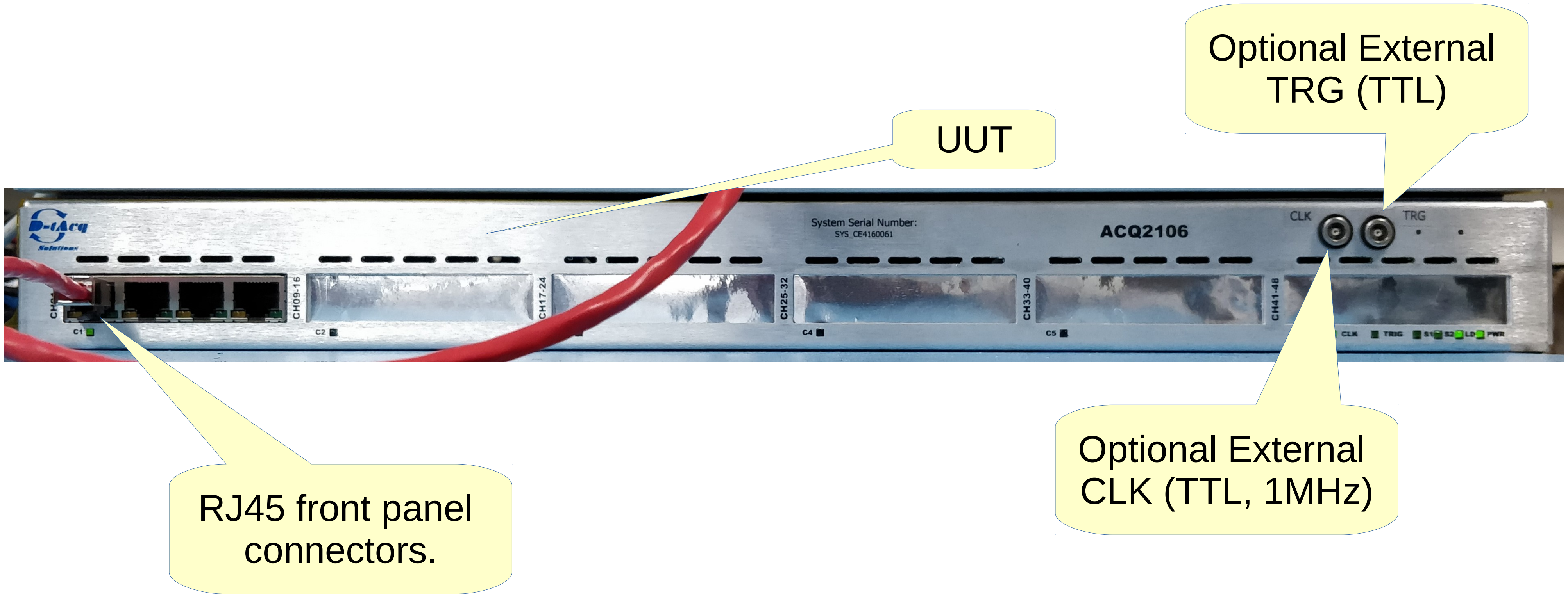
BOLO8BLF Low Latency Control Factory Acceptance Test


Optional External TRG (TTL)
UUT
D-tAcq Solutions
System Serial Number: SYS_CE4160061
ACQ2106
CLK
TRG
CH09-16
CH17-24
CH25-32
CH33-40
CH41-48
C1
C2
C4
C5
CLK
TRIG
S1
S2
LD
PWR
Optional External CLK (TTL, 1MHz)
RJ45 front panel connectors.

Relevant documentation

BOLO8BLF Theory of operation and users guide:
https://github.com/jacklovell/bolodsp-doc/releases

BOLO8BLF Calibration report and guide (including MDSplus usage):
http://www.d-tacq.com/resources/Bolo_calibration_report_user-guide.pdf

BOLO8BLF python control script available from:
https://github.com/D-TACQ/acq400_hapi

BOLO8BLF LLC configuration script available from:
https://github.com/D-TACQ/AFHBA404

BOLO8BLF operation via ethernet connection

Standard BOLO operating procedure via ethernet requires configuration of /mnt/local/sysconfig/bolo.sh. Set the desired channels to calibrate. Example below demonstrates channels one and two. All of the channels can be calibrated at once (or any subset of channels).

```
acq2106_061> cat /mnt/local/sysconfig/bolo.sh

BOLO_ACTIVE_CHAN="1 2"

BOLO_VERBOSE=1

set.site 14 DIODE_DROP_V 0.5

set.site 14 THEAT 1.0

set.site 14 TCOOL 1.0

set.site 14 VBIAS 1.0

COPY_CALIB_DATA=1
```

BOLO8BLF operation via ethernet connection

The BOLO system must be calibrated before use. There is a python wrapper for this which can be found in the D-TACQ github repository:

https://github.com/D-TACQ/acq400_hapi/blob/master/user_apps/special/bolo8_cal_cap_loop.py

This script can be used as such:

```
python bolo8_cal_cap_loop.py --cal=1 --cap=1 --shots=1 acq2106_061
```

The arguments can be changed to perform only a capture and only a calibration by changing cap and cal respectively.

BOLO8BLF & AFHBA404

The BOLO8BLF can be operated in low latency control mode. In this mode an AFHBA404 PCIexpress card, in a host, is used to offload data from the UUT.

In order to use this mode please clone the AFHBA404 github repository:

https://github.com/D-TACQ/AFHBA404

Once this is done (follow instructions on the AFHBA404 github) and once the AFHBA404 is inserted into the host the driver can be loaded. To do this navigate to ~/PROJECTS/AFHBA404/ and then run

```
make
```

in this directory. Once this is complete run

```
sudo ./scripts/install-hotplug
```

and then

```
sudo ./scripts/loadNIRQ
```

These steps are also outlined in the README contained in the github repo.

Low latency data offload using cpucopy

In order to use the low latency control on the UUT the user must first run llc-bolo-harness.py which is contained in the AHFBA404 github repository under the HAPI directory. It is important that the system is not calibrated after this script has been run since a valid calibration will not be obtained with the LLC parameters set. To run the llc-bolo-harness.py command:

```
SPAD_LEN=8 AISITES=1 ./HAPI/llc-bolo-harness.py acq2106_061
```

At this point the UUT is configured for LLC operation and the cpucopy program can now be run (devnum is the index of the port on the AFHBA404):

```
sudo DEVNUM=0 DO32=0 AOCHAN=0 DUP1=0 AICHAN=48 SPADLONGS=8 ./LLCONTROL/afhba-llcontrol-
cpucopy 21000
```

Then run a capture as before using:

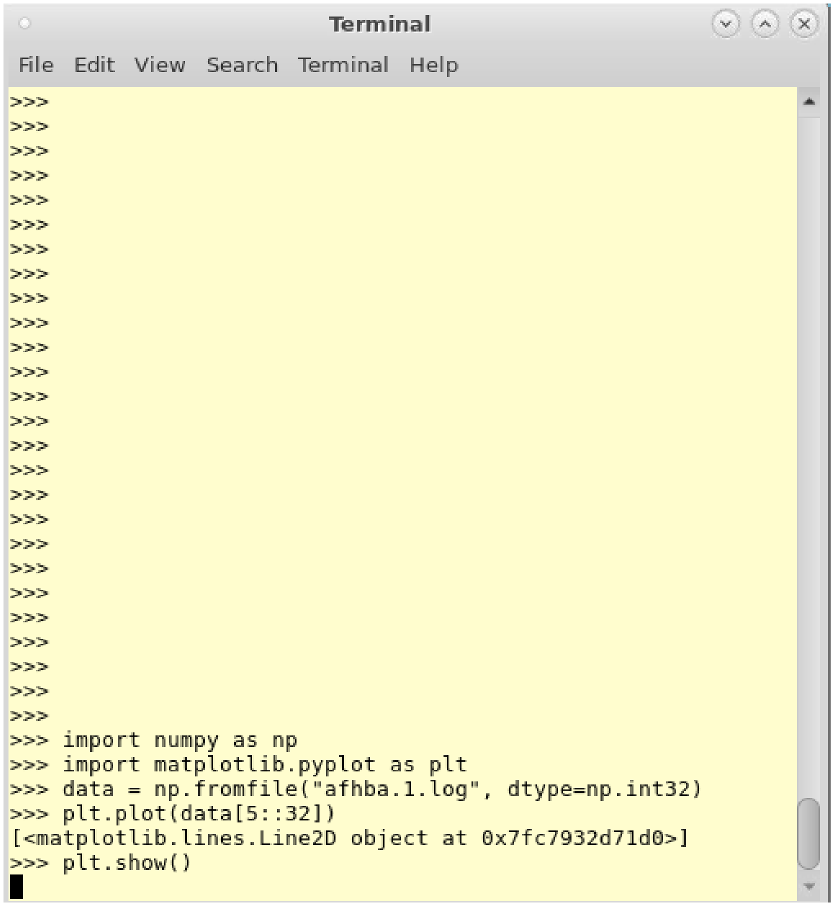
```
python bolo8_cal_cap_loop.py --cal=0 --cap=1 --shots=1 acq2106_061
```

A copy of the data will be stored to afhba.<sfp_port_number>.log

Viewing LLC data on host

To view data on the host machine an interactive python instance was started. The data is loaded using numpy and plotted using matplotlib. The data being plotted below is a from a flashing bicycle lamp.

N.B. the Y axis scaling has not been corrected and so will be inaccurate.



```
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>> import numpy as np
>>> import matplotlib.pyplot as plt
>>> data = np.fromfile("afhba.1.log", dtype=np.int32)
>>> plt.plot(data[5::32])
[<matplotlib.lines.Line2D object at 0x7fc7932d71d0>]
>>> plt.show()
```

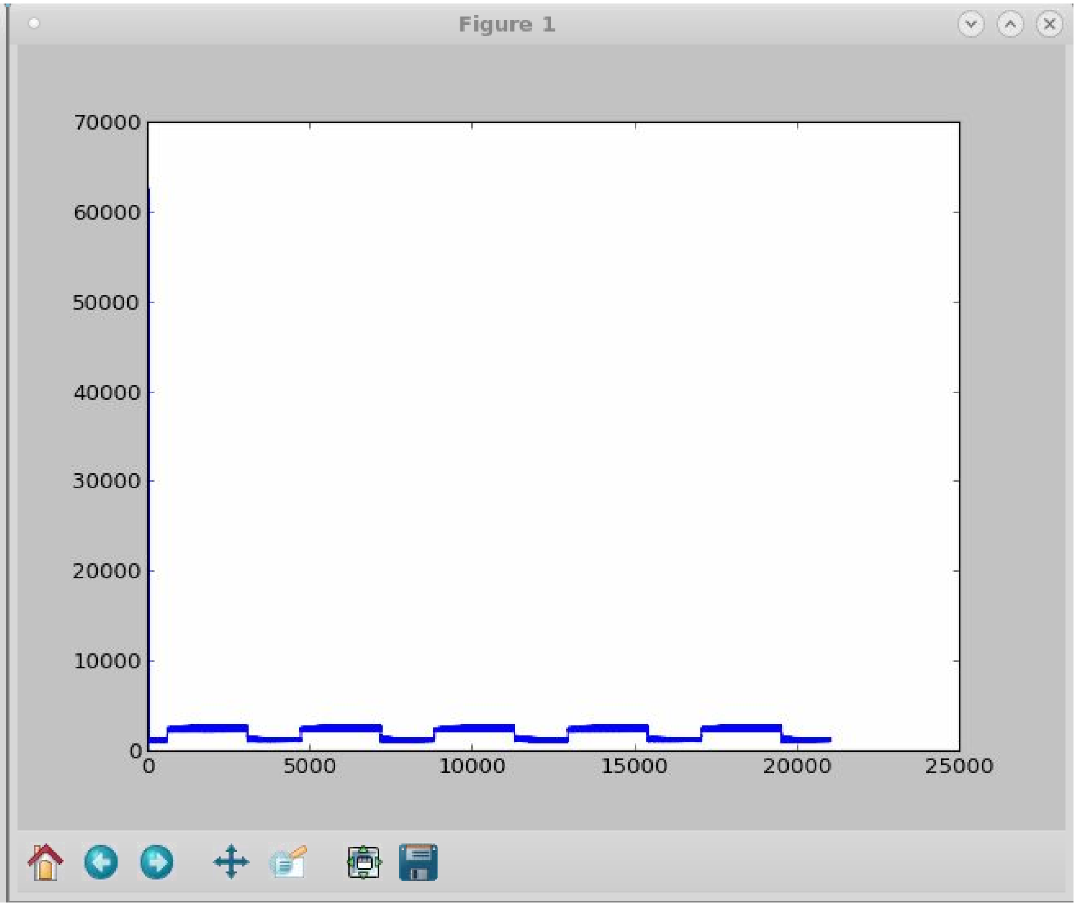


Figure 1