ACQ2xx EPICS User Guide

Rev	***Date***	***Description***
1	11/3/05	Created
2	2/7/08	Simpler installation using /bigffs
3	31/12/08	Faster implementation with dsAcqAi, dsLxbBi, cp7452 support
4	13/4/09	

Document created using OpenOffice.Org www.openoffice.org.

1 Introduction

1.1 Scope

This document describes how to build and use the D-TACQ port of the EPICS IOC and related database records.

1.2 References

1. EPICS: Experimental Physics and Industrial Control System http://www.aps.anl.gov/epics/
2. D-TACQ 2G User Guide

2 Summary of the D-TACQ EPICS Port

- xscale processor port of full EPICS IOC, runs on 2G intelligent digitizers
- dsSysfs device support for easy connection between EPICS database records and ACQ2xx system files
- hi-performance dsAcqAi support for AI current value.
- Hi-performance COS support using dsLxbBi for BI values
- Control - SETARM, SETABORT functions
- Monitoring - mean value of each channel. A remote CA client is able to monitor the data trend concurrent to the main data full rate data flow from ADC array to main memory.
- System health - UPTIME, TEMP1, TEMP2 (on board temperature sensing).
- Easy to create other records.
- No attempt to access bulk data, it is assumed that this will either be by streaming (during the shot), or by post-shot bulk data transfer (ftp, mdsplus, http put). It may be that an EPICS waveform or compressed record will be appropriate for this (up to 1gig byte data sets).

3 Custom Device Support

3.1 dsSysfs device support.

Device support module allows very simple access to file hooks to system and acquisition data, typically exported as files, either in sysfs or a custom Virtual File System (VFS).

The file support uses the aiRecord record type, encoding the VFS file name and mode into the INP field. The device support also caters for running commands to acquire the data.

The same aiRecord type is also used for writing simple control functions (ARM, ABORT), although this is most probably an abuse of the record type.

```
Sysfs relies on an encoded INP field.

Encoding as follows:
@[#!]path[,ftype][,fmt] [args]
@ : literal '@' - EPICS seems to need this #!: path is a program, execute it
ftype: r|w : read or write [implied default: r] a : ascii data (default) b<N> :
binary data, N bytes (N =2,4 supported)
fmt: : for ascii data, sscanf() format for conversion
args: regular space separated command line arguments for #! case
WARNING:
    • the INP record is limited to 40 chars including the @ so scope for
      formatting is VERY LIMITED.
    • ,ftype,fmt may NOT contain spaces
Examples:
    • field(INP, "@/dev/mean/$(cid) rb4) # $(cid) : channel id 01..96, read 4
      byte mean value
    • field(INP, "@#!sysmon -T 1") # get the temperature by running sysmon, read
      ascii value
    • field(INP, "@#!acqcmd setArm") # run the setArm command (using an
      aiRecord?).
```

The device support is documented using inline Doxygen documentation, and 5 variants are supported:

```
sysfs.dbd:
device(ai,INST_IO,devSysfsAi,"dsSysfsAi")
device(ao,INST_IO,devSysfsAo,"dsSysfsAo")
device(bi,INST_IO,devSysfsBi,"dsSysfsBi")
device(bo,INST_IO,devSysfsBo,"dsSysfsBo")
device(waveform,INST_IO,devSysfsWfAi,"dsSysfsWfAi")

** ai has been replaced by the high-performance I/O Intr driven :
# BiBo Binary IO driver Device Support
driver(drvLxb)
device(bi,  INST_IO, devLxb,       "dsLxbBi")
device(mbbi,    INST_IO, devLxbMbbi,   "dsLxbMbbi")
[pgm@hoy ACQ2IOC]$ cat dbd/acqAi.dbd
# BiBo Binary IO driver Device Support
driver(drvAcqAi)
device(ai,  INST_IO, devAcqAi,       "dsAcqAi")
```

The ai, bi types are I/O Intr driven. The AI update rate is controlled by the input sample rate, and parameters of the mean device. The BI records update on COS. The BI driver polls for COS at 100Hz, so the maximum update rate will be 100Hz.
The waveform is triggered by end of shot, and the length of the waveform is the POST length of the

capture.

NB: it max be necessary to increase the value of **EPICS_CA_MAX_ARRAY_BYTES** to accomodate the waveform (see **start.ioc** script)

3.2 lxbBi

Bi records are efficiently handled using blocking IO.

lxBi records assume that the read() call of the device driver will return immediately the first time with an array of bits corresponding to the input state of the port. On subsequent calls, read() will block until there is a Change Of State (COS). The lxbBi device support will signal the changed state to the appropriate records. D-TACQ has implemented 3 device drivers that conform in this way:

- di6 - /dev/di6 - access to FPGA DI lines
- di32 - /dev/di32 - access to DIO32 inputs
- /dev/cp7452/SLOT - access to Adlink CP7452

The device drivers do not actually use a hardware COS interrupt, but instead use a kernel waitq construct to poll every system tick. This results in a 50Hz response, with very little CPU overhead. If a faster overhead is required, it would be straightforward to move the polling signal onto the aux timer interrupt at perhaps 1kHz. D-TACQ does not recommend use of hardware COS interrupts, although this is equally possible using this device driver pattern.

3.3 dsAcqAi

AI records are handled efficiently by this device support.

A core monitor thread reads a single device (/dev/mean/XX), returning a single record for all channels. The monitor then sends an I/O Intr message to cause the records to scan. This implementation is far more efficient than using dsSysFs.

4 Build and Installation

4.1 Build BASE

```
# The standard build now works as of 3.4.9
```

4.2 BUILD acq2ioc

```
# unpack IOC sources from acq2ioc.src.*.tgz
edit CONFIGURE/RELEASE to give path to BASE in your system
make
sudo ./ACQ200.RELEASE    # creates release tarball ./RELEASE/acq2ioc.tgz
```

4.3 Install on target flash disk

For a single card, it's easy to do this manually. For multiple cards and/or record keeping, it's a good idea to create a deploy script.

Example deploy script:

```
#!/bin/bash

ACQ2IOC=acq2ioc.081223123909.tgz
CP7452=cp7452_linux_drv-200812231240.tgz

PAYLOAD="$ACQ2IOC $CP7452 start.ioc.local epics-diff.tar"

NEATLS="ls -l /bigffs | cut -c 57-"

echo deploy EPICS support

for host in $*
do
        ssh root@$host 'echo $(hostname) $(get.release.versions)'

        ssh root@$host rm -f /bigffs/\*ioc\*
        ssh root@$host rm -f /bigffs/cp7452\*
        echo BEFORE:
        ssh root@$host $NEATLS
        scp $PAYLOAD root@$host:/bigffs
        ssh root@$host "cd /bigffs;ln -s $ACQ2IOC acq2ioc.tgz"
        ssh root@$host "cd /bigffs;ln -s $CP7452 cp7452_linux_drv.tgz"
        echo AFTER:
        ssh root@$host $NEATLS
done
```

4.4 Set Option

Uncomment "`EPICS=YES`" entry in **/ffs/rc.local.options**

4.5 Check that IOC is running:

on the target : a **ps** display will show about a dozen **acq2ioc** threads.

You can get access to the epics> prompt at any time by logging in on telnet as user "epics", pass "acq2ioc".

on an EPICS CA :

```
camonitor acq216_023_UPTIME    # shows /proc/uptime updated once per second.
```

4.6 The IOC runs with the following default records created:

Default name convention: `hostname`_ # you can change this in the boot script to be a functional name instead eg POWERMON2_

Record Name	Function
hostname_AIXX acq216_023_CH01	Mean value of Channel XX, update period 0.1 second Output is calibrated Volts.
hostname_DIX acq216_023_DI0	Value of system digital input bit
hostname_BIXX acq196_267_BI00	Value of DIO32 input bit
hostname_BOXX acq196_267_BO31	Value of DIO32 output bit
hostname_AOXX acq196_267_AO16	Value of AO16 analog output channel
hostname_CHXX acq196_267_CH13	Captured waveform update at end of shot
Knobs:	
acq216_023_SETABORT	caput acq216_023_SETABORT 1 # aborts capture
acq216_023_SETARM	caput acq216_023_SETARM 1 # arms capture
acq216_023_TEMP1	Card temperature 1, updated 10 seconds
acq216_023_TEMP2	Card temperature 2, updated 10 seconds
acq216_023_UPTIME	Linux uptime updated 1 second (use ad heartbeat).
acq216_023_STATE	Current capture state of the card
acq216_023_TRIG	Sends a hardware trigger to the capture
acq216_023_update_ms	Update rate for AI
Peripheral Cards:	
hostname_slot_BIXXX acq196_267_4_BI001	CP7452 Binary input
hostname_slot_BOXXX acq196_267_4_BO001	CP7452 Binary output
hostname_ao32_slot_AOxx acq196_267_ao32_5_AO32	AO32 card, analog output channel
hostname_ao32_slot_CHxx acq196_267_ac32_5_CH01	AO32 card, AWG waveform definition
hostname_ao32_slot_BOxx acq196_267_ao32_5_BO00	AO32 card, Binary Output bit definition
hostname_ao32_slot_WOxx acq196_267_ao32_5_WO63	AO32 card, Binary AWG waveform definition
hostname_ao32_slot_PWMxx acq196_267_ao32_5_PWM63	AO32 card, PWM output definition

4.7 Operation Setup

The Mean value devices are updated while the card is capturing data

A simple setup script to run a full speed capture is:

```
PRE=10
POST=1000000                                        # this will be WF len
CLK=100000                                          # 100000 on ACQ196
acqcmd setInternalClock $CLK
set.sys /sys/module/acq200_mean/parameter/skip 20   # gives 20Hz update0
set.route d3 in lemo out fpga
set.event0 DI3 falling
acqcmd setModeTriggeredContinuous $PRE $POST
acqcmd setArm                                       # or use caput SETARM
```

```
Testing:
Loopback DIO32 to AI01.
Loopback CP7452 DO to DI

Set up some test data:

# put a 10Hz walking bit on the AI channels
evgen -r 100 -d /dev/dio32 10000 010000 001000 0001000 &

# toggle c7252 io at 50Hz:

evgen -r 20 -d /dev/cp7452.4/DO0 00000000 111111111 &
```

5 Performance.

Example:

ACQ196 with 600MHz CPU.

Records	**Number**	**Update Rate**
AI	96 AI	20Hz
BI	6 (di6)	COS
BI	32 (di32)	COS
BI	128 (cp7452)	COS
AO	16	
BO	32	
BO	128 (cp7452)	
CONTROLS	11	
TOTAL RECORDS	449	

On a 600MHz system, under the conditions above, we record maximum acq2ioc process usage at 20%.

5.1 *20 updates, 1s*

```
acq196_281_AI02               2008-12-28 17:40:11.017621 4.93958
acq196_281_AI02               2008-12-28 17:40:11.068543 4.93013
acq196_281_AI02               2008-12-28 17:40:11.118043 4.92037
acq196_281_AI02               2008-12-28 17:40:11.169694 4.9472
acq196_281_AI02               2008-12-28 17:40:11.218303 4.97098
acq196_281_AI02               2008-12-28 17:40:11.269415 4.96884
acq196_281_AI02               2008-12-28 17:40:11.321462 4.95878
acq196_281_AI02               2008-12-28 17:40:11.372670 4.93043
acq196_281_AI02               2008-12-28 17:40:11.423763 4.94141
acq196_281_AI02               2008-12-28 17:40:11.478762 4.97128
acq196_281_AI02               2008-12-28 17:40:11.527749 4.96153
acq196_281_AI02               2008-12-28 17:40:11.578785 4.95055
acq196_281_AI02               2008-12-28 17:40:11.629998 4.93927
acq196_281_AI02               2008-12-28 17:40:11.679448 4.93135
acq196_281_AI02               2008-12-28 17:40:11.732298 4.91794
acq196_281_AI02               2008-12-28 17:40:11.784280 4.96823
acq196_281_AI02               2008-12-28 17:40:11.834891 4.9661
acq196_281_AI02               2008-12-28 17:40:11.884990 4.95055
acq196_281_AI02               2008-12-28 17:40:11.935016 4.93501
acq196_281_AI02               2008-12-28 17:40:11.988625 4.92159
```

Tallies with quoted update of 50ms:

```
[dt100@rhum ~]$ camonitor acq196_281_update_ms
acq196_281_update_ms          2008-12-28 17:39:59.678896 51
```