DAQ software used during HAPPEX-II

Bryan Moffit

Massachusetts Institute of Technology, Cambridge, MA, USA

Abstract

This document describes in detail the software that was used by the HAPPEX data aqcuisition (DAQ) system during HAPPEX-II.

HAPPEX Internal Technote *24 June 2007*

Contents

1 Introduction

This document is meant to provide some detail for most of the DAQ software that was used during HAPPEX-II. Hopefully, it will help with the effort to prepare for future HAPPEX type experiments.

The provided locations of all files are those that existed during HAPPEX-II and may not necessarily exist today.

A tarball (using the same directory structure as described in this document) is provided at http://hallaweb.jlab.org/experiment/HAPPEX/docs/HAPPEXDAQ/happex_daq_software.tgz.

2 CODA

Presented here are the scripts that are run by CODA during the start and end of run sequences.

2.1 startEpicsLogger

Language:	shell script
Requires:	N/A
Location:	~apar/scripts/
Started by:	CODA EB (GO)
Control with:	N/A
Configuration Files:	N/A
Makes calls to:	getruninfo
	fastEpicsLogger
Output Files:	N/A
Log Files:	N/A

Description:

Simple script to execute getruninfo and fastEpicsLogger.

2.2 getruninfo

Language:	bash script
Requires:	N/A
Location:	~apar/scripts/
Started by:	startEpicsLogger
Control with:	N/A
Configuration Files:	N/A
Makes calls to:	~apar/scripts/getrunnumber
Output Files:	~apar/datafile/rcRunNumber
Log Files:	N/A

Description:

Simple script to execute getrunnumber, then store the runnumber into the rcRunNumber.

2.3 getrunnumber

Language:	shell script
Requires:	CODA dpwish
Location:	~apar/scripts/
Started by:	getruninfo
Control with:	N/A
Configuration Files:	N/A
Makes calls to:	CODA MSQL database
Output Files:	N/A
Log Files:	N/A

Description:

Shell script that executes dpwish (from CODA distribution) to obtain the current run-number from CODA's MSQL database.

2.4 fastEpicsLogger

Language:	bash script
Requires:	EPICS caget fileToEvent
Location:	~apar/scripts/
Started by:	startEpicsLogger
Control with:	N/A
Configuration Files:	N/A
Makes calls to:	caget
Output Files:	~apar/epics/fast.epics (inserted into datastream with fileToEvent)
Log Files:	N/A

Description:

This script contains various EPICS variables which are read by caget then appended to fast.epics. This file is then inserted into the datastream (as event type 131) using fileToEvent. Insertion is made every 4 seconds.

Language:	perl script
Requires:	perl lib: GetOpt EPICS caget
Location:	~apar/db/scripts/
Started by:	CODA EB (GO)
Control with:	one argument: –C <CODA configuration>
Configuration Files:	Many. Described below.
Makes calls to:	~apar/db/scripts/getHelicity ~apar/db/scripts/getrunnumber
Output Files:	~apar/db/parity$yr_$run.db
Log Files:	~apar/db/error.log

Description:

In order to Analyze a run correctly, a database must be constructed that accurately defines the devices used, oversampling factor, helicity mode, etc., etc.. A PERL script (createDB) has been written to serve just this purpose. The basic function of this script is to use the type of CODA configuration to determine specific files to combine together. Below is a description of those files with their location.

KEY:

$config	CODA Configuration
$oversample	Total # of Integration Gates per Helicity Window
$crate	A Specific Crate, defined in a CODA Configuration

In all cases, if a specific file is not found.. a default file will be used.

- **~apar/db/config/$config.def**
  Contains the Crates that are included in a given configuration.
- **~apar/db/cuts/$config_$oversample.cuts**
  Contains the cut definitions for a given configuration and oversample value. Also contains "curmon" (indicates which devices to use for beam cuts)
- **~apar/db/dacnoise/$crate.dacnoise**
  Contains calibrated dacnoise slopes for ADCs for a given crate.
- **~apar/db/datamap/$crate.datamap**
  Contains the datamap for a given crate. If the CODA Configuration is a single crate configuration, the TIR and Timing board lines will be replaced with default lines (in order for PAN to retreive important timing and helicity information).
- **~apar/db/helicity/current.helicity**
  Helicity information is retreived automatically from EPICS when createDB is run, and saved to this file, at the beginning of each run
- **~apar/db/misc/$config.misc**

Contains so miscellaneous arguments (e.g. anatype, blindstring) for a given configuration.

- **~apar/db/ped/$crate_$oversample.ped**
  Contains pedestals for ADCs and Scalers for a given crate and oversample value.
- **~apar/db/timebrd/timebrd.cfg**
  Contains information from a timing board (Counting House for Multi-crate configurations): Oversample, Integrate Gate, and Ramp Delay. This file is updated automatically at the beginning of each run.

2.6 startAnalyzer

Language:	bash script
Requires:	N/A
Location:	~apar/scripts/
Started by:	CODA EB (GO)
Control with:	N/A
Configuration Files:	~apar/feedback/feedback_enable.dat
	~apar/bryan/panguin/pan/panguin_enable.dat
Makes calls to:	caget
	makeFFBDB_IHWP
	panFFB
	pan_online (script for panguin backend)
Output Files:	N/A
Log Files:	~apar/feedback/feedback.log
	~apar/feedback/runlog/ffb_$run.log

Description:

Script the handle the execution of the online PAN programs. Checks the configuration files to see if those programs should be run. For panFFB: Runs caget to obtain the current state of the IHWP, then runs makeFFBDB_IHWP with the appropriate argument. For the panguin backend: simply executes the pan_online script.

2.7 *epicsRunStart_parity*

Language:	bash script
Requires:	N/A
Location:	~apar/scripts/
Started by:	CODA EB (GO)
Control with:	one argument: CODA Configuration
Configuration Files:	N/A
Makes calls to:	~apar/scripts/getruninfo
	~apar/scripts/halogRunStart_parity
Output Files:	N/A
Log Files:	N/A

Description:

Simple script execute halogRunStart_parity. CODA Configuration (passed from CODA EB) argument is passed on to halogRunStart_parity.

2.8 *halogRunStart_parity*

Language:	bash script
Requires:	caget
Location:	~apar/scripts/
Started by:	epicsRunStart_parity
Control with:	one argument: CODA configuration
Configuration Files:	N/A
Makes calls to:	caget
	dpwish ~apar/scripts/guis/runstart.tcl
Output Files:	~apar/epics/runfiles/halog_start_$run.epics
	~apar/epics/runfiles/Start_of_Run_$run.epics
Log Files:	

Description:

Script that handles the automatic Start of Parity Run HALOG entry. If not commented out, will display a GUI for shift worker. Saves several relevant EPICS variables to the output files, then submits them to the HALOG. Has a line to process the output files and submit them to a MySQL database. Also has another line to add the run to a MySQL database that takes care of the runlist.

2.9 end_clean

Language:	bash script
Requires:	N/A
Location:	~apar/scripts/
Started by:	CODA EB (END)
Control with:	N/A
Configuration Files:	N/A
Makes calls to:	N/A
Output Files:	N/A
Log Files:	N/A

Description:

Script to handle the process killing of fastEpicsLogger and any caget processes.

2.10 epicsRunEnd_parity

Language:	bash script
Requires:	N/A
Location:	~apar/scripts/
Started by:	CODA EB (GO)
Control with:	one argument: CODA Configuration
Configuration Files:	N/A
Makes calls to:	~apar/scripts/halogRunEnd_parity
Output Files:	N/A
Log Files:	N/A

Description:

Simple script execute halogRunEnd_parity. CODA Configuration (passed from CODA EB) argument is passed on to halogRunEnd_parity.

2.11 halogRunEnd_parity

Language:	bash script
Requires:	caget
Location:	∼apar/scripts/
Started by:	epicsRunStart_parity
Control with:	one argument: CODA configuration
Configuration Files:	N/A
Makes calls to:	caget
	dpwish ∼apar/scripts/guis/runend.tcl
Output Files:	∼apar/epics/runfiles/halog_end_$run.epics
	∼apar/epics/runfiles/End_of_Parity_Run_$run.epics
Log Files:	

Description:

Script that handles the automatic End of Parity Run HALOG entry. If not commented out, will display a GUI for shift worker. Saves several relevant EPICS variables to the output files, then submits them to the HALOG. Has a line to process the output files and submit them to a MySQL database.

2.12 endAnalyzer

Language:	bash script
Requires:	∼apar/scripts/cleanAna
Location:	∼apar/scripts/
Started by:	CODA EB (END)
Control with:	N/A
Configuration Files:	N/A
Makes calls to:	∼apar/scripts/cleanAna
Output Files:	N/A
Log Files:	N/A

Description:

Makes four system beeps. Removes pan.root softlink used by panguin backend. Execute cleanAna with panFFB as the argument.

2.13 cleanAna

Language:	bash script
Requires:	N/A
Location:	∼apar/scripts/
Started by:	endAnalyzer
Control with:	one argument: process name
Configuration Files:	N/A
Makes calls to:	kill
Output Files:	N/A
Log Files:	N/A

Description:

Script that attempts to end the provided process with a SIGHUP 31.

3 Parity Feedback

Feedback on charge asymmetry (A_Q) and position differences ($\Delta x, \Delta y$) is mainly handled by a background process running PAN on data that is retrieved from the Event Transfer (ET) System (which is initiated in CODA, see section 2). This PAN process then communicates to EPICS through a shell command. This section goes over each program that controls and aids this process.

3.1 feedback

Language:	bash script
Requires:	N/A
Location:	∼apar/bin/
Started by:	user
Control with:	one argument: on/off
Configuration Files:	N/A
Makes calls to:	N/A
Output Files:	∼apar/feedback/feedback_enable.dat
Log Files:	∼apar/feedback/feedback.log

Description:

This command effectively turns enables/disables feedback before the start of a CODA run. It simply puts a 1 (feedback on) or 0 (feedback off) into ∼apar/feedback/feedback_enable.dat. An entry (with date) is also appended to ∼apar/feedback/feedback.log.

3.2 panFFB

Language:	C++
Requires:	libcoda.a and pan compiled with ET System (ONLINE=1)
Location:	∼apar/feedback/
Started by:	∼apar/scripts/startAnalyzer
Control with:	∼apar/bin/feedback
Configuration Files:	∼apar/db/control.db (created with makePANFFB)
	∼apar/feedback/runDB/control.db_$run (copy)
Makes calls to:	∼apar/epics/epics_feedback
Output Files:	N/A
Log Files:	∼apar/feedback/feedback.log
	∼apar/feedback/runlog/ffb_$run.log

Description:

Analyzes data acquired from ET and computes changes (differences) to source element DAC values. The A_Q or Δx response to DAC values must by added as parameters to ∼apar/db/control.db (this is handled with makeFFBDB).

panFFB is simply a version of PAN that has been compiled with the ET system (ONLINE=1 in the codaclass/Makefile and src/Makefile). After compilation, copy the executable to the above **Location**.

3.3 epics_feedback

Language:	bash script
Requires:	bc (CLI calculator), caget, caput
Location:	∼apar/epics/
Started by:	user or panFFB
Control with:	two input arguments
Configuration Files:	N/A
Makes calls to:	EPICS (caget and caput)
Output Files:	N/A
Log Files:	∼apar/feedback/feedback.log
Description:	

Makes changes to source DAC values using EPICS caget and caput. An entry (with date) is also appended to ∼apar/feedback/feedback.log. First argument is the source DAC to control, second is how much to change to current value. For HAPPEX-II, the first argument:

```
1    IGLdac3:ao_7                       IA
2    IGLdac3:ao_5                       PZT X
3    IGLdac3:ao_6                       PZT Y
4    IGLdac2:G2Ch3Pos IGLdac2:G2Ch4Neg  PITA
5    IGLdac3:ao_4                       IA-HallC
```

The version provided was used in HAPPEX-II (2005) to provide:

- IA feedback
- Hall-C IA feedback
- PITA feedback

PZT feedback was tested in previous years (2002-2003), but has not been tested since.

3.4 makeFFBDB

Language:	perl script
Requires:	perl libs:FindBin, TaFileName
Location:	~apar/feedback/
Started by:	startAnalyzer
Control with:	N/A
Configuration Files:	~apar/feedback/striplist.txt
	~apar/feedback/panFFB.db
	~apar/db/parity$yr_$run.db
Makes calls to:	N/A
Output Files:	~apar/db/control.db
	~apar/feedback/runDB/control.db_$run (copy)
Log Files:	N/A

Description:

Modifies the PAN database file, created by createDB, to only include the parameters and devices needed for parity feedback. striplist.txt contains items to copy exactly to control.db. panFFB.db contains the exact lines to append to control.db. See examples for striplist.txt and panFFB.db.

3.5 makeFFBDB_IHWP

Language:	perl script
Requires:	perl libs:FindBin, TaFileName
Location:	~apar/feedback/
Started by:	startAnalyzer
Control with:	one argument: <IHWP State>
Configuration Files:	~apar/feedback/striplist.txt
	~apar/feedback/panFFB.db_IN
	~apar/feedback/panFFB.db_OUT
	~apar/db/parity$yr_$run.db
Makes calls to:	
Output Files:	~apar/db/control.db
	~apar/feedback/runDB/control.db_$run (copy)
Log Files:	N/A

Description:

Same as makeFFBDB, but uses the IHWP state (IN or OUT) as an argument. This argument only effects with panFFB.db_{IN,OUT} to use. See examples for striplist.txt and panFFB.db_{IN,OUT}.

Language:	(1) bash script (flipper) (2) tcl/tk script (flipper.tcl)
Requires:	tcl/tk lib: BLT EPICS extension: et_wish
Location:	~apar/bryan/birdfeed/with_ca/
Started by:	user
Control with:	N/A
Configuration Files:	~apar/feedback/IHWP.{IN,OUT}
Makes calls to:	EPICS (directly with et_wish)
Output Files:	N/A
Log Files:	N/A

Description:

Set of scripts to help automate changes to source settings *after* the IHWP is inserted or extracted. Bash script simply sets up environment variables for the tcl/tk script (including an option to increment the Slug Number). tcl/tk script looks at the current IHWP setting, reads the appropriate IHWP.{IN,OUT}, and makes the changes to the source elements.

4 PANGUIN

PANGUIN is basically an implementation of the onlineGUI that updates its plots based on the updates made to a ROOTfile by a backend process.

4.1 panguin

Language:	bash scripts
Requires:	N/A
Location:	~apar/bin/
Started by:	user
Control with:	one argument: on/off
Configuration Files:	N/A
Makes calls to:	N/A
Output Files:	~apar/bryan/panguin/pan/panguin_enable.dat
Log Files:	N/A

Description:

This command effectively turns enables/disables panguin before the start of a CODA run. It simply puts a 1 (feedback on) or 0 (feedback off) into ~apar/bryan/panguin/pan/panguin_enable.dat.

4.2 pan_online

Language:	bash script
Requires:	panguin_analyzer
Location:	~apar/bryan/panguin/pan/
Started by:	startAnalyzer
Control with:	panguin
Configuration Files:	N/A
Makes calls to:	panguin_analyzer
	getrunnumber
Output Files:	~apar/bryan/panguin/pan/ROOTfiles/parity06_$run_standard.root
	~apar/HAPPEX/pan/pan.root (softlink)
Log Files:	~apar/bryan/panguin/pan/output/out_$run.txt

Description:

Script to handle the execution of panguin_analyzer and update the softlink of the ROOTfile to pan.root. Also removed older ROOTfiles generated by previous processes.

4.3 panguin_analyzer

Language:	C++
Requires:	N/A
Location:	~apar/bryan/panguin/pan/
Started by:	pan_online
Control with:	panguin
Configuration Files:	~apar/db/parity$yr_$run.db
Makes calls to:	N/A
Output Files:	~apar/bryan/panguin/pan/ROOTfiles/parity$yr_$run_standard.root
	~apar/HAPPEX/pan/pan.root
Log Files:	~apar/bryan/panguin/pan/output/out_$run.txt

Description:

Program to analyze data from ET, and store it to a ROOTfile (updating every 100 helicity pairs).

panguin_analyzer is simply a version of PAN that has been compiled with the ET system (ONLINE=1 in the codaclass/Makefile and src/Makefile). After compilation, copy the executable to the above **Location**.

4.4 online.C

Language:	C++
Requires:	ROOT/CINT
Location:	~apar/HAPPEX/pan/panguin
Started by:	user
Control with:	N/A
Configuration Files:	/apar/HAPPEX/pan/panguin/*.cfg
Makes calls to:	N/A
Output Files:	N/A
Log Files:	N/A

Description:

ROOT script to generate a GUI that looks at TTree variables or histograms stored in a ROOTfile. In “watchfile” mode, will continuously update its plots corresponding to the updates made to the ROOTfile. For the HAPPEX-II configuration, it watched pan.root (a softlink to the real ROOTfile generated by panguin_analyzer). Documentation may be found here: http://www.jlab.org/~moffit/onlineGUI/

5 Birdfeed

birdfeed (sometimes referred to as: runbird) is simply a monitor (just watches stuff) for parity feeedback.

5.1 runbird

Language:	bash script
Requires:	N/A
Location:	∼apar/bryan/birdfeed/with_ca/
Started by:	user
Control with:	N/A
Configuration Files:	N/A
Makes calls to:	∼apar/bryan/birdfeed/with_ca/birdfeed
Output Files:	N/A
Log Files:	N/A

Description:

Script to set up enviroment variables and execute the birdfeed.tcl script.

5.2 birdfeed.tcl

Language:	tcl/tk
Requires:	tcl/tk libs: BLT EPICS extension: et_wish
Location:	∼apar/bryan/birdfeed/with_ca/
Started by:	runbird
Control with:	N/A
Configuration Files:	N/A
Makes calls to:	EPICS (directly with et_wish) getpanFFB_asym chkfeedback
Output Files:	N/A
Log Files:	N/A

Description:

Script to display a GUI and monitor EPICS variables and various log outputs generated by panFFB.

5.3 getpanFFB_asym

Language:	perl script
Requires:	N/A
Location:	∼apar/bryan/birdfeed/with_ca/
Started by:	birdfeed.tcl
Control with:	two arguments: $runnumber $device
Configuration Files:	N/A
Makes calls to:	N/A
Output Files:	N/A
Log Files:	N/A

Description:

Script to scan a panFFB run log (/adaqfs/home/apar/feedback/runlog/ffb_$runnumber.log) for the measured beam asymmetries.

5.4 chkfeedback

Language:	perl script
Requires:	N/A
Location:	∼apar/bryan/birdfeed/with_ca/
Started by:	birdfeed.tcl
Control with:	N/A
Configuration Files:	N/A
Makes calls to:	N/A
Output Files:	N/A
Log Files:	N/A

Description:

Script that checks the status of HAPPEX feedback (looking for it's process ID in the process list, and checking the timestamp on the current panFFB log).