Using ACQ196 FAWG with MDSplus Thin Client.

Prepared By: Peter Milne

Date: October 5 2006

Rev	***Date***	***Description***
1	5 Oct	Initial

Table of Contents

Copyright and Attribution.

Document created using OpenOffice.Org www.openoffice.org.
PDF rendition by extendedPDF, www.jdisoftware.co.uk

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

Document describes how to operate the *ACQ196* 16 channel AO Arbitrary Waveform Generator Function (*AWG*). The general purpose *AWG* on the *ACQ196* is referred to as *FAWG* (FIFO Arbitrary Waveform Generator).

The Examples show control of the card, using three "*TRANSPORTS*":

- Execute on local command shell (best for interactive setup/experiment).
- Using the *HUB* remote interface - for compatibility with existing *MDSplus* installations.
- Using *SOAP*/Web-services, for cross platform applications.

For loading the *AWG*, we show an *MDSplus Thin Client* data upload. The data fetch is controlled from the Host computer, but it could also be controlled autonomously by the *ACQ196*.

For AI data download post shot, we demonstrate:

- Automated data upload post shot, first a quick strided data set before the in memory data *Transform* - data is available in *MDSplus* within 2s of the end of shot, followed by full data set upload after the transform.
- Synchronous data upload - host controls the upload, and blocks while the command completes.
- Asynchronous data upload - host schedules a batch upload job and returns, while the upload continues in the background.

1.1 How to read this doc

Because we're showing several ways to do the same thing, there is a lot of repetition.

In addition, some scripted output is listed literally, then repeated as executed on the machine (so you know what to expect).

It's recommended to read #2 (pictures - easy), check recommended tree structure in #3 - even if this doesn't resemble the site tree structure, it's important to understand the examples. Then, it's a good idea to type through #4 interactively. Finally, for section #5 pick ***one*** of HUB or SOAP (complementary), and also ***one*** of Automated Download or On Demand download. (again, complementary).

Host side scripts are provided by way of example - it's anticipated that users will write their own scripts, and probably not in **`bash`**. As long as they generate equivalent literal output, they should work.

1.2 Alternatives

The FAWG data is held in virtual files, so there are many alternative ways to load it, here are some examples:

- on board **`fungen`** command - generates sine waves
- **`ftp`** or **`curl`** fetch from ftp server
- **`wget`** or **`curl`** http fetch from webserver.

1.3 References

1. Interface Control Document ICD : this is the primary command reference, and is unchanged from the 1G product range, except where overridden by new commands (see Appendix)

2. D-TACQ Network Attached Satellite Data Acquisition (dtNasDaq.pdf)

1.4 Software

- Firmware : may already be installed, if not follow this link:

http://www.d-tacq.com/swrel/acq2xx-linux-2.6.11.11-acq200-111.228.1550-200610042152.tar

- dt100-hub : available for both Linux 2.4 and 2.6:

http://www.d-tacq.com/resources/dt100-hub.tgz

- SOAPI Soap Application Interface:

http://www.d-tacq.com/swrel/SOAPI.200610041513.tgz

- Assorted host-side support scripts:

http://www.d-tacq.com/swrel/host-awgmdsplus.tgz

1.5 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)

1.6 Conventions

- *Target* : the *ACQ196* card, *Host*: a remote PC (*MDSplus* host).
- *AI* data is *downloaded* from the *Target* to the *Host*.
- *AO* data is *uploaded* from the *Host* to the *Target*.
- MDSplus node names - we use the card serial number by convention. A functional name is equally acceptable.

  eg ACQ196_010

- MDSplus tree names - we use names of obscure species of African tree

  eg i*roko, mukwa, mopani, msasa, baobab*.

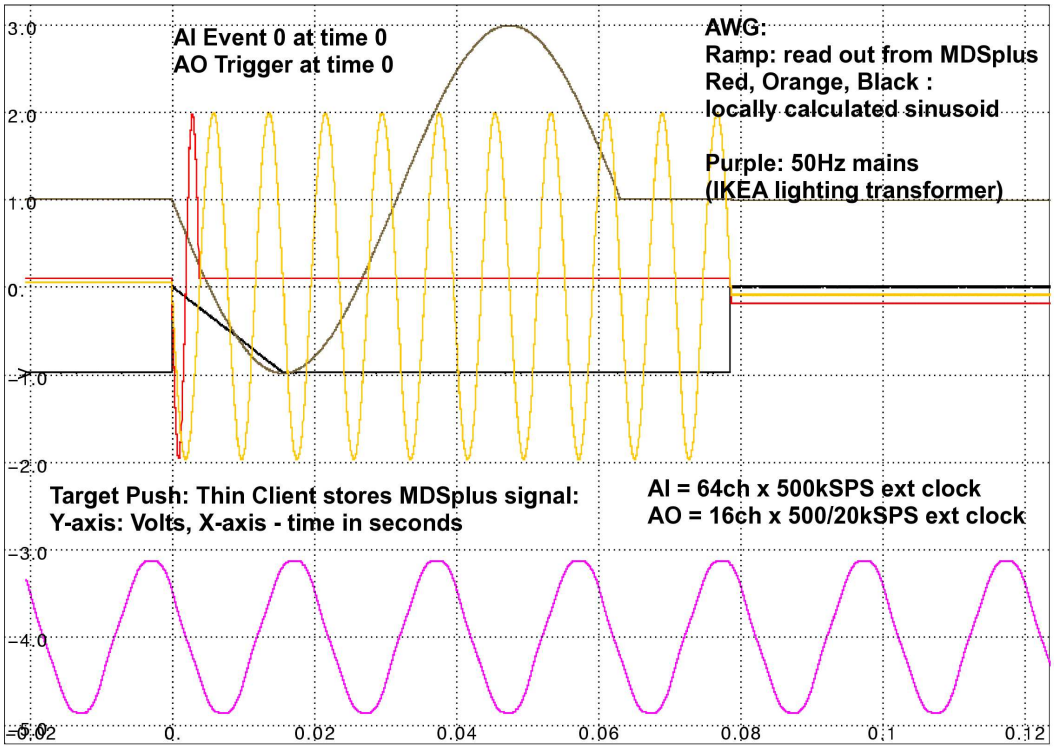
- MDSplus hosts - D-TACQ convention is to use Scottish islands

  eg *kilda, berneray, islay*.

2 Working Example:

2.1 jScope display - AI channels sampling AO FAWG with loopback cable:

10K points pre-event, 100K points post-event. *AWG* loaded with up to 5K samples (max 64K). The free-running AC mains signal shows that sampling is continuous before and after the event.



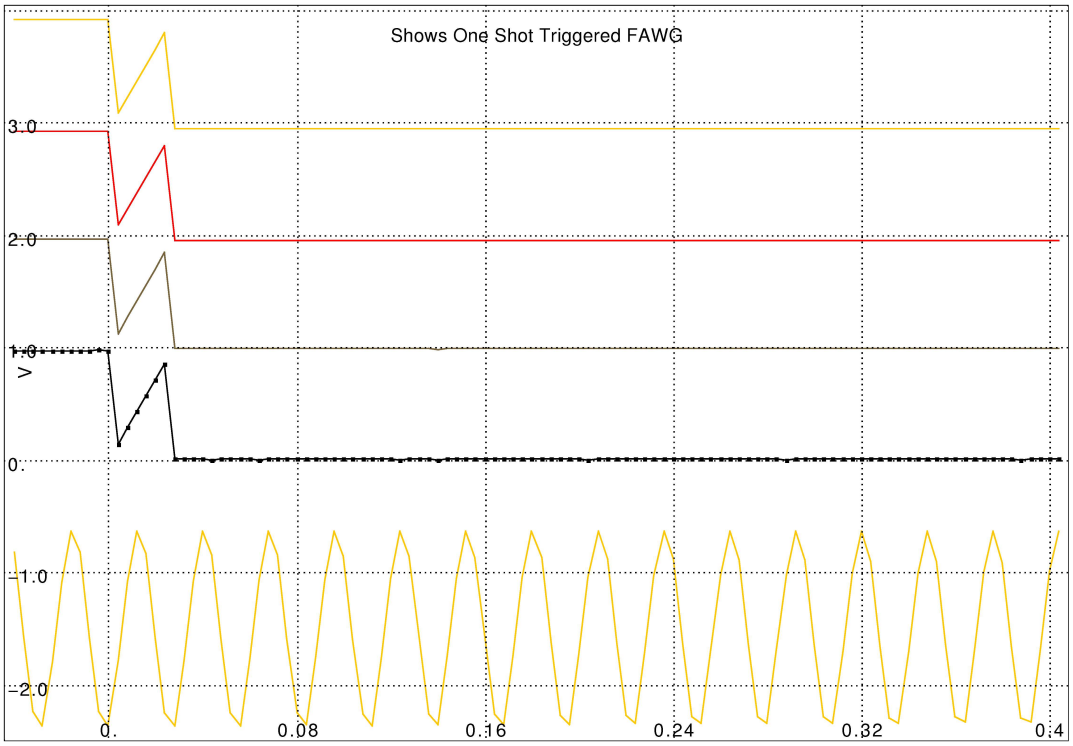
2.2 Auto AI Download after shot.

First, a low sample rate strided data set is uploaded, for display within 2s.

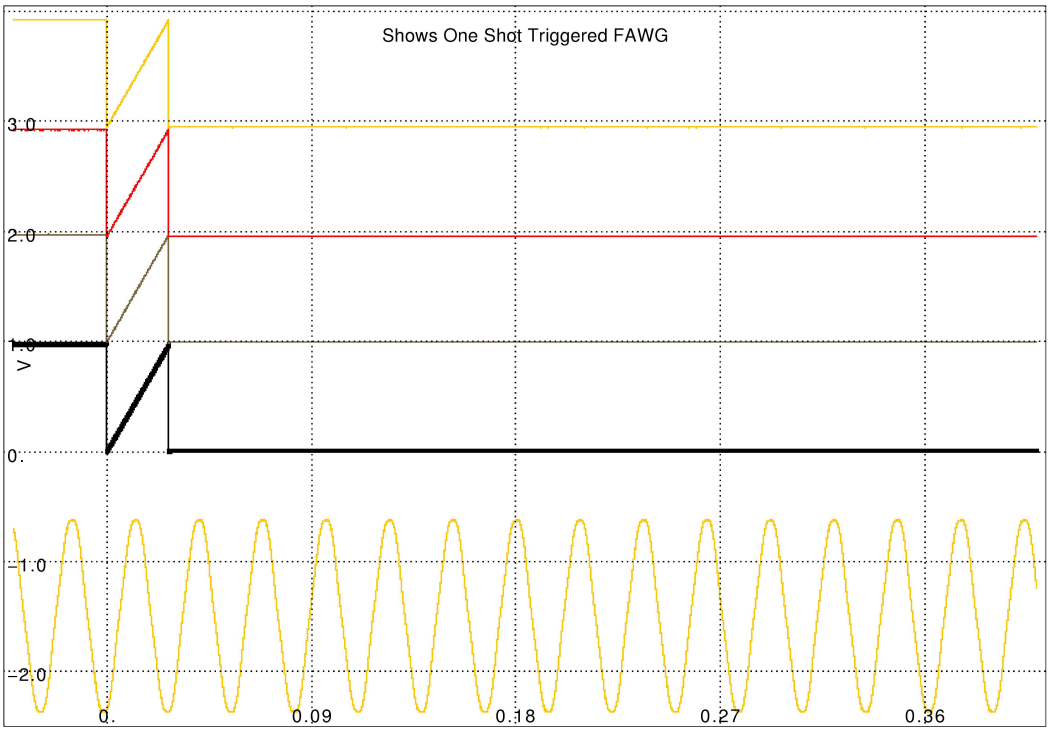
Some time later, a full data set is available in memory. By using *MDSplus* Events, the display is refreshed after each download, and with the full data set we get the full resolution upload shown second. Note that the timebase is correct in both cases.

Typical time to download a full 384MB data set ~90s, it probably isn't a good idea to plot the full data set in `jScope`.

Stride 1000 - sub sampled plot available "instantly" AI clock: 350kHz, AO clock : 17.5kHz



Full Resolution Plot available after upload (up to 90s).



3 Example MDSplus Tree Structure

```
MDSplus tree structure in the examples below:
MDSIP server : kilda
MDS tree     : iroko

[pgm@kilda ~]$ mdstcl
TCL> set tree iroko
TCL> dir

\IROKO::TOP

ACQ196_010 ACQ196 AI data goes here  - SIGNALS!
ACQ196_010AO         ACQ196 AO data sourced from here (shot 1). -
simple vectors of shorts
ACQ216_023

TCL> set def ACQ196_010AO
TCL> dir

\IROKO::TOP.ACQ196_010AO

 :CH01          :CH02          :CH03          :CH04          :CH05
 :CH06          :CH07          :CH08          :CH09          :CH10
 :CH11          :CH12          :CH13          :CH14          :CH15
 :CH16

\IROKO::TOP.ACQ196_010

 :CH01          :CH02          :CH03          :CH04          :CH05
 :CH06          :CH07          :CH08          :CH09          :CH10
...
 :CH81          :CH82          :CH83          :CH84          :CH85
 :CH86          :CH87          :CH88          :CH89          :CH90
 :CH91          :CH92          :CH93          :CH94          :CH95
 :CH96
```

MDSplus ThinClient commands and wrappers are able to iterate sets of channels with a single command, provided the channel node name is expressed in terms of a channel number. The programs use *sprintf*() to create the node name.

D-TACQ provides utility scripts to create such a tree structure:
http://www.d-tacq.com/resources/make_acqtree.tar

Example - first with 16 nodes for *AO*, second with separate nodes for strided data, for an alternate case where it might be preferable to store strided data separate to the full data set.

```
./make_acqtree iroko ACQ196_010AO,16 ACQ196_010,96
./make_acqtree mukwa ACQ196_010S,96 ACQ196_010,96
```

AO Ramps created using gnu-**octave** as described here:
http://www.d-tacq.com/details_page.php?prod_id=applications&page_id=4

4 [Option: Target-side] Setup Procedure:

For initial setup, an easy option is to simply type at a login terminal on the target card.

This procedure is easily automated by saving as a shell script. Be sure to safe to non-volatile storage (/`ffs`) if you want to keep it.

```
Uses two cards
acq196_010 - runs the FAWG
acq196_002 - external clock and trigger.

# Master: acq196_002:
acqcmd setInternalClock 250000 DO0
set.route d0 in fpga out pxi
set.route d3 in fpga out pxi
acqcmd setDIO 1--1--

# Slave: acq196_010
# clock and trigger input from PXI,
set.route d0 in pxi out fpga
set.route d3 in pxi out fpga

# AI uses DI0 external clock, DI3 as event0
set.ext_clk DI0 falling
acqcmd setExternalClock DI0

set.event0 DI3 falling

# AO used DI0/20 external clock, DI3 as trigger
set.ao_clk DI0 falling
set.ao_trig DI3 falling
set.dtacq FAWG_div 20

# AWG: upload data from MDSplus
# we have valid data in shot 1
# MDSIP host: kilda Tree: iroko Node: ACQ196_010AO[:CH%02d]
/usr/local/CARE/load16AOmds kilda iroko:1 ACQ196_010AO

# Alternate: generate local sine wave
# fungen ch K A N   ch = K + A sin(2PI/N)
fungen 3 0 2 64

# Configure AI - 1024 pre, 100K post
acqcmd setModeTriggeredContinuous 1024 102400

# executes immediately _before_ setArm
set.dtacq pre_arm_hook "set.AO commit 0"
# executes immediately _after_ setArm.
setArm.set.dtacq post_arm_hook "set.AO commit 0x24"

# Arm the shot.
acqcmd setArm

# sometime later, supply a trigger eg using DIO page on MB.
```

4.1 Target Side Scripts:

/usr/local/mdsplus/bin/mdsfuns.sh : utility shell functions
/usr/local/mdsplus/bin/load16AOmds : upload data to AWG
/usr/local/mdsplus/bin/async_storeAImds : async download AI data

4.1.1 Loads the AWG from MDSplus:

Run this on the target as part of initialization

/usr/local/mdsplus/bin/load16AOmds

```
#!/bin/sh
# load FAWG waveform set.
# the waveforms are stored as vectors of shorts.

load16() {
        rhost=$1
        ts=$2
        node=$3
        AOf=/dev/acq196/AO/f

        shot=0
        tree=${ts%:*}

        if [ "$tree" != "$ts" ]
        then
                shot=${ts#*:}
        fi
        mdsConnect $rhost
        mdsOpen $tree $shot

        let ch=1
        while [ $ch -le 16 ]
        do
                CH=`printf "%02d" $ch`
                mdsValue --output $AOf.$CH ${node}:CH$CH
                let ch="$ch+1"
        done

        mdsClose $tree
}

if [ $# -eq 3 ]
then
        load16 $1 $2 $3
else
        echo load16 rhost tree node
fi
```

- Example invocation:

```
# AWG: load data from MDSplus
# we have valid data in shot 1
# MDSIP host: kilda Tree: iroko Node: ACQ196_010AO[:CH%02d]
/usr/local/mdsplus/bin/load16AOmds kilda iroko:1 ACQ196_010AO
```

4.1.2 Stores the AI data to MDSplus

Part of **usr/local/mdsplus/bin/mdsfuns.sh**, called from **acq200.pp**
Runs automatically after the shot.

NB: this is not default behaviour. To enable, please see 5.4.1.

```
# target push mdsPut. assumes mdsConnect has been made previously
# do_mdsPutCh <tree> [channels]
do_mdsPutCh() {
        tree=$1
        node=${2:-$HN}
        tb=${3:-:,:,1}
        range=${4:-:}

        log mdsOpen $tree
        mdsOpen $tree
        mdsPutCh --expr %calsig --timebase $tb --field $node:CH%02d
$range
        mdsClose $tree
        log mdsClose $tree
}

do_mdsSetEvent() {
        expression=`printf "setevent('%s',42ub)" $1`
        mdsValue \"$expression\"
}
HN=`hostname`

# example mds put - full data to node $HN
do_mdsPutCh iroko ${HN}
do_mdsSetEvent ${HN}_FULL
```

Sample expanded channel expression -

```
mdsPutCh expands to this for each channel:
MdsPut 6  Build_Signal(Build_With_Units((10.000000 + (-10.000000 -
10.000000)*($VALUE - -32768)/(+32767 -  -32768))
,'V'),$,Build_Dim(Build_Window(0, 102400 - -10240,
Build_With_Units((-10240*ACQ196_010:_DT), 's')),
Build_With_Units(Build_Range(( -10240*ACQ196_010:_DT),
(102400*ACQ196_010:_DT), ACQ196_010:_DT), 's')))
```

Translates as:

```
MdsPut                          Thin Client command
6                               Socket ID for connection to MDSIP
acq196_010:CH04                 Node in tree
        Build_Signal(           Expression ...

        Build_With_Units(       Voltage Calibration.
                (9.9930 + (-9.9345 - 0.9930)*($VALUE - -32768)
                /(+32767 -  -32768)),
                'V'),
        $                       place holder for RAW (array shorts)
        Build_Dim(
                                The timebase : ACQ196_010:_DT holds sample interval
        Build_Window(0, 102400 - -10240,
        Build_With_Units((-10240*ACQ196_010:_DT), 's')
        Build_With_Units(Build_Range(( -10240*ACQ196_010:_DT)
        (102400*ACQ196_010:_DT), ACQ196_010:_DT), 's')
))
```

Please note: the actual formula is defined in */root/.mdsPutCh.nosh*, a text initialization file, and so it is easy to modify the formula as required.

For a full description of **`mdsPutCh`**, please see *2GUG*.

5 Remote Setup Procedure.

Assumes *MDSplus* tree setup as show, using canonical naming.

If the channel node name is not in the canonical form, the commands will still work, but must be called explicitly for each channel in turn.

5.1 Example Host Side Scripts

Host side scripts in this Directory:

`/home/pgm/PROJECTS/ACQ200/project/CARE/`

`awgmdsplus.sh`: functions to
- upload *ACQ196* AWG from *MDSplus*,
- run a shot
- download to *MDSplus*

`run.awgmdsplus`: run wrapper to call all functions in turn

`VERBOSE: =0 : quiet, =1 : loud,  =2 loud, does nothing.`

`=> a dry run using VERBOSE=2 is a good idea.`

`TRANSPORTS:`

`choice of either *HUB* or *SOAP* (WebService) transports.`

5.2 HUB Upload and Shoot Example

Uses embedded script **`/usr/local/mdsplus/bin/load16AOmds`** to automate the upload

```
To use HUB (default):

First, ensure hub is loaded and running correctly, and that the
executable acqcmd is available from
~/bin/acqcmd.  acqcmd should support the "-s" shell command option
(vintage 2003->).
```

5.2.1 Using a second card to supply PXI CLK, TRIG

```
Example:
MB: 20 to supply CLK and TRIG, con
BUT: 10
[pgm@islay CARE]$ VERBOSE=2 ./run.awgmdsplus 20 10 kilda iroko:1
ACQ196_010AO
awgmdsplus $Revision: 1.2 $ MB:20 BUT:10

# master clock setup :
/home/pgm/bin/acqcmd -s 20 set.route d0 in fpga out pxi
/home/pgm/bin/acqcmd -s 20 set.route d3 in fpga out pxi
/home/pgm/bin/acqcmd -s 20 set.mas_clk DO0
/home/pgm/bin/acqcmd -b 20 -- setDIO 1--1--
/home/pgm/bin/acqcmd -b 20 getDIO
/home/pgm/bin/acqcmd -b 20 setInternalClock 250000 DO0

# BUT configure signal routing, AI clock, AO clock.
# AI will run a pre-post capture, using DI3 falling as EVENT0
# AO will run a one-shot, using DI3 falling as TRIG

/home/pgm/bin/acqcmd -s 10 set.route d0 in pxi out fpga
/home/pgm/bin/acqcmd -s 10 set.route d3 in pxi out fpga
/home/pgm/bin/acqcmd -s 10 set.ext_clk DI0 falling
/home/pgm/bin/acqcmd -b 10 setExternalClock DI0
/home/pgm/bin/acqcmd -s 10 set.event0 DI3 falling
/home/pgm/bin/acqcmd -s 10 set.ao_clk DI0 falling
/home/pgm/bin/acqcmd -s 10 set.ao_trig DI3 falling

# Configure the FAWG to run at CLK/50, and load data from MDSplus.
Data in kilda:iroko:ACQ196_010AO:CHxx

/home/pgm/bin/acqcmd -s 10 set.dtacq FAWG_div 50
/home/pgm/bin/acqcmd -s 10 /usr/local/mdsplus/bin/load16AOmds kilda
iroko:1 ACQ196_010AO

# Configure the capture.

/home/pgm/bin/acqcmd -b 10 setModeTriggeredContinuous 10240 102400
/home/pgm/bin/acqcmd -s 10 set.dtacq pre_arm_hook
/usr/local/bin/set.AO commit 0
/home/pgm/bin/acqcmd -s 10 set.dtacq post_arm_hook
/usr/local/bin/set.AO commit 0x24
/home/pgm/bin/acqcmd -b 10 setArm

# Toggle a falling edge on DI3 (use ACQ216 DIO page) to trigger.

VERBOSE=1 ./run.awgmdsplus 20 10 kilda iroko:1 ACQ196_010AO
```

```
Example Output:
[pgm@islay CARE]$ VERBOSE=1 ./run.awgmdsplus 20 10 kilda iroko:1
ACQ196_010AO awgmdsplus $Revision: 1.2 $ MB:20 BUT:10
/home/pgm/bin/acqcmd -s 20 set.route d0 in fpga out pxi
EOF 1 0
/home/pgm/bin/acqcmd -s 20 set.route d3 in fpga out pxi
EOF 2 0
/home/pgm/bin/acqcmd -s 20 set.mas_clk DO0
EOF 3 0
/home/pgm/bin/acqcmd -b 20 -- setDIO 1--1--
ACQ32:
/home/pgm/bin/acqcmd -b 20 getDIO
ACQ32:getDIO=1HH1HH
/home/pgm/bin/acqcmd -b 20 setInternalClock 250000 DO0
ACQ32:
/home/pgm/bin/acqcmd -s 10 set.route d0 in pxi out fpga
EOF 1 0
/home/pgm/bin/acqcmd -s 10 set.route d3 in pxi out fpga
EOF 2 0
/home/pgm/bin/acqcmd -s 10 set.ext_clk DI0 falling
EOF 3 0
/home/pgm/bin/acqcmd -b 10 setExternalClock DI0
ACQ32:
/home/pgm/bin/acqcmd -s 10 set.event0 DI3 falling
EOF 4 0
/home/pgm/bin/acqcmd -s 10 set.ao_clk DI0 falling
EOF 5 0
/home/pgm/bin/acqcmd -s 10 set.ao_trig DI3 falling
EOF 6 0
/home/pgm/bin/acqcmd -s 10 set.dtacq FAWG_div 50
EOF 7 0
/home/pgm/bin/acqcmd -s 10 /usr/local/mdsplus/bin/load16AOmds kilda
iroko:1 ACQ196_010AO
MDS_SOCK=6
OK
DTYPE_SHORT, dim:1 [512] length 2 /dev/acq196/AO/f.01
DTYPE_SHORT, dim:1 [512] length 2 /dev/acq196/AO/f.02
DTYPE_SHORT, dim:1 [512] length 2 /dev/acq196/AO/f.03
...
DTYPE_SHORT, dim:1 [512] length 2 /dev/acq196/AO/f.13
DTYPE_SHORT, dim:1 [512] length 2 /dev/acq196/AO/f.14
DTYPE_SHORT, dim:1 [512] length 2 /dev/acq196/AO/f.15
DTYPE_SHORT, dim:1 [512] length 2 /dev/acq196/AO/f.16
OK
EOF 8 0
/home/pgm/bin/acqcmd -b 10 setModeTriggeredContinuous 10240 102400
ACQ32:
/home/pgm/bin/acqcmd -s 10 set.dtacq pre_arm_hook
/usr/local/bin/set.AO commit 0
EOF 9 0
/home/pgm/bin/acqcmd -s 10 set.dtacq post_arm_hook
/usr/local/bin/set.AO commit 0x24
EOF 10 0
/home/pgm/bin/acqcmd -b 10 setArm
ACQ32:
```

5.2.2 Alternate: To use Front Panel clocking:

```
Procedure:

[pgm@islay CARE]$ . hub-transport.sh
[pgm@islay CARE]$ . awgmdsplus.sh
awgmdsplus $Revision: 1.3 $ MB:20 BUT:10
[pgm@islay CARE]$ VERBOSE=2 init_signals lemo
/home/pgm/bin/acqcmd -s 10 set.route d0 in lemo out fpga
/home/pgm/bin/acqcmd -s 10 set.route d3 in lemo out fpga
/home/pgm/bin/acqcmd -s 10 set.ext_clk DI0 falling
/home/pgm/bin/acqcmd -b 10 setExternalClock DI0
/home/pgm/bin/acqcmd -s 10 set.event0 DI3 falling
/home/pgm/bin/acqcmd -s 10 set.ao_clk DI0 falling
/home/pgm/bin/acqcmd -s 10 set.ao_trig DI3 falling
/home/pgm/bin/acqcmd -s 10 set.dtacq FAWG_div 50


Output Log:
[pgm@islay CARE]$ VERBOSE=1 init_signals lemo
/home/pgm/bin/acqcmd -s 10 set.route d0 in lemo out fpga
EOF 1 0
/home/pgm/bin/acqcmd -s 10 set.route d3 in lemo out fpga
EOF 2 0
/home/pgm/bin/acqcmd -s 10 set.ext_clk DI0 falling
EOF 3 0
/home/pgm/bin/acqcmd -b 10 setExternalClock DI0
ACQ32:
/home/pgm/bin/acqcmd -s 10 set.event0 DI3 falling
EOF 4 0
/home/pgm/bin/acqcmd -s 10 set.ao_clk DI0 falling
EOF 5 0
/home/pgm/bin/acqcmd -s 10 set.ao_trig DI3 falling
EOF 6 0
/home/pgm/bin/acqcmd -s 10 set.dtacq FAWG_div 50
EOF 7 0
```

5.3 Example channel by channel control from host.

For an example of controlling the AO upload entirely from the host side, see **`init_fawg2`**

```
[pgm@islay CARE]$ . ./hub-transport.sh
[pgm@islay CARE]$ . ./awgmdsplus.sh
awgmdsplus $Revision: 1.2 $ MB:20 BUT:10
[pgm@islay CARE]$ init_fawg2 kilda iroko:1 ACQ196_010AO
acqcmd -s 10 mdsConnect kilda
MDS_SOCK=6
EOF 1 0
acqcmd -s 10 mdsOpen iroko 1
OK
EOF 2 0
acqcmd -s 10 -- mdsValue --output /dev/acq196/AO/f.01
ACQ196_010AO:CH01
DTYPE_SHORT, dim:1 [512] length 2 /dev/acq196/AO/f.01
EOF 3 0

...

acqcmd -s 10 -- mdsValue --output /dev/acq196/AO/f.14
ACQ196_010AO:CH14
DTYPE_SHORT, dim:1 [512] length 2 /dev/acq196/AO/f.14
EOF 16 0
acqcmd -s 10 -- mdsValue --output /dev/acq196/AO/f.15
ACQ196_010AO:CH15
DTYPE_SHORT, dim:1 [512] length 2 /dev/acq196/AO/f.15
EOF 17 0
acqcmd -s 10 -- mdsValue --output /dev/acq196/AO/f.16
ACQ196_010AO:CH16
DTYPE_SHORT, dim:1 [512] length 2 /dev/acq196/AO/f.16
EOF 18 0
acqcmd -s 10 mdsClose iroko
OK
EOF 19 0
```

What does it mean?	
`acqcmd` `-s 10`	"Slot 10 Card" via HUB
`--`	disable popt(3) switch processing
`mdsValue`	remote command mdsValue
`--output /dev/acq196/AO/f.01`	save result here
`ACQ196_010AO:CH01`	evaluate this expression

5.4 Data Download to MDSplus

Target Push methods to upload the data.

Assumes channels in canonical *NODE:CH%02d* format

/usr/local/mdsplus/bin/mdsfuns.sh:

do_mdsPutCh is a convenience wrapper on **mdsPutCh.**

5.4.1 Automated Post Shot Download

Initiated from modified file : **/usr/local/bin/acq200.pp**
(save non-volatile copy in **/ffs/dropbear/usr/local/bin/acq200.pp**).
The file has template examples, commented out by default.

5.4.1.1 Minimum latency strided download

```
BEFORE TRANSFORM:

# example early mds put - decimated stride=1000 data to 'd'ecimate
node -
#    tests show display within  2s of end of capture
(TIMEBASE=1,:,1000 do_mdsPutCh iroko )
do_mdsSetEvent ${HN}_STRIDE
```

5.4.1.2 Full Data Set download

```
AFTER TRANSFORM:
(do_mdsPutCh iroko)
```

5.4.2 On Demand target push Downloads.

First, naturally it is possible to call the Thin Client commands
mdsConnect, **mdsOpen**, **mdsPut**, **mdsPutCh**, **mdsValue**, **mdsClose**, **mdsDisconnect**
directly via the remote shell mechanism

If your data is in the canonical *NODE:CH%02d* format, then **do_mdsPutCh** can simplify the commands. If not in the canonical form, **mdsPutCh** may be invoked for each channel in turn.

5.4.2.1 Synchronous Download

Host side software invokes the download and blocks until the transfer is complete.

Example:

```
[pgm@islay CARE]$ time acqcmd -s 10 '.
/usr/local/mdsplus/bin/mdsfuns.sh; do_mdsPutCh iroko'
OK
OK
EOF 1 0

real    0m9.977s
user    0m0.000s
sys     0m0.004s
```

5.4.2.2 Asynchronous Download

Host side software schedules an download as a batch job, control returns to the host immediately and the job completes in the background.

```
[pgm@islay CARE]$ time acqcmd -s 10
'/usr/local/mdsplus/bin/async_storeAImds  iroko'
JOB /tmp/storeAImds.job created
EOF 2 0

real    0m0.213s
user    0m0.000s
sys     0m0.000s



Full timing check (the rsh call is needed because the EVENT lives on
kilda)

[pgm@islay CARE]$ time acqcmd -s 10 'EVENT=ACQ196_010_FULL
/usr/local/mdsplus/bin/async_storeAImds  iroko';time ssh pgm@kilda '.
/usr/local/mdsplus/setup.sh;wfevent ACQ196_010_FULL'
JOB /tmp/storeAImds.job created
EOF 1 0

real    0m0.312s
user    0m0.004s
sys     0m0.000s

real    0m9.865s
user    0m0.016s
sys     0m0.008s


So, for the single card case, async isn't any faster than sync. But
if a single host is controlling multiple cards, then there is scope
for running all the uploads in parallel, and then the limit of
performance shifts from the speed of a single ACQ2xx card to the
speed of the MDSIP server.
```

5.5 Using SOAP (Web Service) Transport

The same sequences initiated through the HUB can also be initiated through the *SOAPI* Web Service interface.

SOAPI provides remote able versions of **acqcmd**, **acq2sh**.

The example shows that actually the same script is run, but the underlying TRANSPORT has be changed.

5.5.1 SOAP vs HUB

The D-TACQ "*SOAPI*" implementation makes the **acqcmd** and **acq2sh** commands available as remote procedure calls. SOAP has the advantage of being cross platform, and by using WSDL service definition, in principle at least, no host side driver at all is required. Alternatively, the HUB may be slightly faster in use and easier to integrate in existing systems.

Assuming a HUB driver has been loaded and configured, then the basic differences from a user point of view are:

- different executables **acqcmd**, **acq2sh** are used.
- cards referenced by *URL* rather than logical slot number - it would of course be easy to map "slot" to "URL" if required.

5.5.2 SOAPI installation

```
To use SOAP :
Unpack the SOAPI package, version

http://localhost/~pgm/www.d-tacq.com/swrel/SOAPI.200610041513.tgz

or better.
```

5.5.3 Upload and Shoot Example repeated

```
This code assumes a subdirectory ./SOAPI to connect to the files:
eg
ln -s ~/MY_SOAPI_DIR  SOAPI


VERBOSE=1 USE_SOAP=1 ./run.awgmdsplus acq216_023 acq196_010 kilda
iroko:1 ACQ196_010AO
```

Example Output:

```
[pgm@islay CARE]$ VERBOSE=1 USE_SOAP=1 ./run.awgmdsplus acq216_023
acq196_010 kilda iroko:1 ACQ196_010AO
awgmdsplus $Revision: 1.2 $ MB:acq216_023 BUT:acq196_010
./SOAPI/acq2sh --url http://acq216_023:6666 set.route d0 in fpga out
pxi

./SOAPI/acq2sh --url http://acq216_023:6666 set.route d3 in fpga out
pxi

./SOAPI/acq2sh --url http://acq216_023:6666 set.mas_clk DO0

./SOAPI/acqcmd --url http://acq216_023:6666 -- setDIO 1--1--
ACQ32:

./SOAPI/acqcmd --url http://acq216_023:6666 getDIO
ACQ32:getDIO=1HH1HH

./SOAPI/acqcmd --url http://acq216_023:6666 setInternalClock 250000
DO0
ACQ32:

./SOAPI/acq2sh --url http://acq196_010:6666 set.route d0 in pxi out
fpga

./SOAPI/acq2sh --url http://acq196_010:6666 set.route d3 in pxi out
fpga

./SOAPI/acq2sh --url http://acq196_010:6666 set.ext_clk DI0 falling

./SOAPI/acqcmd --url http://acq196_010:6666 setExternalClock DI0
ACQ32:

./SOAPI/acq2sh --url http://acq196_010:6666 set.event0 DI3 falling

./SOAPI/acq2sh --url http://acq196_010:6666 set.ao_clk DI0 falling

./SOAPI/acq2sh --url http://acq196_010:6666 set.ao_trig DI3 falling

./SOAPI/acq2sh --url http://acq196_010:6666 set.dtacq FAWG_div 50

./SOAPI/acq2sh --url http://acq196_010:6666
/usr/local/mdsplus/bin/load16AOmds kilda iroko:1 ACQ196_010AO
MDS_SOCK=6
OK
DTYPE_SHORT, dim:1 [512] length 2 /dev/acq196/AO/f.01
...
DTYPE_SHORT, dim:1 [512] length 2 /dev/acq196/AO/f.15
DTYPE_SHORT, dim:1 [512] length 2 /dev/acq196/AO/f.16
OK
```

```
./SOAPI/acqcmd --url http://acq196_010:6666
setModeTriggeredContinuous 10240 102400
ACQ32:

./SOAPI/acq2sh --url http://acq196_010:6666 set.dtacq pre_arm_hook
/usr/local/bin/set.AO commit 0

./SOAPI/acq2sh --url http://acq196_010:6666 set.dtacq post_arm_hook
/usr/local/bin/set.AO commit 0x24

./SOAPI/acqcmd --url http://acq196_010:6666 setArm
ACQ32:
```

Document ends.