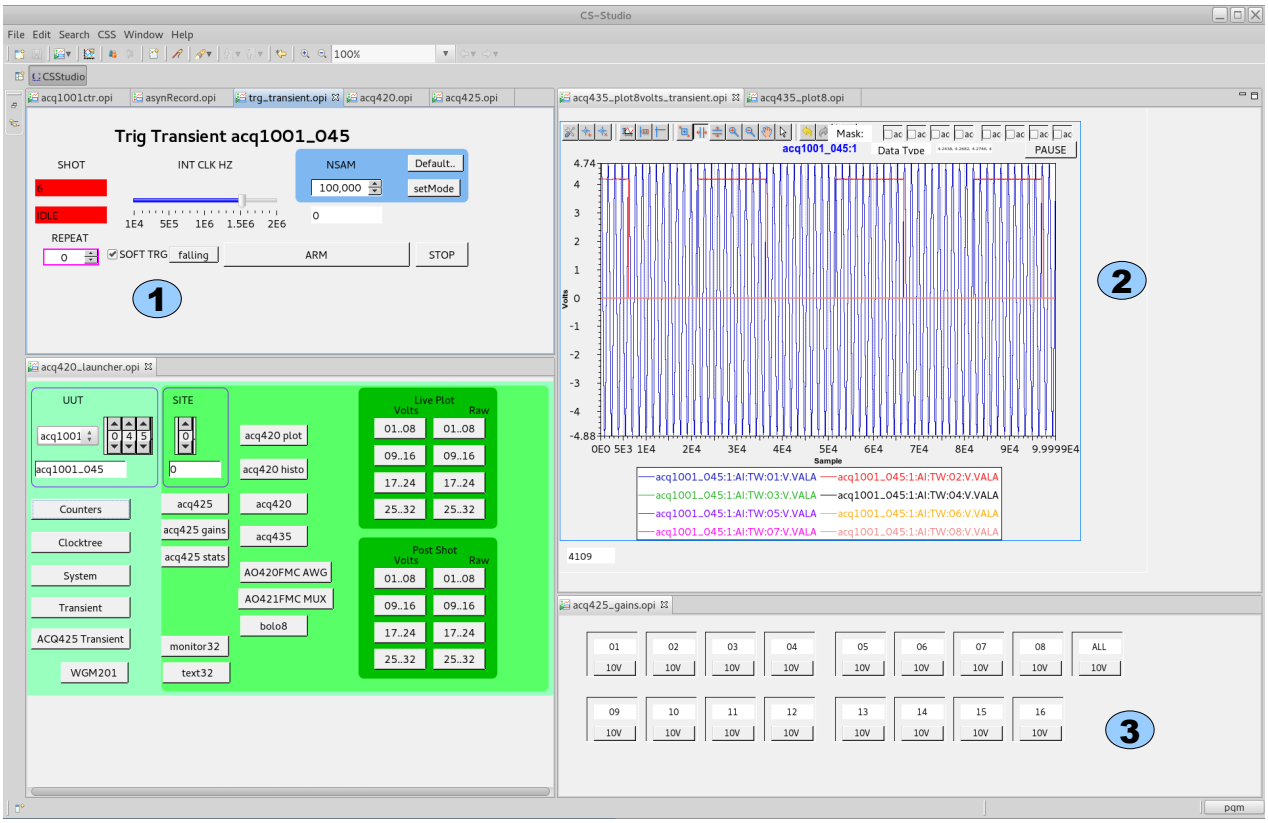
eg

```
acq_demux -a ../acq2006-4x425.def bigfoot1M4

Where the definition file is ..
[pgm@ahoy BOS]$ cat ../acq2006-4x425.def
ACQ=acq196
WORD_SIZE=2
AICHAN=64
```

22 Capture With ACQ425ELF

22.1 One Shot transient with auto-repeat

Configure a one-shot capture at up to 2MSPS with up to 8MS in memory, plot up to 100K points in CSS. nb: do not use with “external stream”.



1 First, press **`Default..`**, optionally change **`NSAM`**, then press **`SetMode`**

Then Control **`Arm/Stop CLK`**, **`Samples`**, **`Trigger`**.

`Repeat`=-1 means autorepeat forever.

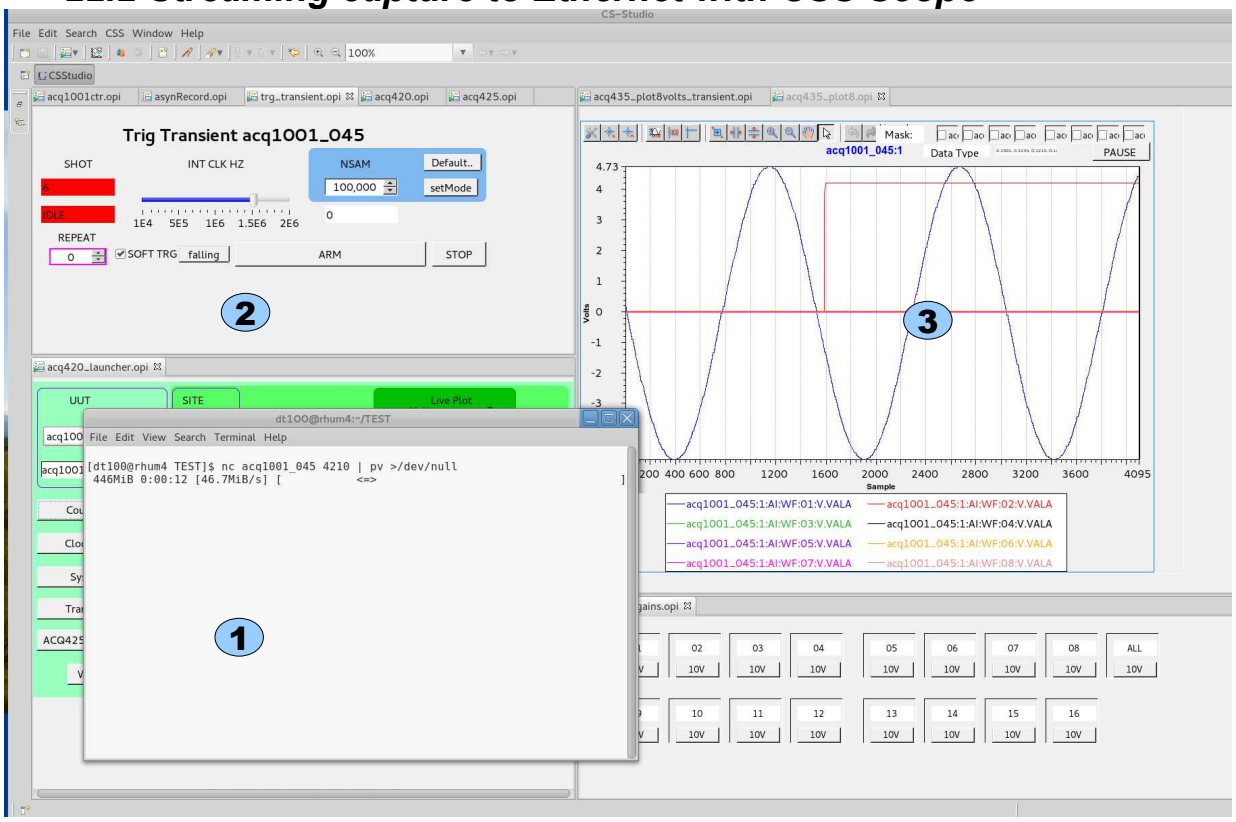
2 Plots 8x 100K points, post shot

To see the other 8 channel, select “**`Post Shot Volts 09..16`**” and arrange layout to suit.

After the shot, channelized data is available at ports 53001..53016.

3 Select gain per channel.

22.2 Streaming capture to Ethernet with CSS Scope



Stream to a remote computer at 16c x 1MSPS, control and view from CSS

1 Stream continuously, Linux or Windows (cygwin)

```
nc UUT 4210 | pv > /dev/null  # discard
nc UUT 4210 | pv > bigrawfile # raw to disk
```

2 Transient controls unused, apart from CLK rate.

Do NOT press "**Arm**"

3

Live Plot Volts: shows a scope display snapshot of data at full rate.

Press "**Live Plot Volts 01..08**" or "**Live Plot Volts 09..16**"

23 Full Rate streaming with ACQ2106

ACQ2106 features 4 MGT ports. Currently, two ports are supported.

The ports are rated at 6Gbps, with an achievable capacity of 400MBytes/s each. Each port is separated by a full-duplex bi-directional DMA channel. DMA processes can be configured for:

1. FARM: the SAME data is sent to each channel for distribution on fiber-optic.

2. SPLIT: data can be split between fiber optic channels. This is useful for connecting to the previous-generation AFHBA card, to increase data throughput.

The DMA channels can operate in

1. PUSH : Analog Input to HOST and

2. PULL : fetch Analog Output from HOST.

Note also that the FPGA-> A9 -> DRAM path is still active. The full data rate can be too high for the A9 to handle, so this path includes programmable decimation. A fully loaded system is then able to stream full rate data out of the MGT ports, while collecting sub-rate data in the A9 system for presentation on Ethernet.

23.1 Host Side Driver

Install host-side driver on HOST OS. We recommend linux-rt 3.10

23.2 Example Test procedure

TBA

23.3 Web Diagnostic

TBA

23.4 CSS Diagnostic

TBA

24 White Rabbit Endpoint

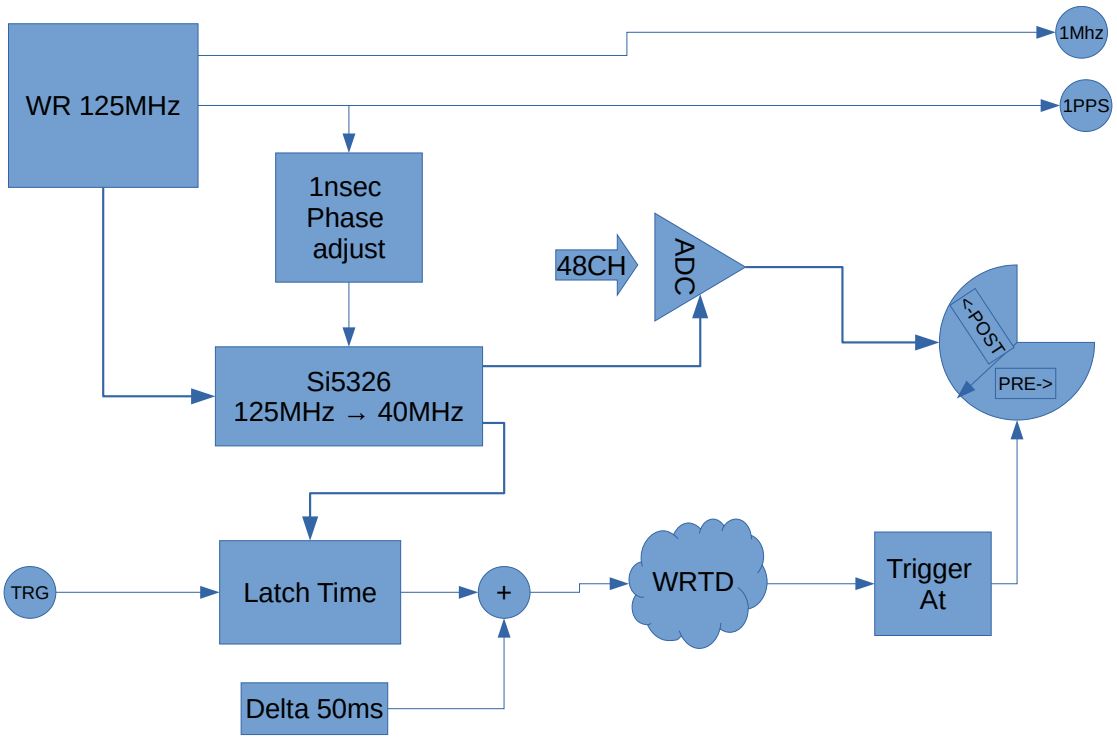
24.1 What is White Rabbit?

White Rabbit is a precision implementation of IEEE1588 network timing.

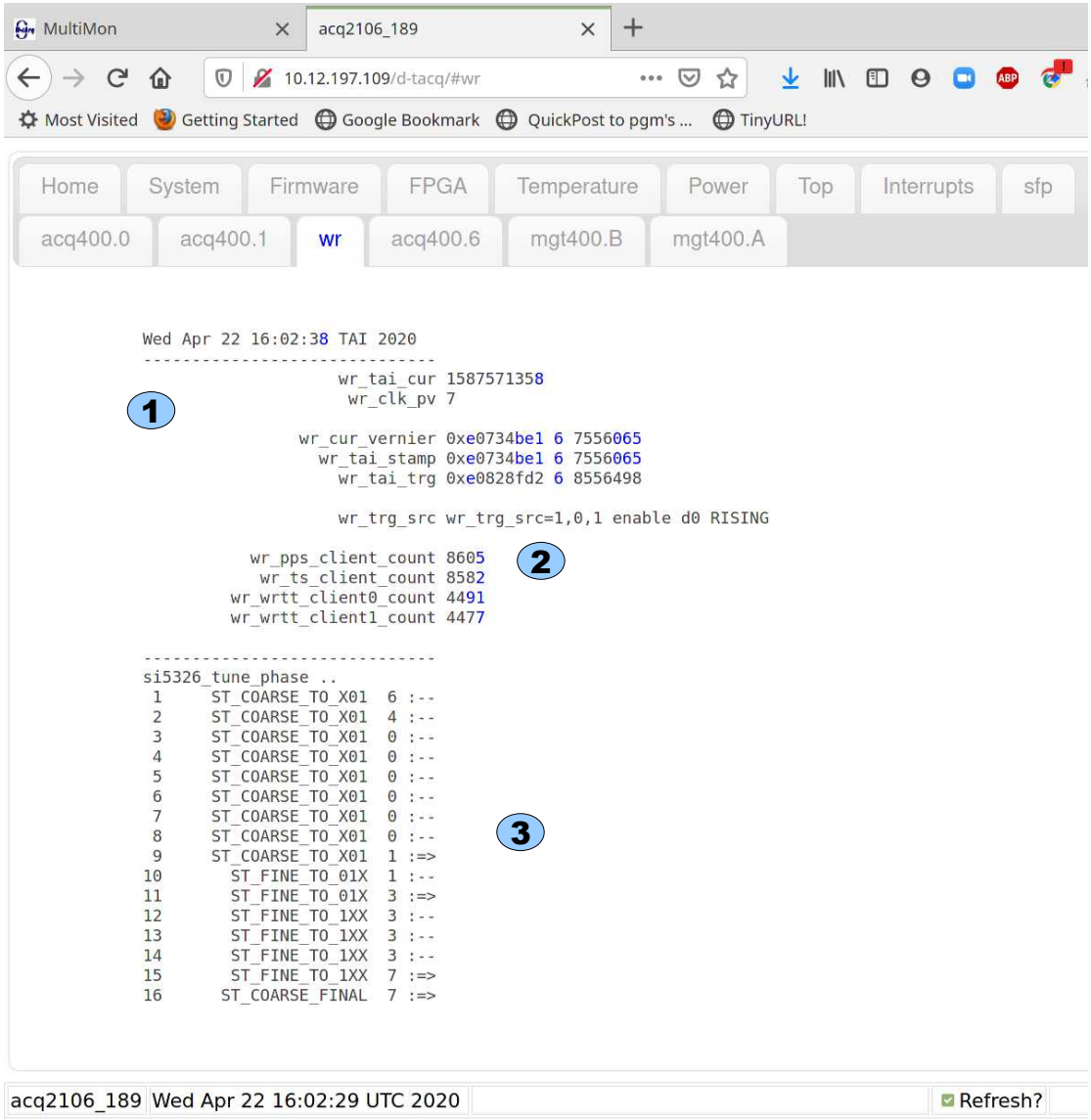
ACQ2106 supports the White Rabbit endpoint function, connecting to a White Rabbit Switch via a fiber-optic link from SFP port “C”. What this provides:

- All nodes share the same master clock with the same frequency and phase. This is regardless of the number of nodes and the distance between them.
- All nodes have the same understanding of time, and data can be timestamped with TAI time, to accuracy better than the sample rate.
- The ACQ2106 sample clock, at any rate 1-80MHz can be phase adjusted to exactly match the common White Rabbit clock.
- ACQ2106 can be triggered by trigger messages using WRTD: White Rabbit Time Distribution. The ACQ2106 supports two independent Global WRTD triggers, WRTT0 and WRTT1.
- ACQ2106 can sample an external trigger and send a corresponding WRTD trigger message.

24.2 Clock and external Trigger Block Diagram



24.3 Web Diagnostic



1 TAI Timestamp and current count: WR is up and working. TAI “Atomic Time”.

2 wr_pps_client_count : 1PPS output is up and running.

wr_ts_client_count : we have a TRG input connected and it is being sampled.

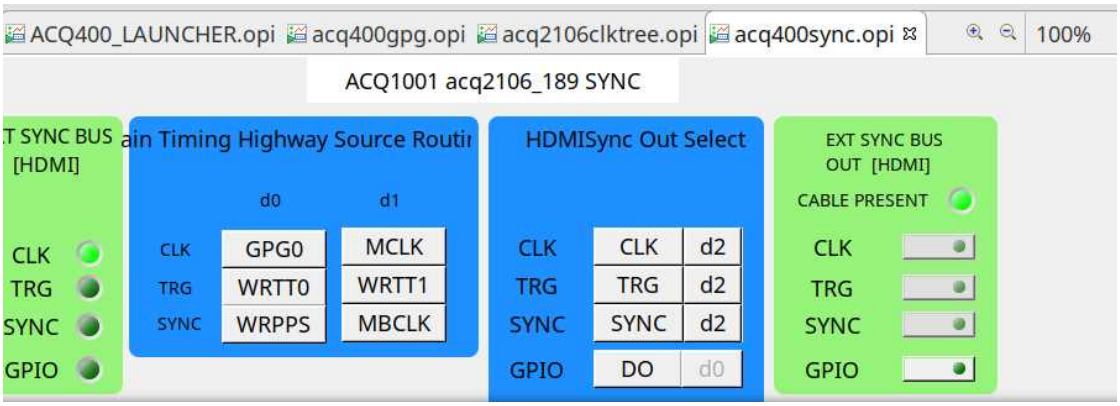
wr_wrtt_clientX_count : the WRTD system is responding to the input trigger and is working normally.

3 The procedure “si5326_tune_phase” has been run and the local clock is now phase locked with the WR clock.

24.4 Trigger Diagram

Bus TRG.d0 is sourced from WRTT0

Bus TRG.d1 is sourced from WRTT1



24.5 Configuration

The WRTD system is configured on boot using this file:

example has a local clock running at 20MHz.

```
acq2106_189> cat /mnt/local/sysconfig/wr.sh
# 20MHz
WRTD_TICKNS=50
WRTD_DELTA_NS=50000000
WRTD_VERBOSE=2
WRTD_RTPRIO=15
# local trigger, system trigger.
WRTD_RX_MATCHES=$(hostname),WRPG_DEMO
WRTD_RX_MATCHES1=test_wrtt1
WRTD_RX_DOUBLETAP=test_double_tap
WRTD_ID=WRPG_DEMO
WRTD_DELAY01=2000000
WRTD_TX=0
```

Each parameter is presented as a client-controllable knob in “Site 11”, a dedicated WRTD control service.

- WRTD_TICKNS=50 : 20MHz clock tick
- WRTD_DELTA_NS: time lag on distributed trigger, we recommend 50msec is safe for any LAN environment.
- WRTD_RX_MATCHES: match any of these triggers to initiate WRTT0
- WRTD_RX_MATCHES1: match any of these triggers to initiate WRTT1
- WRTD_RX_DOUBLETAP: match any of these triggers to initiate a “Double Tap, which is:
    1. WRTT0
    2. Delay WRTD_DELAY01 nsec.
    3. WRTT1
- WRTD_TX is disabled, run it programmatically.

24.6 API : Control knobs presented on Site 11

Knob	Description
MODEL	Standard FRU tracking
MTYPE	
SERIAL	
Si5326:TUNEPHASE:BUSY	Clock Tuning Status
Si5326:TUNEPHASE:OK	
WRTD_DELAY01	Make dynamic changes to defaults
WRTD_DELTA_NS	Delay to guarantee timely trigger
WRTD_ID	string:TX ID
WRTD_RX	1: enable RX 0: disable RX
WRTD_RX_DOUBLETAP	nsecs: set WRTT0, delay nsec, WRTT1
WRTD_RX_MATCHES	match1[,match2..]: matches for WRTT0
WRTD_RX_MATCHES1	match1[,match2..]: matches for WRTT1
WRTD_TICKNS	number of nsec in a clock tick
WRTD_TX	1: enable TX 0: disable TX
WRTD_VERBOSE	set debug level
WRTD_TX_MASK	set site selector mask 0:global only
tai_date	current date
wrtd_commit_rx	reset rx, loading new dynamic prams
wrtd_commit_tx	reset tx, loading new dynamic prams
wrtd_reset_rx	reset rx, use existing prams
wrtd_reset_tx	reset tx, use existing prams
wrtd_tx	run WRTD TX server in foreground
wrtd_tx_immediate	send immediate WRTD message
wrtd_txi	send immediate WRTD message
wrtd_txq	send quick WRTD message

24.6.1 Sequence

1. Set parameters WRTD_xx as required
2. Active the changes for the RX server or TX server with

- `wrtd_commit_tx=1  or`
- `wrtd_commit_rx=1`

24.6.2 RX server

The RX server listens for incoming WRTD messages.

The RX server compares incoming WRTD messages against a message filter and takes appropriate action. There is only ONE RX server instance per UUT.

24.6.3 TX server

The TX server blocks on a TAI_TIMESTAMP_LATCH register. This register latches TAI time on an external edge. The TX server responds by sending a

WRTD network message to request an action (WRTT) in the near future.:
WRTT_TAI = TAI_TIMESTAMP + WRTD_DELTA_NS.

24.6.4 TX Immediate

A TX Immediate packet can be sent at any time from software command.

The timestamp in the packet is

WRTT_TAI = TAI_TIME_NOW + WRTD_DELTA_NS

TX Immediate is provided primarily for subsystem test purposes.

- set.site 11 wrtd_txi 1 # send default packet once
- set.site 11 wrtd_txi CUSTOM # send packet “CUSTOM”

24.6.5 TX Quick

A TX Quick packet can be sent at any time from software command.

The packet contains a “Timestamp” code TAI_QUICK (0xFFFFFFFF) with special meaning:

When a WRTT trigger register is loaded with the TAI_QUICK it will trigger immediately.

TX Quick is seen as a kind of multicast Emergency Stop command.

24.6.6 WRTD Packet format

The WRTD packet conforms to CERN specification.

```
struct wrtd_message {
        unsigned char hw_detect[3]; /* LXI */
        unsigned char domain;       /* 0 */
        unsigned char event_id[WRTD_ID_LEN];
        uint32_t seq;
        uint32_t ts_sec;
        uint32_t ts_ns;
        uint16_t ts_frac;
        uint16_t ts_hi_sec;
        uint8_t flags;
        uint8_t zero[2];
        uint8_t pad[1];
};
```

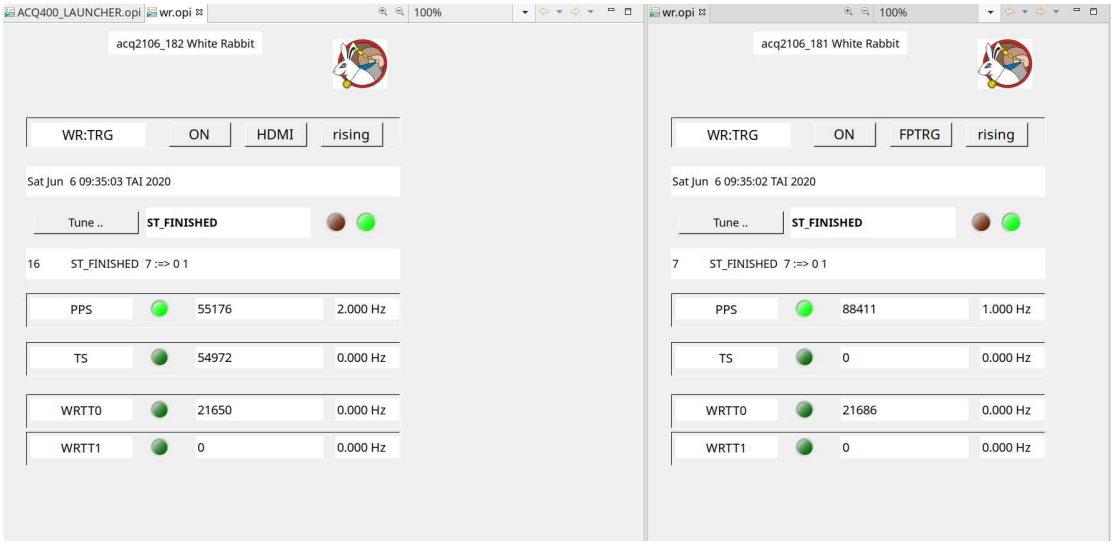
D-TACQ hardware makes use of the following fields in in this packet:

- event_id
- ts_sec, ts_ns

Event_id is used in two ways:

- Filter string: ascii text in the normal zero-terminated string sense eg
  - “ACQ400\0”
- MASK: the last byte in event_id, event_id[WRTD_ID_LEN-1] is treated as a binary encoded “Site select mask”
  - Default : 0x0, packet refers to the global triggers WRTT0, WRTT1
  - Non zero: each bit encodes a different trigger select. This applies to the TIGA appliance, where up to 6 sites support local WRTT controls
    - 0x01 : Global WRTT0
    - 0x02 : Global WRTT1
    - 0x04 : Local, Site 1
    - 0x08 : Local, Site 2 …
    - 0x80 : Local, Site 6
  - Multiple bits in the mask can be active at the same time, so that a single message packet can result in multiple triggers.
- For further information, please see the TIGA documentation.

24.7 OPI : WR specific GUI screen:



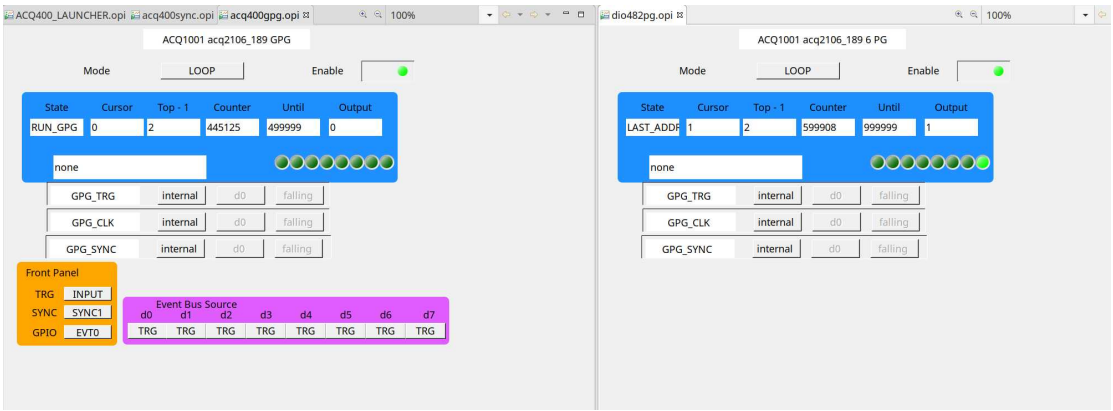
24.8 WRPG : White Rabbit Pulse Generator

WRPG is a programmable pulse generator, using the precision of WR clocking and the programmability of a hardware State Transition List.

Hardware: DIO482-PG : a DIO482 module with

- 4 digital outputs on LEMO-00
- Status LEDS
- Single digital input (TRG-IN)
- Single clock output (CLK-OUT)
- The digital outputs are programmed using STL at well known site specific sockets.
- A single ACQ2106 could host up to 6 DIO482-PG devices, for 6 independent pulse timers, or a mix of AI and DIO functions.

24.8.1 OPI



At the left, Site 0 global GPG, driving motherboard signal lines.
On the right, a Site 6 DIO482PG specific GPG (work in progress).

24.8.2 Services

Port	Description
450[123456]1	Site 1..6 WRPG GPG _STL
450[123456]1	Site 1..6 GPG STL recap
450[123456]1	Site 1..6 WRPG GPG_DUMP

24.8.3 Knobs

Knob	***Function***
Direct "GPIO" output	
DO32	DO32 pattern to output direct
DO32_immediate_mask	Selects direct outputs, NOTGPG (most of these will be LEDS
Pulse Generation Control	
GPG:ENABLE	GPG Enable
GPG:MODE	GPG Mode : ONCE, LOOP, LOOPWAIT
GPG_CLK	GPG CLK select: default WR01M
GPG_CLK:DX	GPG CLK Line
GPG_CLK:SENSE	GPG CLK Sense
Clock Output Control	
CLK	internal/external: only external is valid
CLK:DX	CLK.d0 .. CLK.d7
CLK:SENSE	rising/falling
CLKDIV	Divider 1..65535
Trigger Input	
TRG	enable/disable
TRG:DX	TRG.d0..d6 select global trigger lines as normal, excepting, site-specific TRGIN : replaces site dX, LOCAL front panel trigger WRTT0 : local WRTT, replaces d7, all sites.
TRG:SENSE	rising/falling

25 Timed Captures: VCR mode

ACQ1001 can be configured with one or two USB disks.

The system can be configured to duplicate raw capture data to each disk.

The capture can be scheduled to start at a given time.

25.1 Packages

/mnt/packages/39-transient-1502192115.tgz

/mnt/packages/34-at-1502112135.tgz

25.2 Custom Configuration

ls /mnt

acq1001_044> cat /mnt/local/sysconfig/transient.init

TCON=/usr/local/bin/stream2disks

MB=32000

VERBOSE=1

25.3 Scheduled start:

acq1001_044> echo set_arm | at 01:33

job 4 at Fri Feb 20 01:33:00 2015

25.4 Capture Log:

Feb 20 01:33:00 (none) user.notice stream2disks: start shot:5 MB:32000 /diska /diskb

Feb 20 01:33:00 (none) user.notice acq400_streamd: data_engine_0 0xf3004001 sites=none aggregator=1 on

Feb 20 01:33:00 (none) user.notice acq400_streamd: site0 sites=1

Feb 20 01:33:00 (none) user.debug acq400_stream[16387]: hello world B1004

Feb 20 01:33:00 (none) user.debug acq400_stream[16389]: G_aggsem:0xb6f42000

Feb 20 01:33:01 (none) user.notice soft_trigger: trigger start

Feb 20 02:58:35 (none) user.notice stream2disks: finished

25.5 Result

```
acq1001_044> df

/dev/sda1 31250016 31250016 0 100% /diska
/dev/sdb1 31250016 31250016 0 100% /diskb

acq1001_044> find /diska/ | head
/diska/
/diska/000
/diska/000/000000
/diska/000/000001
/diska/000/000002
/diska/000/000003
/diska/000/000004
/diska/000/000005
/diska/000/000006
/diska/000/000007
acq1001_044> find /diska/ | tail
/diska/305/030590
/diska/305/030591
/diska/305/030592
/diska/305/030593
/diska/305/030594
/diska/305/030595
/diska/305/030596
/diska/305/030597
/diska/305/030598
/diska/305/030599
```

ie 30500 x 1MB files on disk.

Offload the data using the network (rather than remove the USB sticks .. ftp server on acq1001 and recursive wget on the host..

25.6 Offload

acq1001_044> tcpsvd -vE 0.0.0.0 21 ftpd /diska

time wget -r ftp://root:password@acq1001_044

...

--2015-02-20 10:17:39-- ftp://root:*password*@acq1001_044/305/030599

=> \u201cacq1001_044/305/030599\u201d

==> CWD not required.

==> SIZE 030599 ... done.

==> PASV ... done. ==> RETR 030599 ... done.

[<=>] 0 --.-K/s in 0s

2015-02-20 10:17:39 (0.00 B/s) - \u201cacq1001_044/305/030599\u201d saved [0]

FINISHED --2015-02-20 10:17:39--

Downloaded: 30600 files, 30G in 34m 37s (14.7 MB/s)

25.7 Suggestion for improved accuracy

In VCR mode, with good NTP performance, the system will soft-trigger, with a jitter of about 1s.

D-TACQ recommends using a local GPS with a ONE PPS output. First, the GPS conditions NTP to ensure it's accurate (and works off the Net).

Second, the ONE PPS signal should be used as a hardware trigger. Software enables the capture during the second before the scheduled start: the capture then starts at the start of the designated second, with microsecond jitter.

26 Hardware In Loop HIL mode

HIL mode allows a remote client to

- load AWG
- run a shot
- offload AI data

Both the AWG and AI data are in RAW (demux) format, so the host computer is responsible for all data formatting. Capture is controlled as follows:

- pre-configure clocks, triggers, transient length
- load AWG : write data to port 54201
- run the shot : read text from port 53999 (ends “SHOT_COMPLETE”)
- read AI data from port 53000

27 MGT-DRAM-8 : Memory Expansion.

MGT-DRAM-8 is an 8GB memory expansion module for ACQ2106.

It replaces, and plugs into the socket used by MGT482.

MGT-DRAM-8 is able to store ADC data continuously at 1280MB/s, and is suitable for networked digitizers requiring long transients, providing a 16x expansion over the mainboard DRAM. After the shot, data is offloaded on Ethernet, with a typical offload rate of 20MB/s. Data offload can either be by:

- Host Pull: HOST pulls the data from ACQ2106.
- Target Push: ACQ2106 is an FTP client to HOST FTP server (deprecated)

A highly functional host side script is provided: mgtdramshot.py. The shot cycle can also be controlled from EPICS, or EPICS can be used to monitor control from the script. Capture can be controlled from EPICS, with automated data offload and store from the script.

MGT-DRAM-8 is NOT suitable for continuous capture applications – use MGT482 and the fiber-optic link for that.

MGTDRAM8 is paged in 4MB blocks, and all capture and offload operations work in units of blocks/buffers. Offload is in groups of 12 (channels multiple of 3) or 16 buffer (channels is a multiple of 4).

27.1 Installation

- Connect an antistatic strap and open the top of the ACQ2106 box (12 screws)
- Remove MGT482, if fitted.
- Carefully Fit MGT-DRAM, fasten with 2 screws.
- Fit Heatsink and clip.
- Replace the top.

27.2 Software bootup

The device-tree set from u-boot MUST be acq2106.dtb and NOT acq2106sfp.dtb:

```
grep dtb /tmp/u-boot_env
devicetree_image="dtb.d/acq2106.dtb"
```

If it's not set to the right value, it will be necessary to break into the u-boot prompt on boot and to change the u-boot environment variable.

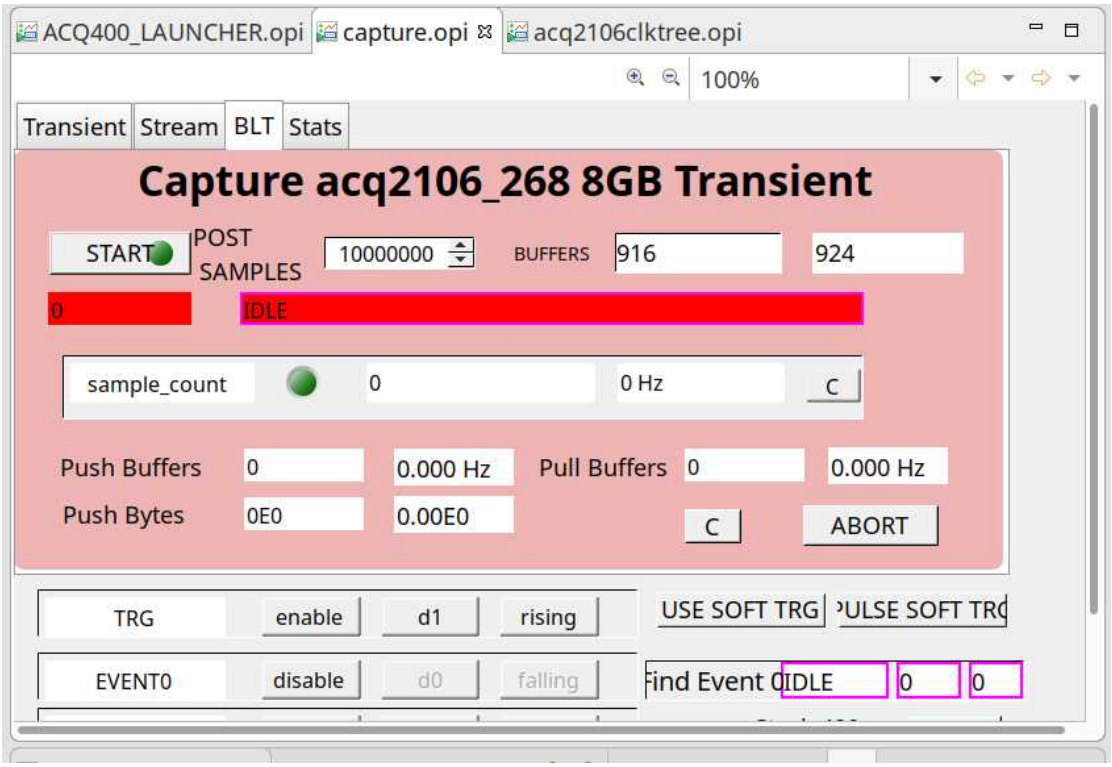
After that, MGT-DRAM is auto-detected and the software will “do the right thing”.

27.3 Control From EPICS

EPICS PV's are provided to manage and monitor the shot.

- acq2106_999:MODE:BLT:POST : set number of samples or
- acq2106_999:MODE:BLT:BUFFERS : set number of buffers
- acq2106_999:MODE:BLT:SET_ARM : set 1 to run the shot
- NB: while we show cs-studio GUI, this is not essential .. the control PV's exist and can be used regardless of UI choice.

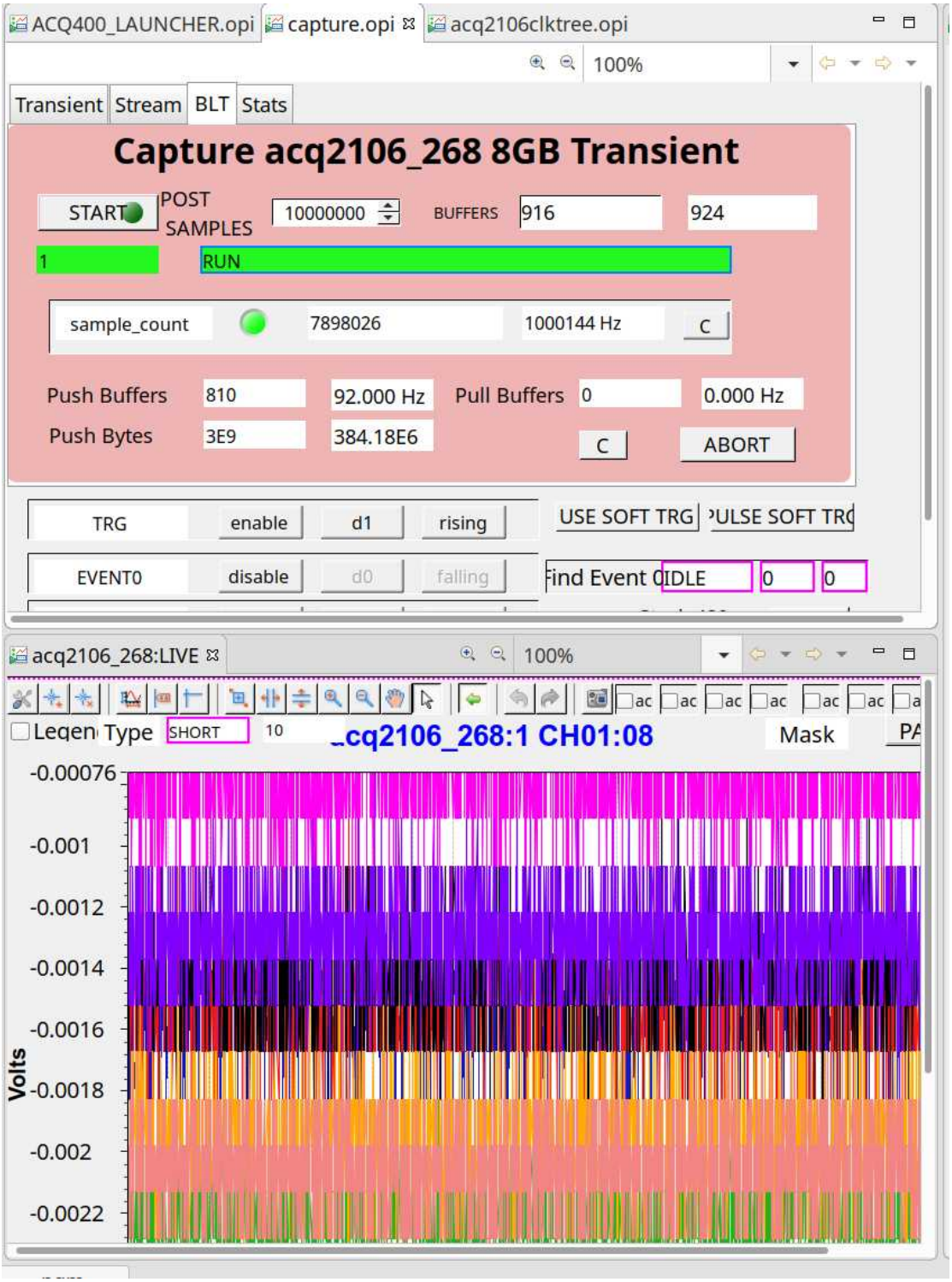
Control from EPICS, before start:



Showing POST=10,000,000, system calculation BUFFERS=916, rounded up to 924 buffers to allow offload to complete in groups of 12 (192 channel system).

Control From Epics, in shot

Live Status and Live data plot, Push Buffers increments until completion.



Offload using nc

This isn't a serious example, more to show that, this is ALL that is required..

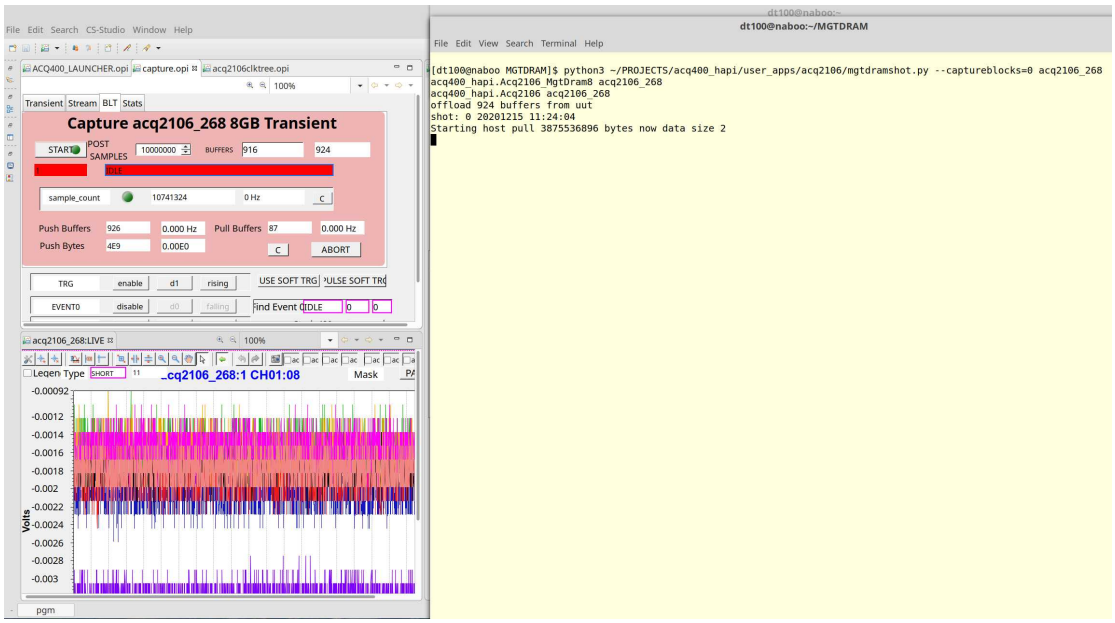
```
[peter@eigg-fs ~]$ nc -i4 acq2106_157 53991 | pv | sha1sum
2.92GiB 0:02:43 [33.3MiB/s] [                                <=>    ]
5.13GiB 0:04:38 [4.59MiB/s] [                          <=>          ]
 5.3GiB 0:04:48 [30.1MiB/s] [              <=>                      ]
7.81GiB 0:06:57 [19.2MiB/s] [                  <=>                  ]
27ef4276744480847f2de2c8eacced3745b84781  -
```

Offload Using mgtdramshot.py

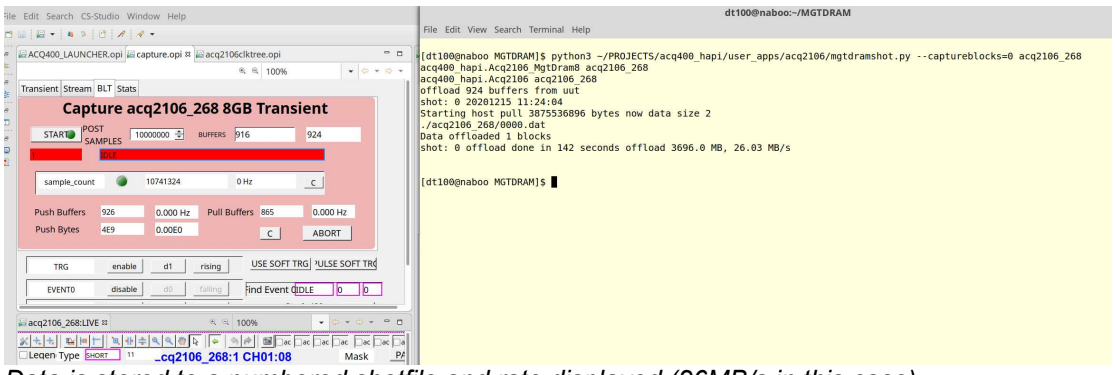
Run the script after the capture has finished..

```
python3 ~/PROJECTS/acq400_hapi/user_apps/acq2106/mgtdramshot.py --
captureblocks=0 acq2106_268
```

Note the Pull Buffers count incrementing during offload.



On Completion of offload



Data is stored to a numbered shotfile and rate displayed (26MB/s in this case).

27.4 Control From Script:

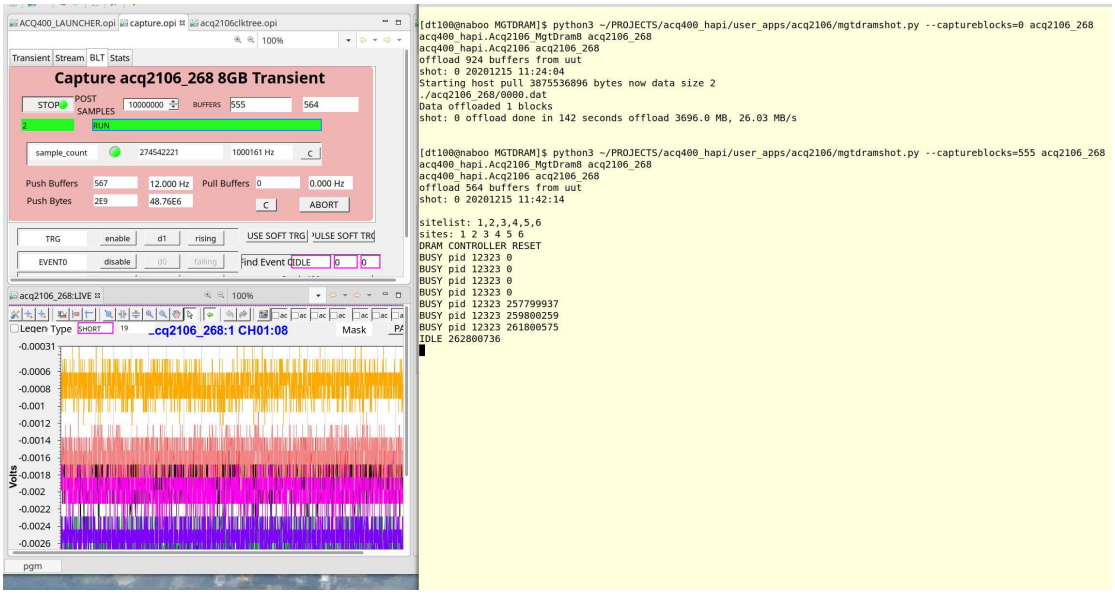
As well as offloading the data, mgtdram shot can control the shot.

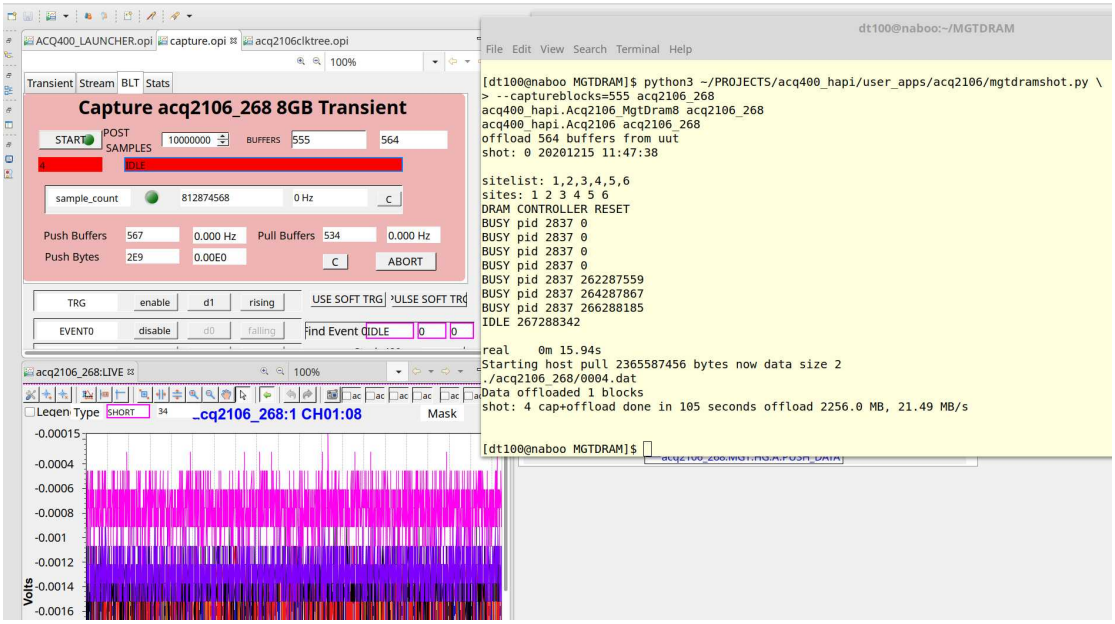
The EPICS UI can act as an observer in this process, providing live status update.

eg : capture and offload 555 blocks.

```
./mgtdramshot.py --captureblocks=555 acq2106_268
```

Control From Script, with EPICS UI showing live status.





On Completion :

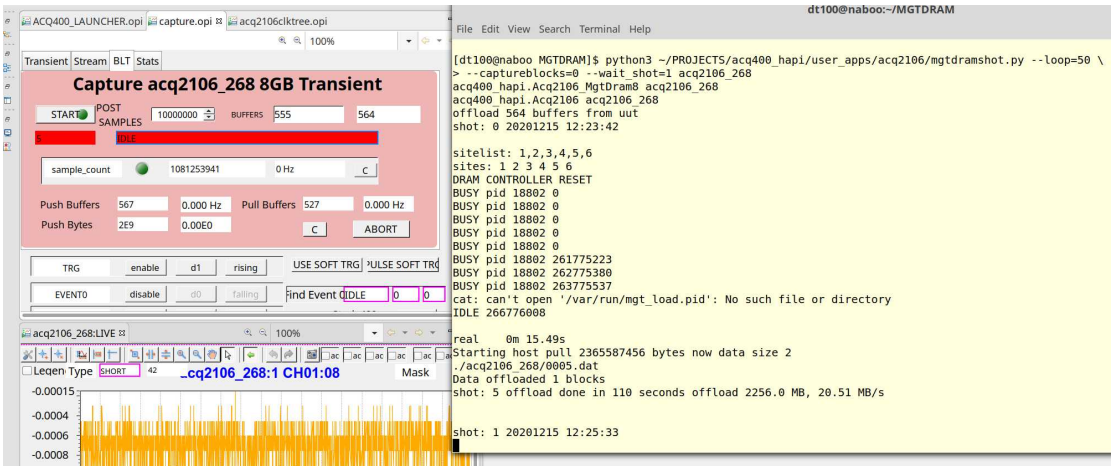
27.5 Control From EPICS, auto offload from script.

In this scenario, the control script mgtdramshot is set up to loop waiting for every completion.

The shot cycle is managed from EPICS as per 27.3

mgtdramshot starts an offload to a numbered shot file on completion of every shot. The shot number comes from the box, this is auto-incremented every shot, and may be set by the user eg at start of day.

```
python3 ~/PROJECTS/acq400_hapi/user_apps/acq2106/mgtdramshot.py \
        --loop=50 --captureblocks=0 --wait_shot=1 acq2106_268
```



27.6 [Deprecated] Configure Data offload method: FTP

FTP is a possible way to offload data. The ACQ2106 is the FTP client, and an automated routine sends data one block at a time to a remote FTP server.

27.6.1 Automating the ftp upload.

First, activate the FTP client package.

```
mv /mnt/packages.opt/90-custom_ftp*  /mnt/packages
```

Then, create an automated upload environment as follows. Set FTPHOST and password. Save and reboot

```
cp /usr/local/CARE/mnt-local-sysconfig-custom_ftp.init \
     /mnt/local/sysconfig/custom_ftp.init
vi /mnt/local/sysconfig/custom_ftp.init
sync;sync;reboot
```

```
#!/bin/sh
# custom_ftp.init
# to customise, copy the else stanza to
/mnt/local/sysconfig/custom_ftp.init
# to create your own customized .netrc
HN=$(hostname)
FTPHOST=brotto

(
echo export HOME=/root
echo export FTPHOST=$FTPHOST
echo export MGTOFFLOADCUSTOM=/usr/local/CARE/mgt_offload_custom.ftp
) > /etc/mgtsh.env

cat - >/root/.netrc <<EOF
machine $FTPHOST
login dt100
password mypassword

macdef init
cd /data/$HN

EOF
chmod 600 /root/.netrc
```

- check that the target dir /data/$HN exists on FTPHOST
- test auto login after reboot , should log in and change dir automatically:

```
source /etc/mgtsh.env
ftp $FTPHOST
```

27.7 Automation.

MGT-DRAM enables a site service at “Site 14”. This allows the user to specify actions. Separately an “action service” is configured at port 53990.

The control technique is:

- first specify the action at site 14

- then connect to port 53990 to run the action, reading a live transcript of action progress.

27.7.1 Configure Actions

```
acq2106_078> get.site 14
help
mgt_offload
mgt_reset_counters
mgt_run_shot
mgt_taskset
```

- mgt_run_shot [NBLOCKS] : run a shot for NBLOCKS.
  - eg mgt_run_shot 2000 # prepare run for 2000 blocks (8GB)
  - Clock, trigger and aggregator must be pre-configured.
- mgt_offload [BLOCK][-BLOCK1] : prepare to offload BLOCK or a range of BLOCKS.
  - eg mgt_offload 0-1999 # prepare offload first 2000 blocks (8GB).

27.7.2 run_shot action

```
set.site 14 mgt_run_shot 2000
```

```
nc UUT 53990
/usr/local/bin/procServ: spawning daemon process: 28904
Warning: No log file and no port for log connections specified.
..
BUSY pid 28843 SIG:SAMPLE_COUNT:COUNT 0
BUSY pid 28843 SIG:SAMPLE_COUNT:COUNT 188017
...
BUSY pid 28843 SIG:SAMPLE_COUNT:COUNT 30190636
IDLE SIG:SAMPLE_COUNT:COUNT 32189230
real  0m 21.33s
END
```

27.7.3 offload action

```
set.site 14 mgt_offload 0-2000
```

```
nc UUT 53990
axi0 start OK 0000 OK
axi0 start OK 0001 OK
axi0 start OK 0002 OK
axi0 start OK 0003 OK
```

For increased speed, the upload may aggregate multiple blocks into one file.

27.7.4 Full Host-side automation

A complete data test suite is available from ACQ400HAPI. It sets the modules to simulate mode (to generate ramp data), and requires that an appropriate validator is available on the host computer

Example:

```
cd acq400_hapi/user_apps/acq2106/
./mgtdramshot.py --captureblocks 2000 –sim=1,2,3,4 \
--validate ~/bin/validate-multisite-ramp-4x425 \
--loop 100 acq2106_054

--captureblocks N : number of 4 MB blocks to capture (2000=8GB)
--validate V      : V is a validation program  (eg 4xACQ425)
--loop L          : number of times to repeat. (100 standard)
```

```
[peter@eigg-fs ~]$ nc -i4 acq2106_157 53991 | pv | sha1sum
2.92GiB 0:02:43 [33.3MiB/s] [                              <=>   ]
5.13GiB 0:04:38 [4.59MiB/s] [                      <=>           ]
 5.3GiB 0:04:48 [30.1MiB/s] [            <=>                     ]
7.81GiB 0:06:57 [19.2MiB/s] [               <=>                  ]
27ef4276744480847f2de2c8eacced3745b84781  -

mgtdramshot --host_pull=1
```

27.8 Higher bandwidth to memory Stack and Stack/Stagger

MGT-DRAM images for ACQ48x modules are now available with a “Stack” FPGA image that will “Stack” the output of a pair of ACQ48x onto a double width bus. This increases the available memory bandwidth to in excess of 1280MB/s. In standard “Stack” configuration, ACQ48x modules are populated in pairs in sites 1,2, extending to 3,4 and to 5,6. In the special “Stack/Stagger” configuration, a pair of ACQ480 modules are mounted in sites 1 and 3; this allows for better front panel layout.

Software handles the various configurations via a single command: stack_480

1. stack_480 2x4 : 2 modules x 4 channels, capture total 8ch x 80MSPS
2. stack_480 2x8 : 2 modules x 8 channels, capture total 16ch x 40MSPS
3. stack_480 4x8 : 4 modules x 8 channels, capture total 32ch x 20MSPS
4. stack_480 6x8 : 4 x 8 channels, capture 48ch x 12.5MSPS.

Note that there are two consequences of Stack mode:

1. The ZYNQ live data path only sees one half of the stack bus, so live plots are available for half the number of channels.
2. The raw data format changes. This is handled by the HAPI program

host_demux.py --stack_480=OPTION where option matches the run time option chosen previously.

Transcript

```
[dt100@brotto acq2106]$ ./mgtdramshot.py --captureblocks=2000 --
loop=100 --sim=1,2,3,4 --validate=~/bin/validate-4x480 acq2106_113
shot: 99 20181125 07:29:00

sitelist: 1,2,3,4
sites: 1 2 3 4
DRAM CONTROLLER RESET
/usr/local/bin/procServ: spawning daemon process: 27899
Warning: No log file and no port for log connections specified.
BUSY pid 27892 SIG:SAMPLE_COUNT:COUNT 0
..
BUSY pid 27892 SIG:SAMPLE_COUNT:COUNT 674635
BUSY pid 27892 SIG:SAMPLE_COUNT:COUNT 10678333
BUSY pid 27892 SIG:SAMPLE_COUNT:COUNT 20682028
BUSY pid 27892 SIG:SAMPLE_COUNT:COUNT 40689392
BUSY pid 27892 SIG:SAMPLE_COUNT:COUNT 50693083
BUSY pid 27892 SIG:SAMPLE_COUNT:COUNT 60696768
BUSY pid 27892 SIG:SAMPLE_COUNT:COUNT 70700464
BUSY pid 27892 SIG:SAMPLE_COUNT:COUNT 90707837
BUSY pid 27892 SIG:SAMPLE_COUNT:COUNT 100711521
BUSY pid 27892 SIG:SAMPLE_COUNT:COUNT 110715210
IDLE SIG:SAMPLE_COUNT:COUNT 130722599

real    0m 20.66s
user    0m 0.01s
sys 0m 0.06s
END
>customisation: source /usr/local/CARE/mgt_offload_custom.ftp
>NGROUP 16 blocks per upload
0.........1.........2.........3.........4.........5.........6.......
..7.........8.........9.........10
0.........1.........2...
>buffers: 2001 interrupts: 125
>real    4m 52.67s
>user    0m 1.54s
>sys 0m 1.63s
>END
upload 8000 MB done in 0:04:53.293895 seconds, 27 MB/s

run "~/bin/validate-4x480 acq2106_113"
wrapcount:16384
0001f4000000 bytes 8000 Mbytes 0 errors
done in 319 seconds
```

28 Boot time Customization.

28.1 Include Custom packages

Move optional packages from /mnt/packages.opt to /mnt/packages.

28.2 Configuration files

28.2.1 /mnt/local/sysconfig/bos.sh

Provides limits to transient captures

```
PREMAX=0
POSTMAX=4000000
```

28.2.2 /mnt/local/sysconfig/epics.sh

Customisation for local EPICS IOC: standard EPICS environment variables and custom D-TACQ changes

```
# standard EPICS vars eg large arrays
export EPICS_CA_MAX_ARRAY_BYTES=500000

# set an alias prefix to alias every record
#export IOC_GLOBAL_ALIAS_PFX=bl22b-di-adc-01

# set a script to source BEFORE iocInit()
#export IOC_PREINIT=/mnt/local/epics-custom

# set a script to source AFTER iocInit()
#export IOC_POSTINIT=/mnt/local/epics-custom
```

28.2.3 /mnt/local/sysconfig/site-1-peers

Sets “peer knobs” on slave sites to be controlled by the site 1 knob.

eg

```
PEERS=1,2
KNOBS=gain,clkdiv,clk,trg,sync,rgm,event0,gx
```

28.2.4 /mnt/local/sysconfig/transient.init

Customisation for transient capture. eg

```
COOKED=1 NSAMPLES=32768 NCHAN=64 TYPE=LONG
echo /mnt/local/sysconfig/transient.init set up soft_trigger, two
sites
set.site 1 trg=1,1,1
run0 1
```

28.2.5 /mnt/local/acq400.sh

Optional file

```
REBOOT_KNOB=y
ETH1_E1000X=y
```

1. Creates a site 0 knob, and EPICS PV to allow external reboot.
2. ACQ2106: Enables fiber ethernet on MGT Port D, where available.
   nb: fiber ethernet is enabled AFTER the FPGA has been loaded.

28.2.6 /mnt/local/network

File for non-default (eth0, DHCP) network settings.

ie a custom static ip address, and/or configured E1000X (eth1), see also 28.2.5 .. example setting eth1 to use dhcp:

```
acq2106_172> cat /mnt/local/network
/etc/network/default-networkrc eth1
```

28.2.7 /mnt/local/acq420_custom

Optional File

```
DRVR_CUSTOM=data_32b=1
BLEN=4194304 NBUF=128
```

DRVR_CUSTOM=data_32b=1 : force device driver to load with 32 bit data.

Custom buffer settings, recommended for high throughput,

eg ACQ425-16-2000, ACQ424-32-1000

BLEN: Buffer length, default = 1M, recommend 4M (must be power of 2)

NBUF: number of buffers. Total 512MB is safe, higher counts may be possible, only useful for transient capture.

28.2.8 /mnt/local/sysconfig/acq400_streamd.conf

Streaming options:

```
--subset=[start-channel,]length :
      reduce output channel count to length,
      starting from start-channel [1]
--sum=[start-channel,]length :
      sum over length channels, starting from start-channel [1]
      sum output as stream of int32 from port 4270

STREAM_OPTS="--subset=8 -sum=4"
```

Example Use Case:

ACQ425ELF-16-1000-18.

In 16 bit mode, data rate is 32MB/s. ACQ1001 can support this data rate.

But in 32 bit mode, the data rate is too high for continuous transfer to Ethernet. But it is possible to select a subset of channels, to keep the rate within bounds, eg:

```
# plot first 8 channels
STREAM_OPTS="--subset=8"
# plot second 8 channels
STREAM_OPTS="--subset=9,8"
```

28.3 Final boot customisation

store final adjustments in /mnt/local/rc.user

```
# for soft trigger on streaming connect - most systems do this
cp /usr/local/CARE/acq400_streamd.0.conf-soft_trigger \
/etc/sysconfig/acq400_streamd.0.conf
```

29 Reliability Features

29.1 Watchdog Timer

The ZYNQ SOC includes a hardware watchdog timer wdt. This is a failsafe feature, once enabled by software, software has to service the wdt at a regular interval. If it's not serviced, the wdt will reboot the unit.

29.1.1 Local Service

The wdt may be serviced locally by running the watchdog daemon:

```
watchdog /dev/watchdog0
```

This may be of limited use, since the software process will only stop if the entire OS is locked up. It's possible that the system may lose functionality, but not reset..

29.1.2 Remote Service

With Remote Service, the watchdog is serviced by some outside agent on the network. Eg the HOST PC. Now the HOST can decide “is the unit functioning normally” - if so, it will keep servicing the wdt. But, if the HOST decided the unit is not functioning normally, it can stop servicing the wdt, forcing the UUT to reboot, no matter what state it's in.

To enabled this feature, first enable the custom_wdt optional package.

The remote service is then activated and serviced via http:

```
connect browser to:
http://acq2106_020/cgi-bin/watchdog.cgi?WDT_RESTART
```

Now keep pressing “refresh”. If we don't refresh, the UUT will reboot.

This can be automated easily eg:

```
eigg> while [ 1 ]; do
wget http://acq2106_020/cgi-bin/watchdog.cgi?WDT_RESTART
sleep 10
done

It can actually be self hosted (for test purposes, this is really
equivalent to the “watchdog” daemon:

acq2106_020> while [ 1 ]; do
wget http://acq2106_020/cgi-bin/watchdog.cgi?WDT_RESTART
sleep 10
done
```

30 Appendix: Install a new firmware release

30.1 Firmware release format:

The firmware release contains both:

- ESW: Embedded Software: to run on the embedded ARM controller.
  - The software is universal to all carriers and modules, with boot time customization to suit the configuration.
- GW: Gateware – FPGA images or personalities.
  - Multiple FPGA images are supplied, the images are SPECIFIC to a particular hardware configuration, and the appropriate personality is select by the ESW on boot, after enumerating the hardware.
  - The FPGA personality is selected from stock in the release, but a particular personality may be promoted to load preferentially. No image will be loaded unless it passes basic compatibility requirements.

```
•  acq4xx-639-20180914090352.tgz
•  acq4xx : ACQ400 series, common firmware release.
•  639    : release number
•  20180914090352 : release timestamp
•  .tgz   : contrary to the name, this is a regular tar archive,
   NOT zipped globally, but it does contain a series of zipped
   tar files.
```

30.2 Releases located on web site.

The latest release is always to be found at

Contact/Resources ,

Embedded Firmware Image,

Current 4G release.

Download using you web browser. If the UUT is connected directly to the net you can download it directly as follows:

```
[pgm@hoy5 ~]$ ssh root@acq1001_329
acq1001_329> cd /tmp
acq1001_329> wget http://www.d-tacq.com/swrel/acq4xx-639-
20180914090352.tgz
```

Otherwise, scp the release file to the UUT, then log in

```
scp acq4xx-639* root@UUT:/tmp
ssh root@UUT
```

30.3 Updating the release

Enter the commands in bold:

```
acq1001_329>/mnt/bin/update_release /tmp/acq4xx-641-
20180916212543.tgz

processing release acq4xx-641-20180916212543
./
./bin/
./bin/backup_sd
./bin/check_version
...
./local/
RELEASE : /tmp/release.md5
CURRENT : /tmp/current.md5
--- /tmp/release.md5
+++ /tmp/current.md5
...
post-copy version check:
RELEASE : /tmp/release.md5
CURRENT : /tmp/current.md5
RELEASE acq4xx-198-20140207104747
Clean Release Installed
ALL GOOD

sync;sync;reboot
```

30.3.1 Custom Packages

The update process will warn if any custom packages are being overwritten by the upgrade. The user should note these and restore them after the upgrade is complete

```
RELEASE : /tmp/release.md5
CURRENT : /tmp/current.md5
--- /tmp/release.md5
+++ /tmp/current.md5
+a94581e4c5e319f8e968fa5cd7f5d6c8  ./packages/35-custom_gpg-1711302206.tgz
+1b8ab75e97747749db20f58adfd9c0ee  ./packages/38-custom_8pps-1706141620.tgz
+1126dd9fcdde819586aec2b8e16ae521  ./packages/99-custom_awg-1707101755.tgz
+cb3b68227f1b8d3bfe6f5d0704a6c564  ./packages/99-custom_hil-1412222203.tgz
-1126dd9fcdde819586aec2b8e16ae521  ./packages.opt/99-custom_awg-1707101755.tgz
-cb3b68227f1b8d3bfe6f5d0704a6c564  ./packages.opt/99-custom_hil-1412222203.tgz
Warning, patching detected
```

To restore the functionality, after the update, enable the new versions of the custom packages eg:

```
mv /mnt/packages.opt/35-custom_gpg*  /mnt/packages
...
```

Possibly run installation-specific customiser eg
/mnt/local/custom_mag_config_new_release

Depending on specific requirement.

30.3.2 Patch FPGA Images

A system may be shipped with a patch FPGA.

```
/mnt/acq2106_08_08*tgz
```

The “patch FPGA” is promoted in preference to the standard stock FPGAs in `/mnt/fpga.d`

Sometimes, the “patch FPGA” is there because it’s adding extra functionality that you want eg FIR filtering. In that case, update the patch from stock with the latest equivalent image after update.

Sometimes, the “patch FPGA” was simply there as a “between-release” patch. In that case, delete it and allow the stock image from the release to load.

30.3.3 Boot Customisation

The firmware update process does not touch customisation files in

`/mnt/local/`

31 Appendix: Brief Guide to EPICS and CSS

31.1 What is it and why should I care

The D-TACQ ACQ400 series intelligent controllers all include an EPICS IOC to manage run time logic and present data values. CSS screens are provided to monitor and manage the system remotely.

EPICS is an industrial strength control system used in many large scientific installations eg accelerators, light sources and ITER.

The core is an IO Controller (IOC). There may be many IOC's running on many embedded computers. The IOC contains a database of Process Variables PV's. The PV's are visible to networked clients using Channel Access (CA). Each PV represents a control point in the system controlled by the IOC. PV's may be scalars – eg control knobs “Start|Stop” or scalar outputs “TEMPERATURE”, or they can be waveform vectors eg CH01 Volts. The IOC database may include logic to control generation and presentation of the PV's – for example, scaling raw analog to volts. The IOC can be controlled and monitored on the network from a Channel Access client. There are simple clients, and there are graphical clients such as Control System Studio CSS, which enables highly functional cross-platform Operator Interfaces - OPI.

NB: you don't have to use CSS/EPICS. But it does provide a good way to control and view the system. And even non-EPICS clients are able to use selected EPICS PV's transparently through the site service interface.

31.2 Monitoring the embedded IOC

If you're familiar with IOC's, the IOC console is available through the command

```
acq1001_048> acq4xx-epics-console
@@@ Welcome to procServ (procServ Process Server 2.6.0)
...
epics> dbgrep *
dbgrep *
acq1001_048:1:AI:CH:01
acq1001_048:1:AI:CH:02
...
```

It's worth trying this command to at least see what is available

eg “epics> dbgrep *” will list all the PV's.

31.3 Client Side Tools

31.3.1 EPICS Base

This isn't strictly necessary at all. However, if you're familiar with EPICS and

have it pre-installed, you have instant access to all PV's using the text-based CA commands. eg it's possible to log waveforms using the camonitor command.

31.3.2 Control System Studio

This is strongly recommended, not least because all the D-TACQ supplied OPI's are specific to this tool

31.3.3 Other OPI clients

Several other OPI clients are available for EPICS CA, including

motif: medm, edm and qt: epicsQt.

These tools are NOT considered here.

31.4 Notes on Installing Control System Studio CSS

31.4.1 Freely available download:

General informationORNL CSS Site

Product Download Product Download

Choose latest version of Basic EPICS , select your platform, choice of

- Windows, 32 bit or 64 bit
- Linux, 32 bit or 64 bit
- Mac OSX, 64 bit.

Download the file, unzip and run the “css” program.

31.4.2 CSS is an extension to Eclipse

Eclipse is the well-known IDE for software development. CSS use the Eclipse “Rich Client” interface to control its display. So CSS is both a design environment and a run-time enviroment.
We already did the design, but so far we haven't succeeded in providing just a run-time executable, so it's a little more complex than it really needs to be, but still isn't difficult to set up.

Eclipse has a concept of a PROJECT and a WORKSPACE

- PROJECT : is the set of OPI's downloaded from D-TACQ
- WORKSPACE : holds your local settings.

If you have many UUT's, you might have the one PROJECT and many WORKSPACES. Each WORKSPACE would reference the same PROJECT,

but may save local settings, eg HOST ID (UUT) and window configuration

Eclipse includes a window manager with MDI control, and NSEW docking, you can use this to set up your screen the way it's best for you.

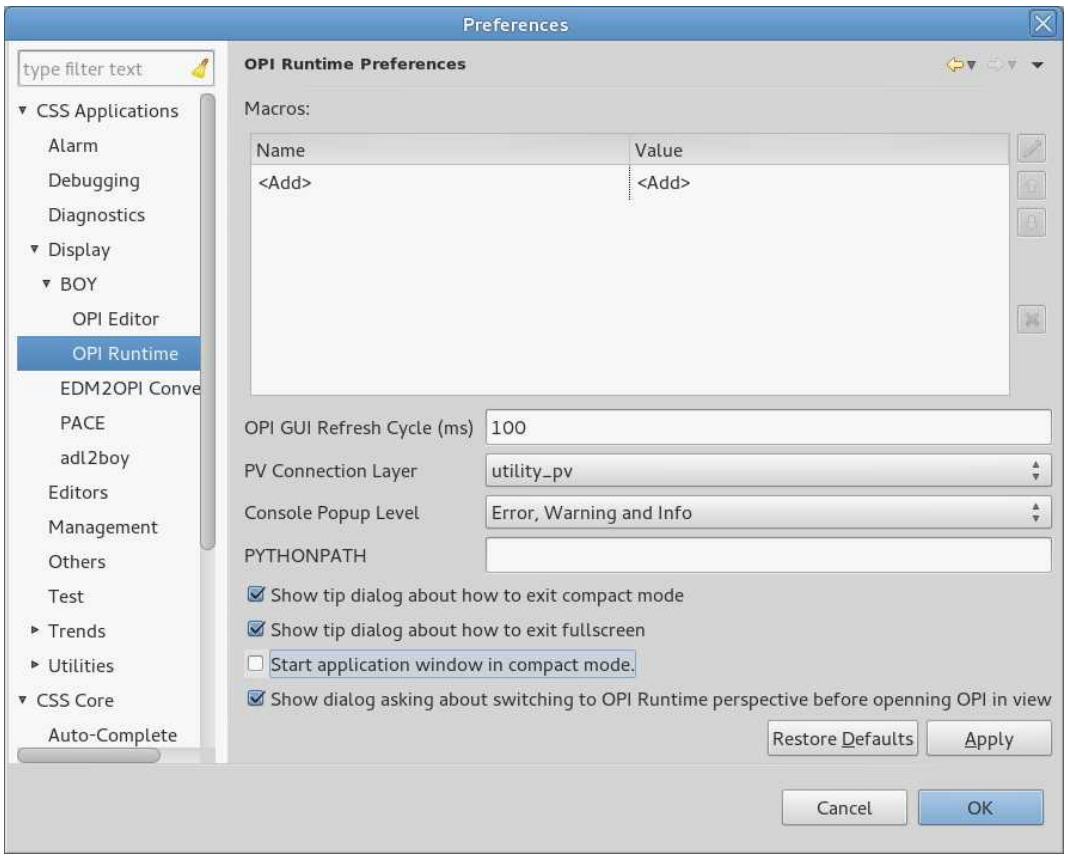
31.4.3 Create Project and Workspace

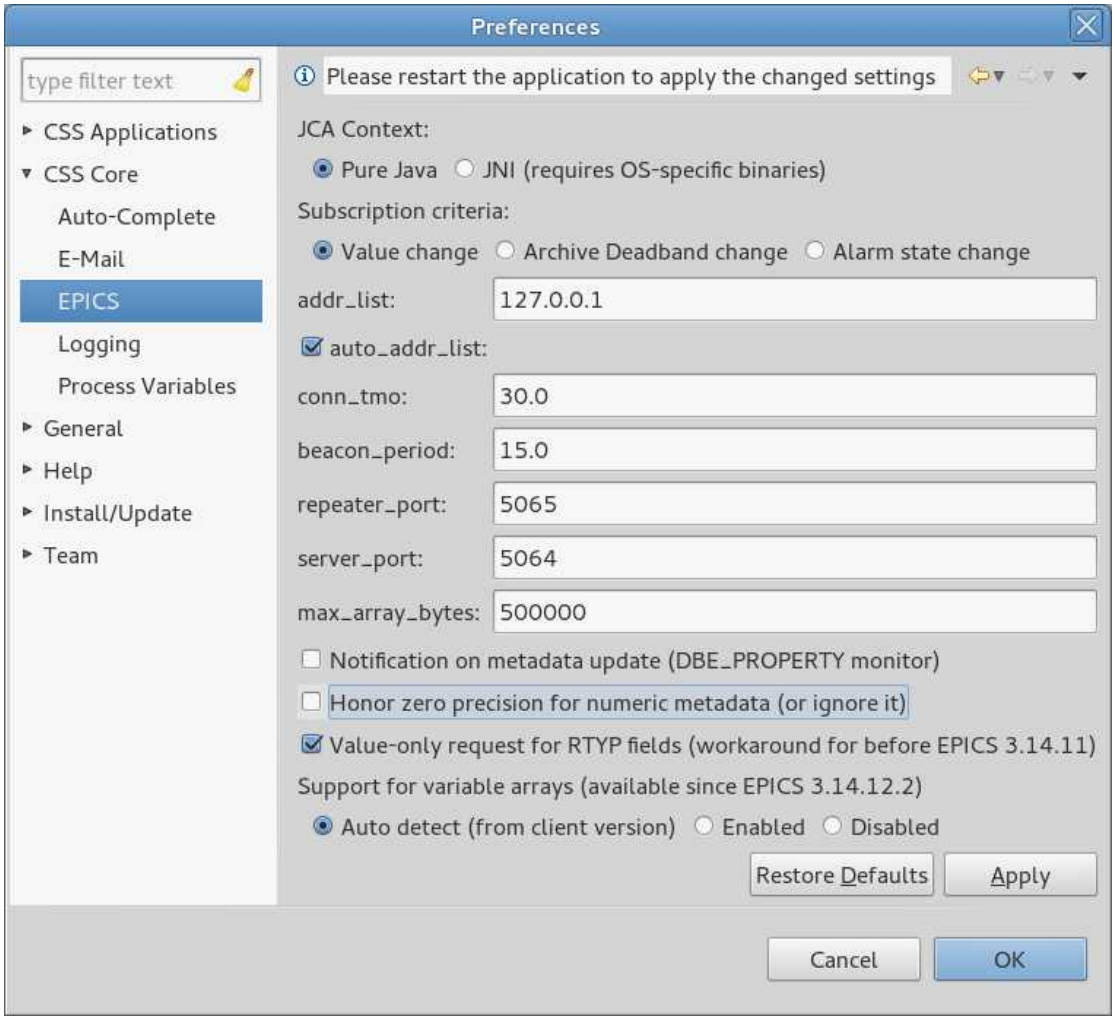
- unzip PROJECT
- Launche CSS and accept default workspace
- Import PROJECT

31.4.4 Set CSS Preferences.

Menu | Edit | Preferences

- Display/BOY/OPI runtime
- Macros: add name: UUT, value: the name of the UUT eg acq2106_077
- For ACQ1014, omit UUT, add name UUTLEFT, UUTRIGHT



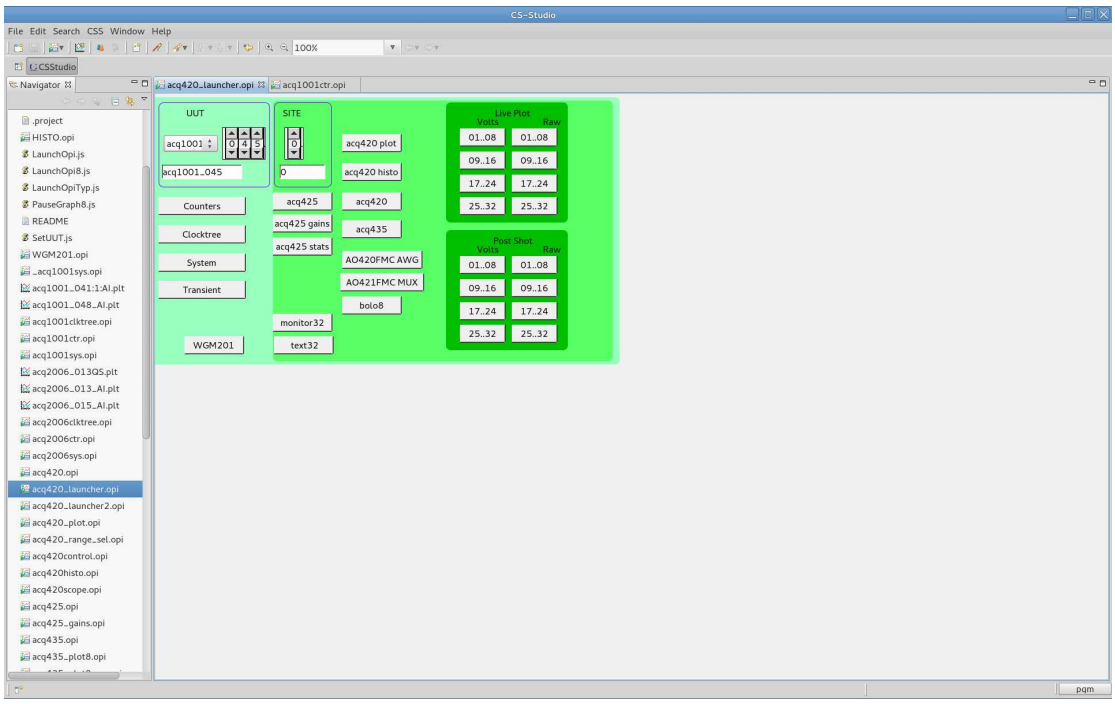


- Set max_array_bytes = 500000
- auto_addr_list should be set, but if PV discovery doesn't work, entering actual UUT IP address (or name if DNS is working) here can help. along with initially disabling local firewall
- Restart, shortcut: Menu|File|Restart CSS

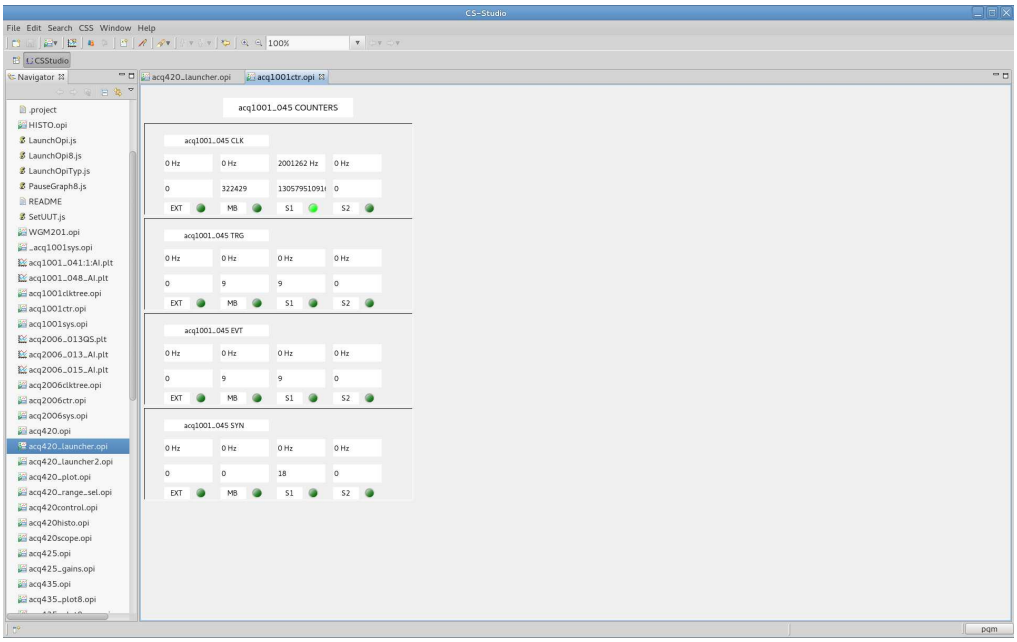
31.4.5 Run the project

- Open in design mode, select the launcher – always run the launcher
- Close the navigate, console, properties windows
- Select “Compact Mode” F8
- Select a UUT, (name and number).

- Select the Launcher OPI – *ALWAYS USE THE LAUNCHER*!
- Select UUT type, name, optional site ..



- Now

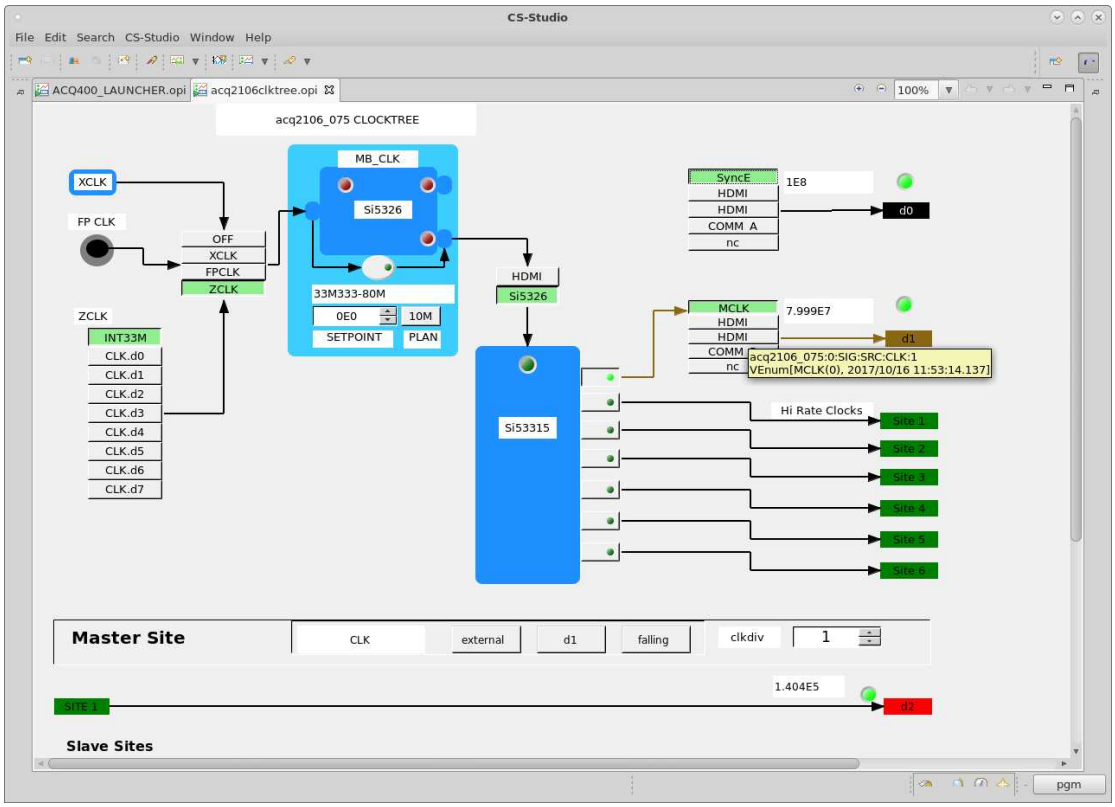


press a button to launch an OPI (eg counters, check it connects), then pick screens and arrange as required.

32 Appendix : ACQ2106 Clocktree

32.1 Default boot with ACQ424ELF

Default

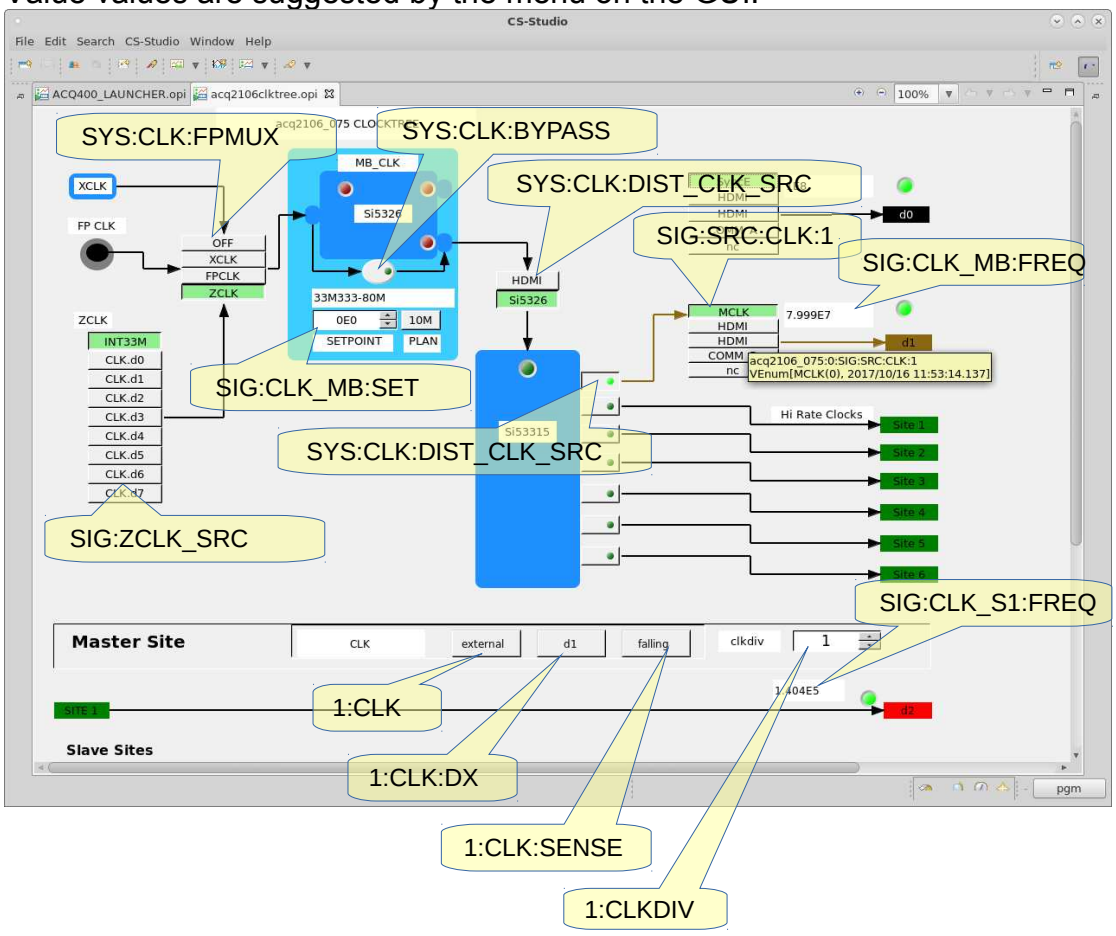


32.2 Control Names

Each control has a name (discover by hovering in the GUI).

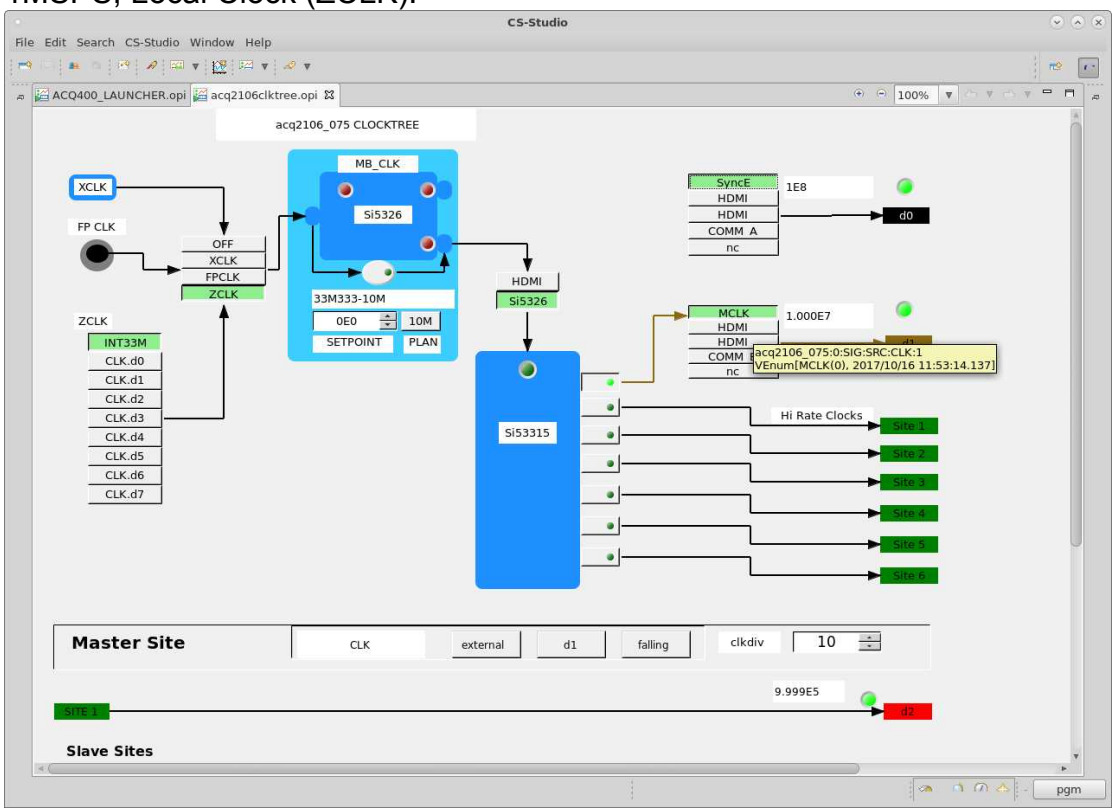
This name can be used for scripted control.

Value values are suggested by the menu on the GUI.



32.3 Example: Factory set boot

1MSPS, Local Clock (ZCLK).



From Boot:
/mnt/local/rc.user:
/usr/local/CARE/acq2106+acq424.init

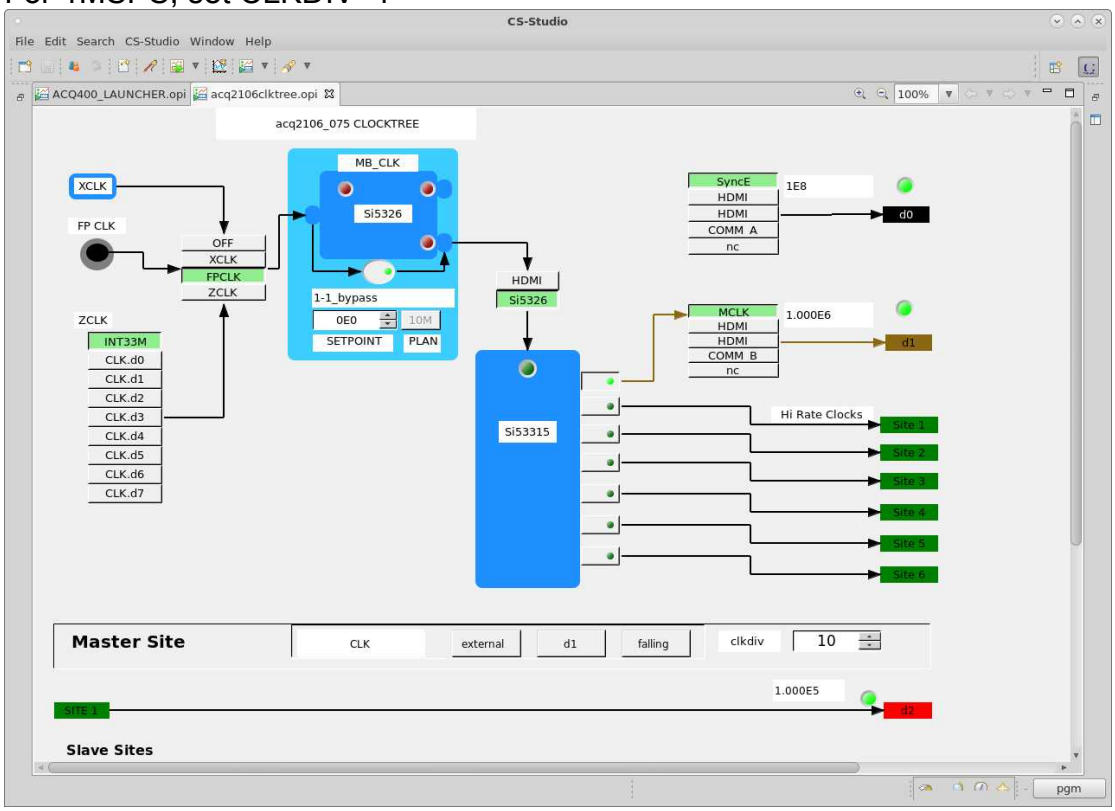
```
set.site 0 SIG:ZCLK_SRC INT33M
set.site 0 SYS:CLK:FPMUX ZCLK
set.site 0 SYS:CLK:DIST_CLK_SRC Si5326
set.site 0 SYS:CLK:OE_CLK1_ZYNQ 1
load.si5326 /etc/si5326.d/si5326_33M333-10M.txt
set.site 1 CLKDIV=10
```

32.4 Example: Configure a 1MHz external clock

1MHz plant clock on front panel.

100KSPS sample rate set with CLKDIV=10.

For 1MSPS, set CLKDIV=1



From Boot:
/mnt/local/rc.user:
/usr/local/CARE/acq2106+acq424.init

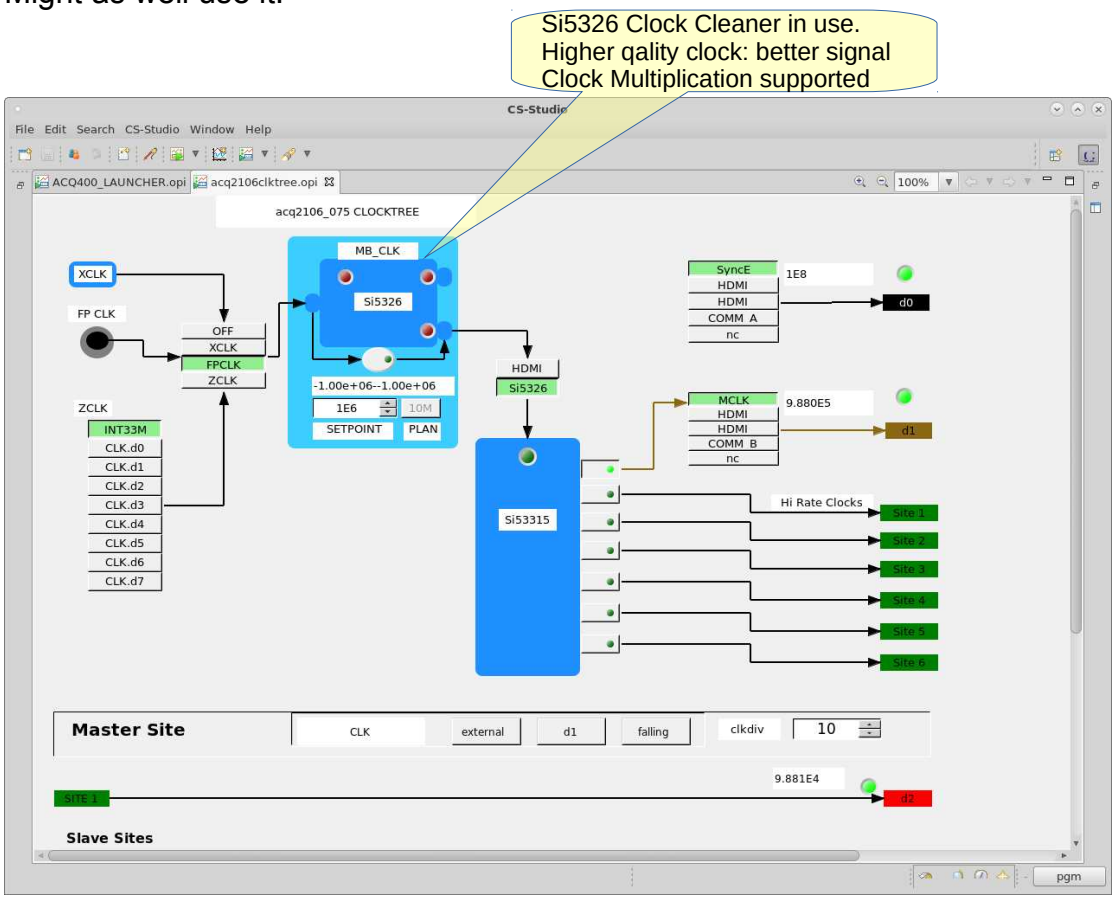
Configured by from a remote HOST, HAPI example

```
[pgm@hoy4 acq400_hapi_tests]$ python
Python 2.7.13 (default, May 10 2017, 20:04:28)
>>> import acq400_hapi
>>> uut=acq400_hapi.Acq400("acq2106_075")
>>> uut.s0.SYS_CLK_FPMUX="FPCLK"
>>> uut.s0.SYS_CLK_BYPASS="1"
>>> uut.s0.SIG_CLK_S1_FREQ
'SIG:CLK_S1:FREQ 100036'
```

32.5 Example 1MHz external clock, with clock cleaner

ACQ2106 includes a really high quality Clock Cleaner / Generator, Si5326.

Might as well use it:



From Boot:
/mnt/local/rc.user:
/usr/local/CARE/acq2106+acq424.init

Configured by from a remote HOST, HAPI example

```
[pgm@hoy4 acq400_hapi_tests]$ python
Python 2.7.13 (default, May 10 2017, 20:04:28)
>>> import acq400_hapi
>>> uut=acq400_hapi.Acq400("acq2106_075")
>>> uut.s0.SYS_CLK_FPMUX="FPCLK"
>>> uut.s0.SIG_CLK_MB_FIN="1000000"
>>> uut.s0.SIG_CLK_MB_SET="1000000"
>>> uut.s0.SIG_CLK_S1_FREQ
'SIG:CLK_S1:FREQ 100129'
```