TestDAQ High-Level Users' Guide

Jim Braun

June 17, 2004

1 Introduction

The following document describes the proper use of higher-level Icecube DAQ components including stfapp, testdaq-collector, testdaq-control, and testdaq-lightsource. Emphasis is placed on the use of testdaq components to produce raw data, assuming proper hardware and software setup. Setup of DOMs and DOMhub computers is not within the scope of this document. DOM setup is discussed in depth by Kael Hanson, [1] DOMhub software setup is described by John Jacobsen and Mark Krasberg, [2] and David Hays is the authority on DOMhub application. [3]

2 Stfapp

STF is a framework for testing DOM mainboards and integrated DOMs where pass/fail criteria may be applied, described in detail by Chuck McParland. [4] Stfapp is the client-side application needed to consolidate communication with stfserv over many DOMs, display test results, and store results in the production database.

2.1 Using the Stfapp GUI

To start the stfapp GUI, type

```
java -cp {BFD_HOME}/lib/stfapp.jar icecube.daq.stf.STF
```

at the command prompt. The stfapp command window will pop up. Next, the user must open a DOM testing session. Stfapp is able to connect to DOMs over a terminal server or through domhub application, and stfapp provides its own implementation to communicate with DOMs over the DOR interface.

2.1.1 Connecting to DOMs Using a Terminal Server

On the stfapp main window, choose "Connect to Terminal Server" under the "Connect" menu to pop up the terminal server connection dialog. The user must specify the IP address of the terminal server, as well as the lowest port number (base number) and the number of sequential port numbers above the base number to scan. Stfapp will detect any DOMs connected to ports within this range and ignore unconnected ports. It may be necessary to reboot the terminal server before connecting with some terminal server models.

2.1.2 Connecting to DOMs Through DOM Hub Application

First, Make sure domhub application is running and bound to the rmiregistry on the domhub computer. On the stfapp main window, choose "Connect to DOMHub" under the "Connect" menu to pop up the domhub connection dialog. The user must specify the hostname or IP address of the domhub and click "OK". Stfapp issues a request to domhub application to power unpowered channels, and stfapp will bind all DOMs domhub application lists as available.

2.1.3 Connecting to DOMs Through the Stfapp DOR Interface

This option requires stfapp to run on the domhub computer. On the stfapp main window, choose "Open direct DOR session" under the "Connect" menu to begin probing DOR cards attached to the local computer. Stfapp will enable power on unpowered channels and softboot all detected DOMs.

2.1.4 The DOM Panel

The leftmost panel contains identifiers for each DOM detected by stfapp. During mainboard testing, the identifier is the mainboard serial issued at LBNL. During DOM testing, the identifier is the DOM production ID. These identifiers are retrieved from the STF database. When first tested, a mainboard's identifier is the DOM ID, and the user is prompted to enter the proper mainboard serial. Likewise, when an assembled DOM is first tested, the DOM is identified by the mainboard serial, and the user is prompted to enter the production ID of the DOM. The user must select which DOMs to test by highlighting them in the DOM panel or by choosing "Select All DOMs" from the "Test" menu.

2.1.5 Loading Tests

On the stfapp main window, choose "Find STF Tests" from the "File" menu to pop up a file chooser dialog. The user may choose an individual stf setup file or a directory containing many setup files. After selecting, stfapp parses the tests, loads them, and displays them in the test panel.

2.1.6 The Test Panel

The rightmost panel contains names of tests currently loaded by stfapp. The user must select which tests to run by highlighting them in the test panel or by choosing "Select All Tests" in the "Test" menu. Tests loaded in stfapp are modifiable by right-clicking on the name of the test within the test panel, which will pop up the test modification dialog. The dialog displays all user modifiable test parameters and their current values. To make a change, the user can simply change a parameter value and click "OK". Changes are reflected on the original file on disk as well as in memory.

2.1.7 Running Tests

To run tests, highlight the appropriate DOMs and tests. Choose "Start Test Run" from the "Test" menu to begin the test sequence. Stfapp prepares the DOMs for the test sequence and asks how many iterations of each test to perform. (e.g. If the user sets this value to 2, then each highlighted test will run twice, etc.) Stfapp will ask for the current temperature of the DOMs, and finally whether the DOMs are integrated. (i.e. Are the DOMs just mainboards, or have they been integrated into full DOMs?) Stfapp now begins testing, and the test result panel appears in the foreground.

2.1.8 The Test Result Panel

The test result panel displays results as a grid of red and green dots, where green indicates a test is passed, and red indicates failure. The result display consists of a master, or "root" panel, which displays

a summary grid of results with DOMs on the vertical axis and tests on the horizontal axis, as well as a detailed panel for each test. The detailed panel corresponds to a specific test highlighted in the Test Panel, and contains horizontal entries for each iteration performed. Grid dots indicate the successfulness of each iteration of the test for each DOM. Double-clicking on a grid dot in this panel pops up a test result panel, displaying the test ouput and graphing any array data contained within the output. The root panel grid dot summarizes the results contained in the detailed panel: A green dot indicates all iterations of the test passed, whereas a red dot indicates at least one iteration of the test failed. Double-clicking on a column in the root panel opens the appropriate detailed panel.

2.1.9 The Information Panel

Any test failures are appended to a text area located beneath the DOM panel and test panel on the stfapp main window.

2.2 Using Stfapp in Command Line Mode

In command line mode, stfapp offers the same features. To connect to DOMs using domhub application, type

```
java -cp {BFD_HOME}/lib/stfapp.jar icecube.daq.stf.STF [directory] [isDomTest] [#iterations]
[temperature] [domhub IP address]
```

where directory is the path of the directory containing stf setup files, isDomTest is true/false indicating whether the DOMs to be tested are integrated, #iterations is the number of times each loded test will be run, and temperature is the current temperature of the DOMs. Similarly, a user can connect to a terminal server using

```
java -cp {BFD_HOME}/lib/stfapp.jar icecube.daq.stf.STF [directory] [isDomTest] [#iterations]
[temperature] [terminal server IP address] [base port] [number of ports]
```

and could communicate directly with the driver using

```
java -cp {BFD_HOME}/lib/stfapp.jar icecube.daq.stf.STF [directory] [isDomTest] [#iterations]
[temperature]
```

3 Testdaq-collector

Testdaq-collector receives hit, monitor, and time calibration data over TCP/IP from domhubapp, applies time calibration to hit and monitor records and appends this data in a header, and consolidates all run data into a ZIP archive. Table 1 describes the header applied to hit and monitor records, while Table 2 contains information about the tcalib record reader. Testdaq-collector also processes the external monitoring data generated by testdaq-lightsource and testdaq-control. Table 3 describes the format of the external monitor file written by testdaq-collector.

Quantity	Datum Size	Comments
Wrapped Record Length	Int (32 bits)	Length of hit record + wrapper
Format ID	Int (32 bits)	Current hit format is 1, current monitor format is 101
DOM ID	Long (64 bits)	Binary, not character, representation of DOM ID
Reserved Space	Long (64 bits)	Zeroes, reserved for future fields
Calibrated Time	Long (64 bits)	Rapcal corrected DOR time of event

Table 1: Testdaq-collector wrapper for hit and monitor records

Quantity	Datum Size	Comments
DOM ID	Long (64 bits)	Binary, not character, representation of DOM ID

Table 2: Testdaq-collector wrapper for time calibration records

Quantity	Datum Size	Comments
Record Length	Short (16 bits)	Length of event
Format ID	Short (16 bits)	Current format ID is 301
Timestamp	Long (64 bits)	Approximate DOR time corresponding to record
Pulser Mode	Int (32 bits)	Valid values are 0-9
Pulser Frequency	Int (32 bits)	Frequency of pulser in Hz
Pulse Width	Int (32 bits)	Width in nanoseconds
Pulse Amplitude	Int (32 bits)	Amplitude DAC setting
Filter Wheel Position	Int (32 bits)	
Filter Wheel ID	Int (32 bits)	ID of selected filter
Monochromator Wavelength	Int (32 bits)	Wavelength setting of monochromator (nm)

Table 3: External monitor record format

Entry	Comment
ziparchive.hit	Wrapped hit data
ziparchive.mon	Wrapped monitor data
ziparchive.tcal	Wrapped time calibration data
ziparchive.extern	External monitor data
ziparchive.xml	Run header as defined through testdaq-io

Table 4: Testdaq-collector ZIP archive structure for "ziparchive107234954359.zip"

3.1 The ZIP Archive

Testdaq-collector compresses all data into a zip archive for ease of storage. The zip file is systematically named using several parts. The first portion is a comment entered at run time by the operator. Next is the system time in milliseconds when the zip was created. All archives produced end in the ".zip" extension. The zip file contains five entries as described in table 4. To decompress "ziparchive107234954359.zip" at the command line, use

```
unzip ziparchive107234954359.zip
```

3.2 Starting Testdaq-collector

To start testdaq-collector and bind datacollector to the RMI registry, first be sure rmiregistry is running. If not, start it:

```
rmiregistry&
```

Start testdaq-collector:

```
java -cp {BFD_HOME}/lib/testdaq-collector.jar icecube.testdaq.datacollector.
DataCollectorFactoryImpl
```

At this point, testdaq-collector is prepared to accept data from domhub application.

4 Testdaq-lightsource

Testdaq-lightsource consists of three RMI servers which control the external light necessary for several DOM production tests. [5]

4.1 Pulser

The external pulser creates pulses of light of known width and amplitude which are carried to DOMs via a network of optical fibers. Characteristics of the pulse are controlled by the mode setting passed from testdaq-control. Table 5 describes valid modes.

Value	Comment
0	All off
1	Marker pulse
2	Narrow pulse
3	Big pulse
4	Sync output
5	Relay
6	Marker + narrow pulse
7	Marker + big pulse
8	Narrow + big pulse
9	Marker + narrow + big pulse

Table 5: Valid pulser modes.

4.1.1 Starting the Pulser Server

With rmiregistry running, issue the command

```
java -cp {BFD_HOME}/lib/testdaq-lightsource.jar icecube.testdaq.lightsource.PulserServer
[hostname] [port] [pulse width] [pulse amplitude]
```

where "hostname" is the address of the terminal server connected to the pulser and "port" is the port number to which the pulser is connected. The pulse width and amplitude are currently set internally by the pulser and cannot currently be read. The width and amplitude arguments do not change the characteristics of the pulse. They merely report the current hardware state to testdaq-control.

4.2 Monochromator and Filter Wheel

The monochromator and filter wheel servers can be started in a manner similar to the pulser server. To start the monochromator, be sure rmiregistry is running and issue the command

```
java -cp {BFD_HOME}/lib/testdaq-lightsource.jar icecube.testdaq.lightsource.MonochromatorImp
[hostname] [port]
```

where "hostname" is the address of the terminal server connected to the pulser and "port" is the port number to which the pulser is connected. To start the filter wheel server, issue the command

```
java -cp {BFD_HOME}/lib/testdaq-lightsource.jar icecube.testdaq.lightsource.FilterWheelImp
[hostname] [port]
```

5 Testdaq-control

TestDAQ is a scaled-down version of the IceCube DAQ designed for DOM production testing. Testdaq-control is the testDAQ user interface. Roles of testdaq-control include requesting DOM data streams from domhub application, connecting testdaq-collector to DOM data streams, and many configuration tasks.

5.1 Using the Testdaq-control GUI

To start the GUI, type

```
java -cp {BFD_HOME}/lib/testdaq-control.jar icecube.testdaq.control.TestDAQControl
```

5.1.1 Connecting to Domhub Application

To connect to a local domhub and detect its DOMs, choose "Connect to DOM Hub" from the "Connect" menu in the testdaq-control main window. Testdaq-control then requests domhub application to power any unpowered pairs and softboot any DOMs it detects.

5.1.2 Connecting to Testdaq-collector

To connect to a testdaq-collector instance, choose "Connect to DataCollector" from the "Connect" menu in the testdaq-control main window. Alternatively, testdaq-control has the capability to start a testdaq-collector instance and bind it to the rmiregistry. To do this, choose "Start DataCollector" from the "Control" menu in the testdaq-control main window.

5.1.3 Connecting to Pulser, Monochromator, and Filter Wheel Servers

The user can connect to these servers with the proper command under the "Connect" menu. Connections to these servers must be in place if the run description XML file requests a change (e.g. changing pulser frequency from 100Hz to 1000Hz). Make sure the servers are running and bound to the RMI registry.

5.1.4 Loading a Run Description File

TestDAQ configuration information for a series of tests is contained in an XML formatted file, which is described in detail in section 5.3 on page 9. To load a run description file, choose "Load Tests" from the "File" menu in the testdaq-control main window. Select the desired file using the file chooser pop up dialog and click "OK". Testdaq-control parses the tests from the file and displays them in the test panel.

5.1.5 The Test and Status Panels

The test panel displays information about the tests currently loaded by testdaq-control, including the name and minimum duration of the test. Also, if testing is in progress, the current test is highlighted.

The status panel, at the very bottom of the testdaq-control window, contains basic information regarding the state of testdaq-control, including domhub and datacollector connection state, elapsed time, and total estimated completion time of the current test sequence.

5.1.6 Starting Tests

To begin testing, choose "Start Test Sequence" from the "Control" menu in the testdaq-control main window. Testdaq-control then starts domapp on each DOM and prompts the user for a location to store the output zip file. Enter the full path to your data directory, without a following slash. Testdaq-control is

Parameter	Comment
NUM_SAMPLES	Number of bins in ATWD data by channel
SAMPLE_SIZE	Number of bytes per ATWD bin by channel
NUM_FADC_SAMPLES	Number of bins in FADC data
HW_INTERVAL	Interval in seconds between hardware monitor events
CONF_INTERVAL	Interval in seconds between configuration monitor events
PULSER_MODE	External pulser mode, as in Table 5
PULSER_FREQUENCY	Frequency of pulser, in Hz
MONOCHROMATOR_WAVELENGTH	Wavelength of monochromator output, in nm
WHEEL_POSITION	Position of filter wheel

Table 6: Global parameters of "Pulser Test - Full Samples (1550b)".

relatively intolerant of erroneous input. Testdaq-control then prompts for a run description, which testdaq-collector uses in the name of the zip archive. Testdaq-control now begins taking data and sequencing through loaded tests.

5.2 Using Testdaq-control in Command Line Mode

Testdaq-control command line mode is much easier to use and is fully automated. To start a run, use

```
java -cp {BFD_HOME}/lib/testdaq-control.jar icecube.testdaq.control.TestDAQControlComandLine
[test description file] [output location] [run description] [-d domhub hostname]
[-c data collector hostname] [-p pulser hostname] [-m monochromator hostname]
[-f filter wheel hostname]
```

The test description file, output location, run description, and at least one domhub are required. Not specifying a data collector causes testdaq-control to start one on the local machine. Other arguments are optional.

5.3 The Run Description XML File

The run description file contains all configuration information for a sequence of tests. The description file in figure 1 is a straightforward example. This file contains one test named Pulser Test - Full Samples (1550b) which will run for 100 seconds.

5.3.1 Global Test Parameters

Global parameters include ATWD data format configuration, monitor record rate, and pulser configuration. Table 6 examines the test "Pulser Test - Full Samples (1550b)".

```
<?xml version="1.0"?>
<DOM_Test_Configuration>
        <test>
                <testName name="Pulser Test - Full Samples ( 1550b )"/>
                <executionTime value="100"/>
                <param name="NUM_SAMPLES" atwdChannel="0" value="128"/>
                <param name="NUM_SAMPLES" atwdChannel="1" value="128"/>
                <param name="NUM_SAMPLES" atwdChannel="2" value="128"/>
                <param name="NUM_SAMPLES" atwdChannel="3" value="128"/>
                <param name="SAMPLE_SIZE" atwdChannel="0" value="2"/>
                <param name="SAMPLE_SIZE" atwdChannel="1" value="2"/>
                <param name="SAMPLE_SIZE" atwdChannel="2" value="2"/>
                <param name="SAMPLE_SIZE" atwdChannel="3" value="2"/>
                <param name="NUM_FADC_SAMPLES" value="255"/>
                <param name="HW_INTERVAL"  value="5"/>
                <param name="CONF_INTERVAL" value="10"/>
                <param name="PULSER_MODE" value="1"/>
                <param name="PULSER_FREQUENCY" value="100"/>
                <param name="MONOCHROMATOR_WAVELENGTH" value="400"/>
                <param name="WHEEL_POSITION" value="0"/>
                <domConfiguration type="non-specific">
                        <dom>
                                <param name="DAC_LED_BRIGHTNESS" value="0"/>
                                <param name="DAC_ATWD_ANALOG_REF" value="2048"/>
                                <param name="DAC_INTERNAL_PULSER" value="1000"/>
                                <param name="DAC_FAST_ADC_REF" value="700"/>
                                <param name="DAC_SINGLE_SPE_THRESH" value="520"/>
                                <param name="DAC_ATWD1_RAMP_RATE" value="3000"/>
                                <param name="TRIG_MODE" value="2"/>
                                <param name="DAC_ATWD0_RAMP_RATE" value="3000"/>
                                <param name="ATWD_SELECT" value="0"/>
                                <param name="DAC_FE_AMP_LOWER_CLAMP" value="800"/>
                                <param name="ANALOG_MUX_SELECT" value="1"/>
                                <param name="DAC_PMT_FE_PEDESTAL" value="1925"/>
                                <param name="DAC_ATWD1_RAMP_TOP" value="2097"/>
                                <param name="DAC_ATWD1_TRIGGER_BIAS" value="850"/>
                                <param name="FE_PULSER_RATE" value="1"/>
                                <param name="DAC_MULTIPLE_SPE_THRESH" value="600"/>
                                <param name="DAC_ATWD0_RAMP_TOP" value="2097"/>
                                <param name="DAC_ATWD0_TRIGGER_BIAS" value="850"/>
                                <param name="PMT_HV_LIMIT" value="0"/>
                                <param name="PMT_HV_DAC" value="0"/>
                        </dom>
                </domConfiguration>
        </test>
</DOM_Test_Configuration>
```

Figure 1: Sample run description file

Parameter	Comment
DAC_ATWD0_RAMP_RATE	ATWD0 ramp rate
DAC_ATWD0_RAMP_TOP	ATWD0 upper ramp limit
DAC_ATWD0_TRIGGER_BIAS	ATWD0 trigger bias
DAC_ATWD_ANALOG_REF	Analog voltage reference
DAC_ATWD1_RAMP_RATE	ATWD1 ramp rate
DAC_ATWD1_RAMP_TOP	ATWD1 upper ramp limit
DAC_ATWD1_TRIGGER_BIAS	ATWD1 trigger bias
DAC_LED_BRIGHTNESS	on-board LED brightness control
DAC_INTERNAL_PULSER	Internal pulser amplitude
DAC_FAST_ADC_REF	Fast ADC reference (pedestal shift)
DAC_SINGLE_SPE_THRESH	Single SPE discriminator threshold
TRIG_MODE	Trigger mode selection, described in table 8
ATWD_SELECT	ATWD 0 or 1
DAC_FE_AMP_LOWER_CLAMP	front end amp lower clamp voltage – CURRENTLY UNUSED
ANALOG_MUX_SELECT	Analog mux channel selection
DAC_PMT_FE_PEDESTAL	PMT front end pedestal
FE_PULSER_RATE	Internal pulser rate
DAC_MULTIPLE_SPE_THRESH	Multiple SPE discriminator threshold
PMT_HV_LIMIT	Max allowable value of HV DAC
PMT_HV_DAC	Value of HV DAC

Table 7: DOM-specific parameters of "Pulser Test - Full Samples (1550b)".

5.3.2 DOM Specific Parameters

Every test must contain at least one dom element, which contains parameters which may vary between DOMs. Again examining "Pulser Test - Full Samples (1550b)", The line

```
<domConfiguration type="non-specific">
```

indicates the dom configuration supplied is valid for all discovered DOMs. If the line was

```
<domConfiguration type="specific">
```

a separate dom element must be present for each DOM in the run, and the dom element tag indicates the DOM ID:

```
<dom domId="00173d4d442a">
```

Table 7 describes other DOM-specific parameters, mostly taken from Arthur Jones. [6]

5.4 Analysis Tools Contained Within Testdaq-control

Testdaq-control contains two simple but useful tools to analyze data produced by TestDAQ. EZPlot is a simple graphing utility capable of reading both wrapped and unwrapped hit records, and Coincidence searches data for multiple DOM coincidences given a time window.

Value	Comment
0	Test pattern
1	CPU requested trigger
2	SPE trigger

Table 8: Description of TRIG_MODE parameter

5.4.1 EZPlot

To launch EZPlot, use

```
java -cp {BFD_HOME}/lib/testdaq-control.jar icecube.testdaq.ezplot.Plotter
```

EZPlot can ingest engineering hit records in three ways. EZPlot can read zip archives by choosing "Load Zip File" from the "File" menu or read an unzipped hit file by choosing "Load Hit File" from the "File" menu. Also, EZPlot can read engineering format records directly from a socket served by domserv, domhub application, etc. by choosing the option "Read from Socket" To start displaying data, choose "Start" from the "Control" menu. The user can increase or decrease speed by changing the display rate, which can be set using "Set Plotter Rate" from the "Control" menu. Choosing "Stop" from the "Control" menu causes EZPlot to pause on the current record until the "Start" command is given. A new file may be loaded at any time. DOM ID, rapcal corrected UTC time, and DOM clock ticks are displayed at the bottom for each record. A blue bar at the bottom indicates the status of the EZPlot buffer. If the buffer is continually empty, records are being displayed faster than EZPlot cat read them, which occasionally occurs while reading from a socket.

5.4.2 Coincidence

To start a coincidence scan of a zip archive, issue the command

```
java -cp {BFD_HOME}/lib/testdaq-control.jar icecube.testdaq.coincidence.Coincidence [zip archive]
[time window] [number of DOMs]
```

Coincidence will search the zip archive for "number of DOMs" hits occuring within "time window" picoseconds and report information including DOM ID, UTC time of event, and DOM clock. A minimum of 20MB memory is required for each DOM in the run (i.e. a zip file containing data from 10 DOMs requires a minimum of 200MB memory).

References

[1] Hanson, K "IceCube DOM Users' Guide"
http://docushare.icecube.wisc.edu/docushare/dsweb/Get/Document-4071/DOM+User+Guide.pdf

[2] Jacobsen, J and M Krasberg "Installing DOM Hub Firmware and Software"
http://docushare.icecube.wisc.edu/docushare/dsweb/Get/Document-6741/dom_hub_configuration_document_2_4.doc

[3] Hays, D "DOM Hub Application Developers Guide"
http://docushare.icecube.wisc.edu/docushare/dsweb/Get/Document-7264/DHAppDevGuide.doc

[4] McParland, C "Simple Test Framework (STF) Test Plan and Environment"
http://docushare.icecube.wisc.edu/docushare/dsweb/Get/Document-3132/STFtestEnvironment.pdf

[5] Wendt, C "Illumination for DFL Production Tests"
http://docushare.icecube.wisc.edu/docushare/dsweb/View/Collection-954

[6] Jones, A "DOM_MB_pld.h File Reference" http://deimos.lbl.gov/ arthur/dom-mb/DOM_MB_pld_8h.html