CCLDAS Database System Manual

by Spenser Burrows

Introduction

In 2012, the CCLDAS accelerator facility deployed an integrated database system that automatically saves and analyzes waveforms acquired by the beamline detectors and records various accelerator settings. This system greatly streamlines data analysis for users inside CCLDAS and customers outside our group by allowing us to provide datasets with high-quality information about the characteristics of the dust particles that reach the target chamber.

When the PSU selects a dust particle, a National Instruments LabVIEW program records waveforms from each of the three beamline detectors, the time that the dust particle was observed, and makes rough estimates for mass, velocity, and charge. These waveforms are saved as binary files and a unique entry is made in an SQL database with timestamp and dust information. The dust particles are then continuously examined by a batch processing VI that runs new waveforms through an IDL data analysis program. This program first rejects false triggers and dust particles that failed to traverse the entire beamline, then provides high quality values for the mass, charge and velocity of the remaining dust particles. The SQL database is then updated with this information, which can be retrieved using a LabView VI that generates live mass versus velocity plots, .csv format metadata files, and/or .HDF5 format combined waveform and metadata files using data selected by experimental information, time, mass, and velocity constraints supplied by the user.

Further engineering data, such as information about the state of the pelletron, control settings, and the location of the dust beam are also stored in the database to allow us to characterize the performance of our accelerator and help us to determine ways in which we can improve that performance.

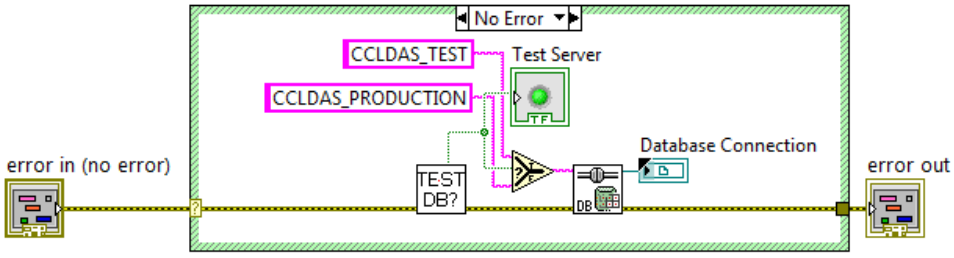
The database system’s automation of functions that would otherwise have to be performed manually reduces the workload of the research science staff at CCLDAS, and allows their time to be spent in more valuable ways. The system’s near-real time feedback to the operator about the experiment being run increases the efficiency of our use of the accelerator. It allows us to get more work done with the same amount of time and funding.

Commonly used LabVIEW database toolkit code structures

Connection VIs:

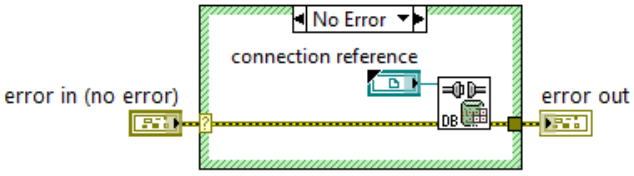
While the many various subvis that make up the database system differ greatly and carry out various functions, there are several pieces of database toolkit code that appear repeatedly.

In order to post, retrieve, or update information in the database a connection from labview to the database must be established. This is done using a vi named **connect in.vi**. This vi simply establishes a connection to the appropriate database (either the test database used for debugging or the production database, determined by a setting in a constants sub vi) and passes a connection reference out.



This connection can then be used by other database vis to communicate with the database.

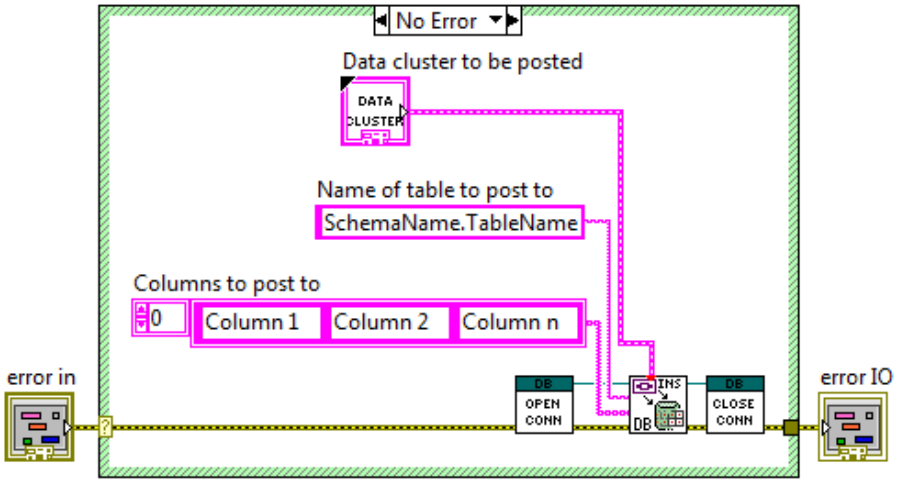
Once the connection is no longer used, it should be severed using a sub vi named **connect out.vi**. This vi has an input for a connection reference generated by **connect in.vi** and closes it when executed.



Generic Database Posting Code:

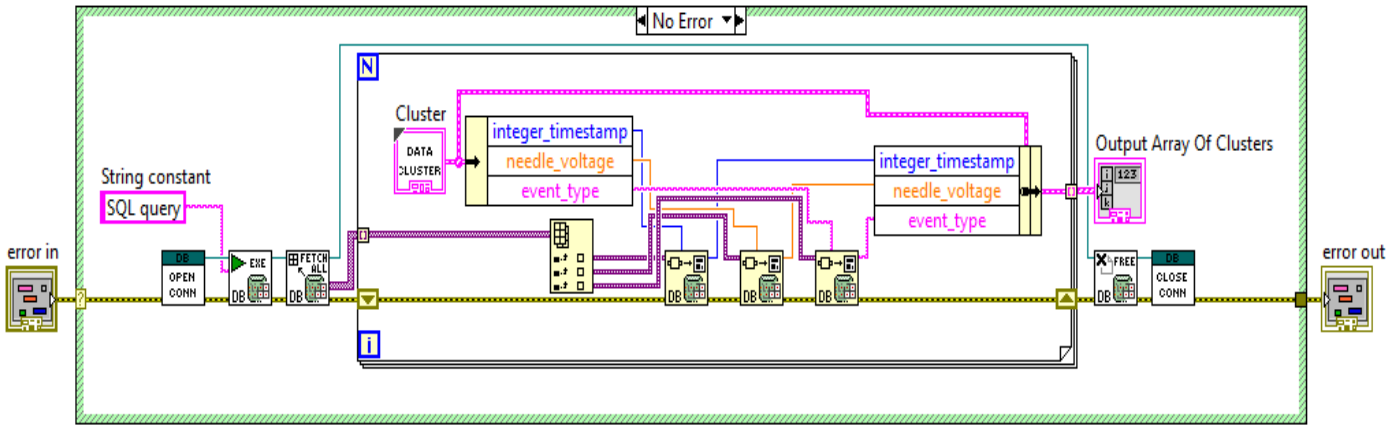
In order to post information to and create a new entry in the database, a database

toolkit subvi named **DB Tools Insert Data VI** is used. This vi is wired up to connection opening and closing vis, a string constant that defines which table in the SQL database to post to, an optional string array that defines which columns in that table to post to, and either a data cluster or a single value to post.



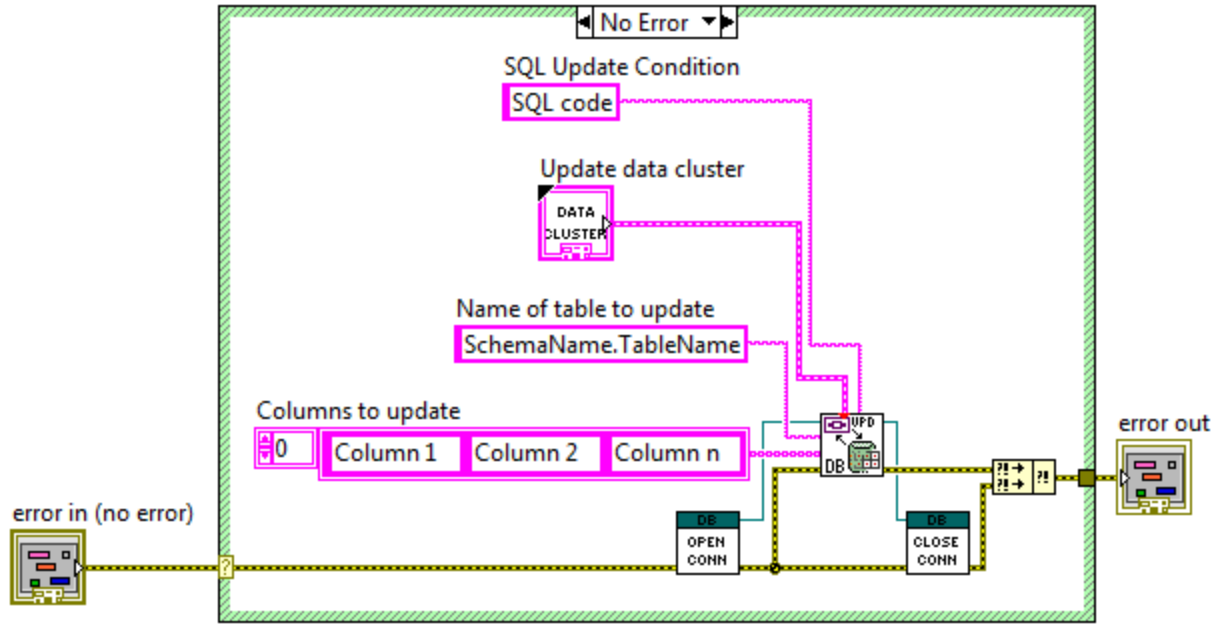
Generic Database Retrieval Code:

In order to retrieve information from the database, the an SQL query (the format for which can be found online or in an SQL textbook) is entered into the **DB Tools Execute Query VI**. This VI creates a recordset reference, which is passed to the **DB Tools Fetch Recordset Data VI** which retrieves a data array. The data array can then be passed to a FOR loop to sort out the different rows, and indexed to get the different columns separated. The columns are then connected to the **Database Variant To Data Function** vi along with a constant to allow the function to determine the datatype of the array element input. The output of the function is the desired data type, which can then be bundled and output as an array of clusters. The **DB Tools Free Object VI** should then be used to free the recordset reference.



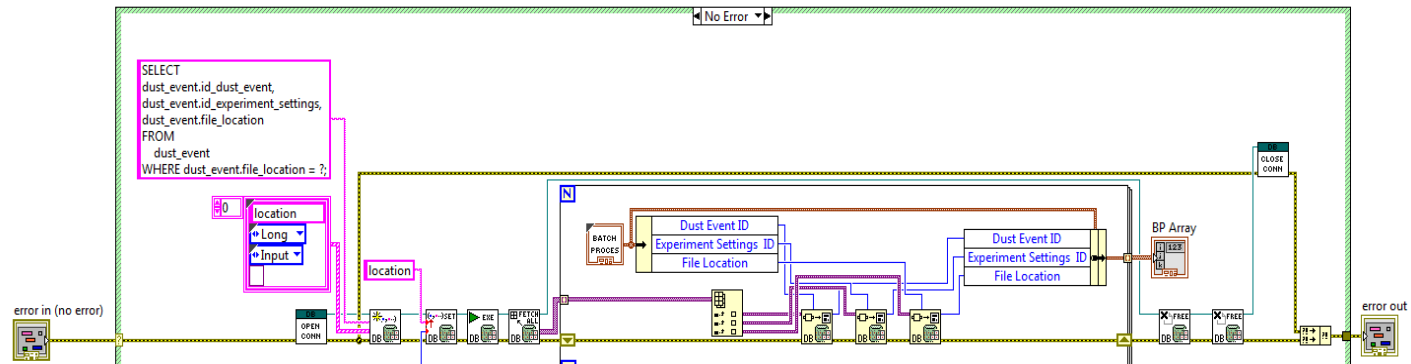
Generic Database Update Code:

In order to update existing database entries with new information the subvi **DB Tools Update Data VI** is used. Like the query vi and input vis, this subvi uses information about the table to update, the columns to update, and an SQL statement to determine what in the to update with a data cluster.



Generic Parameterized Database Retrieval Code:

In order to retrieve information from the database in a way that prevents unintentional SQL injections, a parameterized SQL query (the format for which can be found online or in an SQL textbook) and an array of parameters are entered into the **DB Tools Create Parameterized Query VI**. This VI creates a command reference which is passed to the **DB Tools Set Parameter Value VI,** which uses a name or parameter number and a value to set the the parameter. The command reference is then passed to the **DB Tools Execute Query VI**. This VI creates a recordset reference, which is passed to the **DB Tools Fetch Recordset Data VI** which retrieves a data array. The data array can then be passed to a FOR loop to sort out the different rows, and indexed to get the different columns separated. The columns are then connected to the **Database Variant To Data Function** vi along with a constant to allow the function to determine the datatype of the array element input. The output of the function is the desired data type, which can then be bundled and output as an array of clusters. The **DB Tools Free Object VI** should then be used twice to free the command and recordset references.

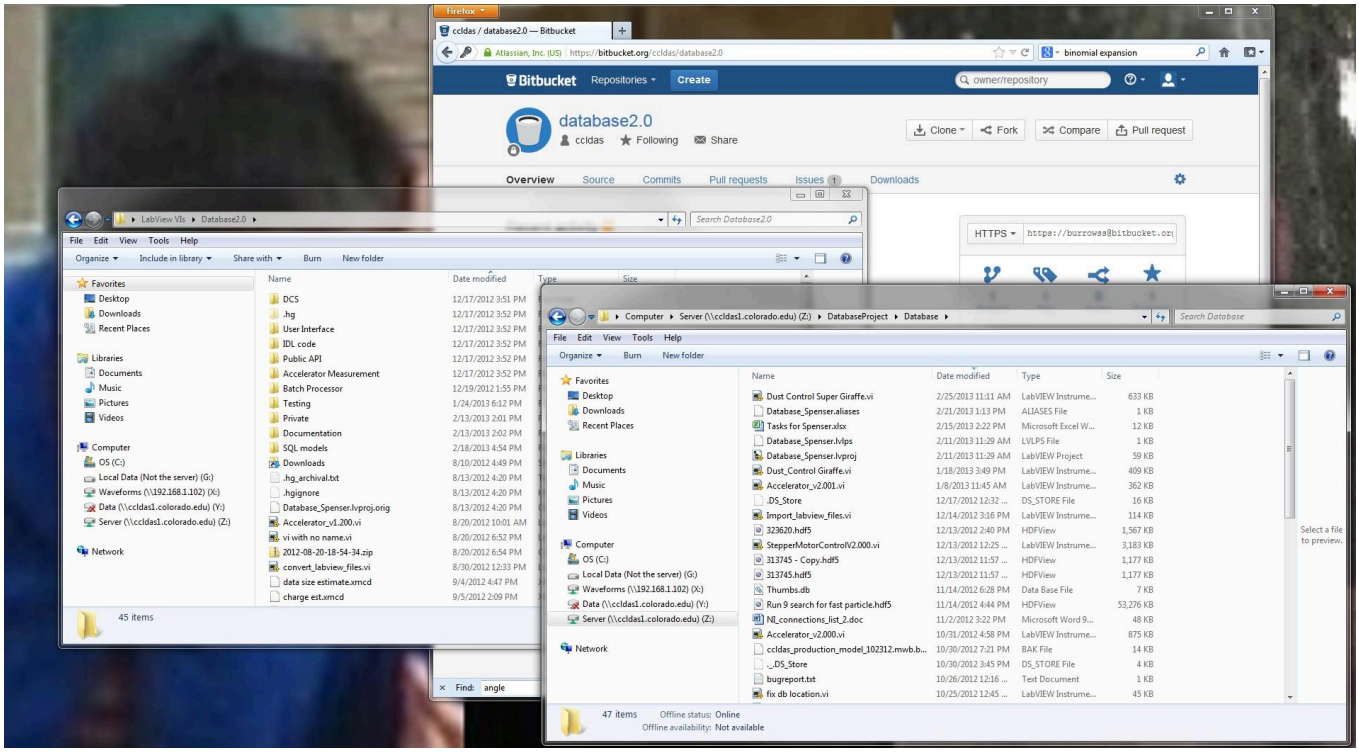


Required Software List

In order for the database programs to be run, a series of software needs to be installed:

1. **National Instruments LabVIEW 2012**
2. **NI LabVIEW Database toolkit**
3. **vipm-windows**
4. **h5labview-0.9.0.1**
5. **My SQL workbench**
6. **My SQL 32 bit ODBC driver (needs to be used to talk to labview and set up with odbcad32.exe rather than the 64 bit version)**
7. **IDL**
8. **HDF5**
9. **Labview device drivers**

File Locations



The files required to run the database are all stored in three different locations.

First, they are on the Clavius computer in the high bay in the **C:\Users\Clavius\Desktop\LabView VIs\Database2.0** folder. This copy is the one actually used during normal operations.

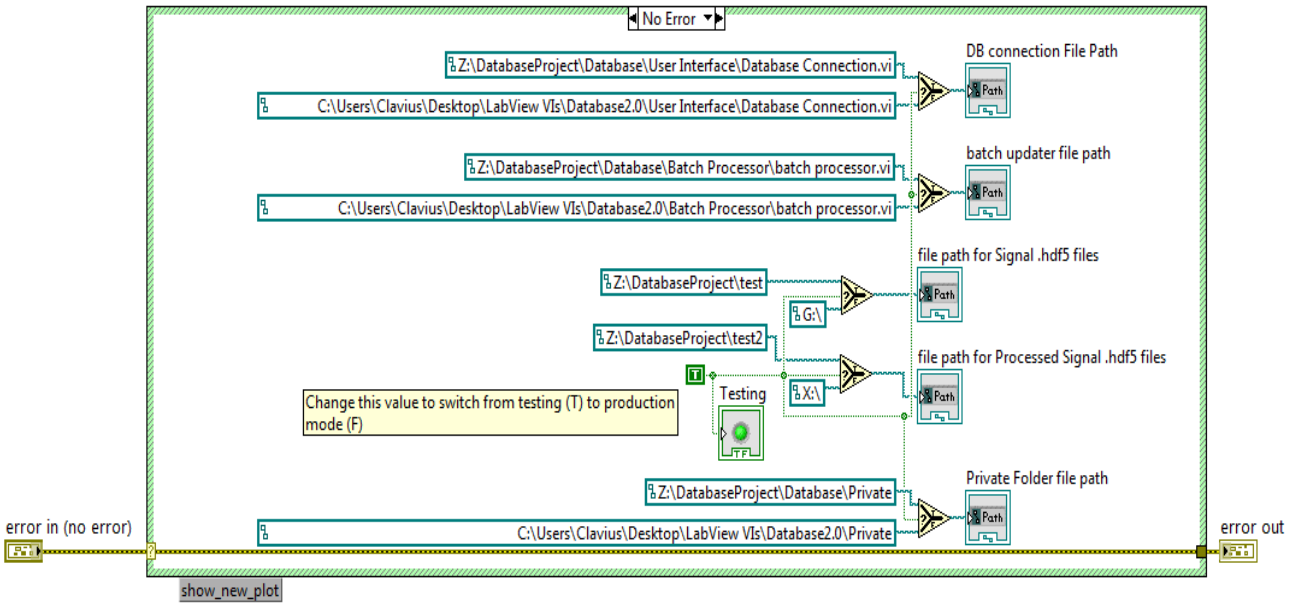
Second, they are on the server in the **Z:\DatabaseProject\Database** folder. This copy is the one normally used for development and testing.

Finally, a copy is stored online at **https://bitbucket.org/ccldas/database2.0.** This is a backup source control copy that has been used to document changes during development. If either of the other two copies is lost, the source control copy can be used to recover the database files.

Deploying new versions/Reverting to old versions

The procedure for deploying a new version to the production database downstairs is as follows:

1. Commit all changes to bitbucket to make sure the source control copy of the database files is up to date
2. Copy the changed files from their location on the server in the **Z:\DatabaseProject\Database** folder to the Clavius computer in the high bay in the **C:\Users\Clavius\Desktop\LabView VIs\Database2.0** folder. This can be done file by file, or by simply replacing the whole folder. **HOWEVER, if the whole folder is replaced, the constants.vi program in the private folder must be updated to reflect that the database is in production mode!**
3. **Constants.vi** can be updated by opening the block diagram and changing the true false constant to **FALSE.** This changes the appropriate values in the database to production values.

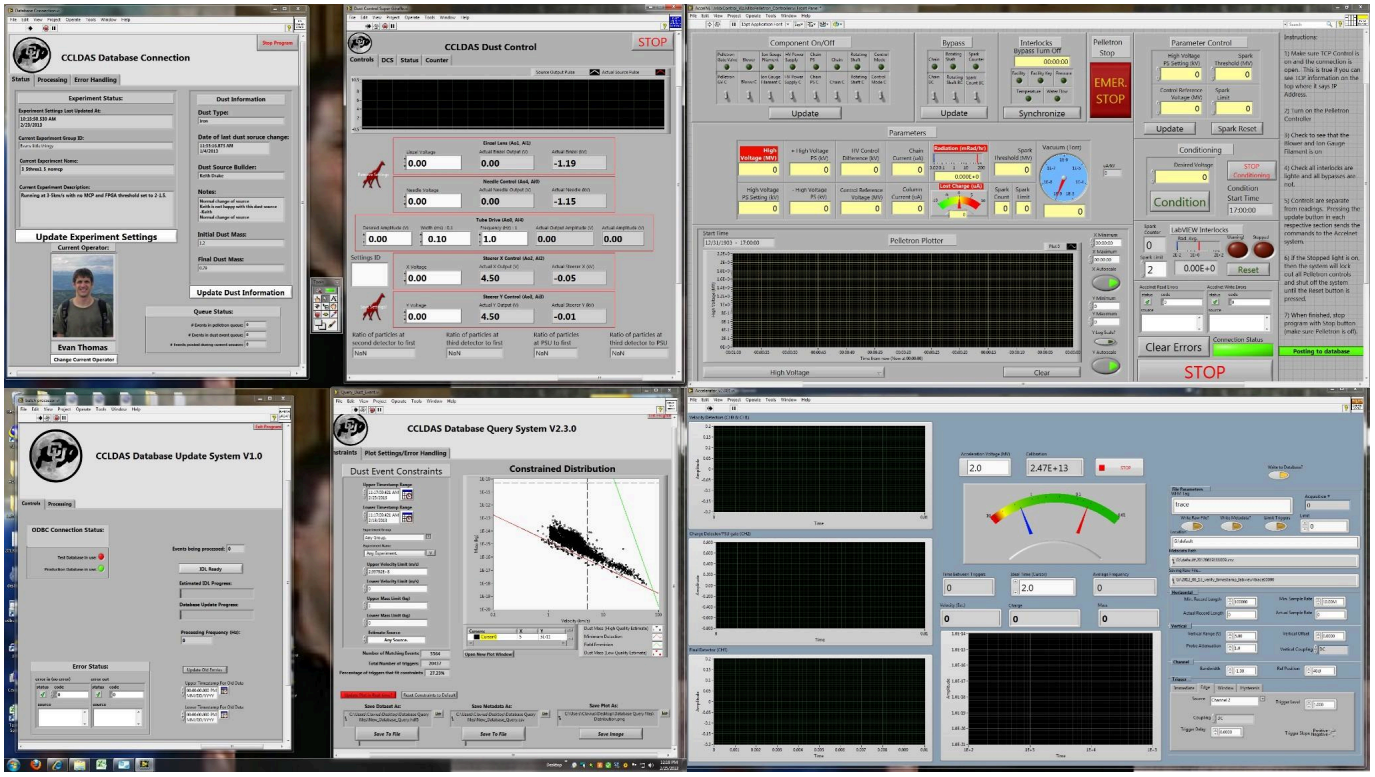


To recover files:

1. Open the Tortoise HG workbench
2. Select a good reversion
3. Select the file to be reverted
4. Right click
5. Select revert to revision.

This will revert the selected file to the proper reversion.

Dataflow Overview



Each time the data acquisition program is triggered, a subvi that enqueues waveforms is called. Dust event waveforms are then dequeued by **database connection.vi,** which creates a database entry, writes binary labview waveform files, then updates the database to show when file writing is complete.

Batch processor.vi then takes files that have been writen and converts them to .hdf5 files. When these .hdf5 waveform files are written, the database is updated to show that they are ready for analysis. **Batch processor.vi** then puts these waveforms through IDL and updates the database with the IDL results. Analyzed waveforms are then moved to the server and the database is updated with their new location.

At every step of the process after the .hdf5 waveform files have been written, **Query_Dust_Event.vi** can plot dust points or retreve constrained datasets in either .hdf5 (incuding metadata and waveforms) or .csv (metadata only) format.

SQL model overview

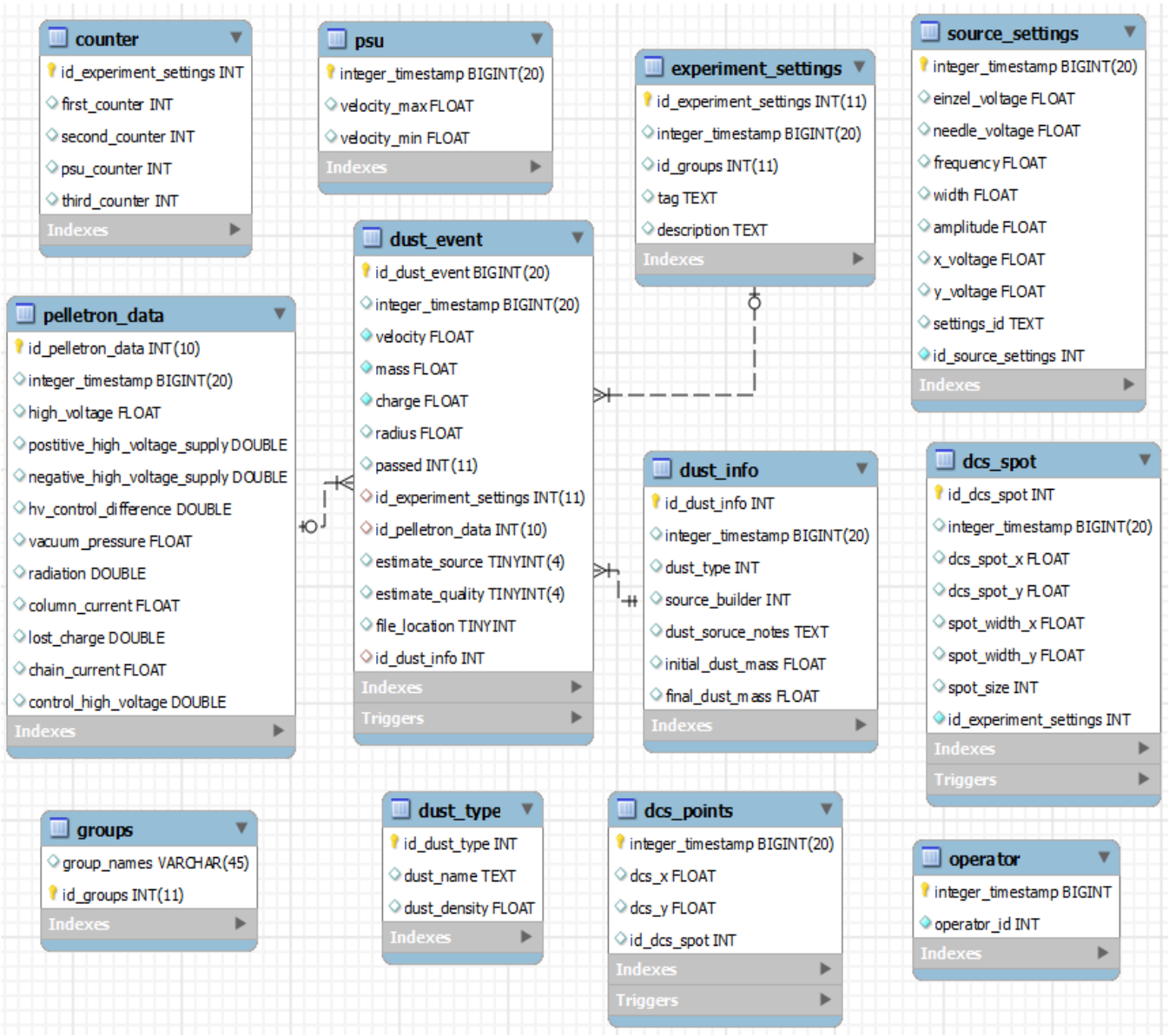
The main schema used for the accelerator database is named CCLDAS_PRODUCTION. This schema contains multiple tables for different data types that are updated and queried at different times during normal operation.

Several tables are updated manually from **database connection.vi**. The **dust_info** table stores information about the dust source. The **experiment_settings** table stores information about the various experiments being run. The **groups** table contains a list of experiment groups. The **operator** table contains information about the accelerator's current operator. The **psu** table stores information about PSU settings for a given experiment.

Other tables are updated automatically. Whenever the pelletron is on, the **pelletron_data** table is updated at a rate of 1 hz. This table stores information about the state of the pelletron. Whenever the dust source is being operated, the **Dust Control Super Giraffe.vi** program updates the **dcs_spot** and **dcs_points** tables with information from the dust coordinate sensor, the **counter** table with information from the detector counters. The super giraffe also updates the **source_settings** table whenever a dust source control is changed or when the program is stopped.

Whenever PSU selects a dust particle traveling down the beamline, the **database connection.vi** program also posts an entry into the **dust_event** table, which contains information about individual dust events such as metadata and the location of their assoicated detector waveform files. Whenever a dust event is posted, the **dust_event** table is also updated with references to the current entries in the **experiment_settings**, **pelletron_data**, and **dust_info** tables so that the dust event can be more easily categorized.

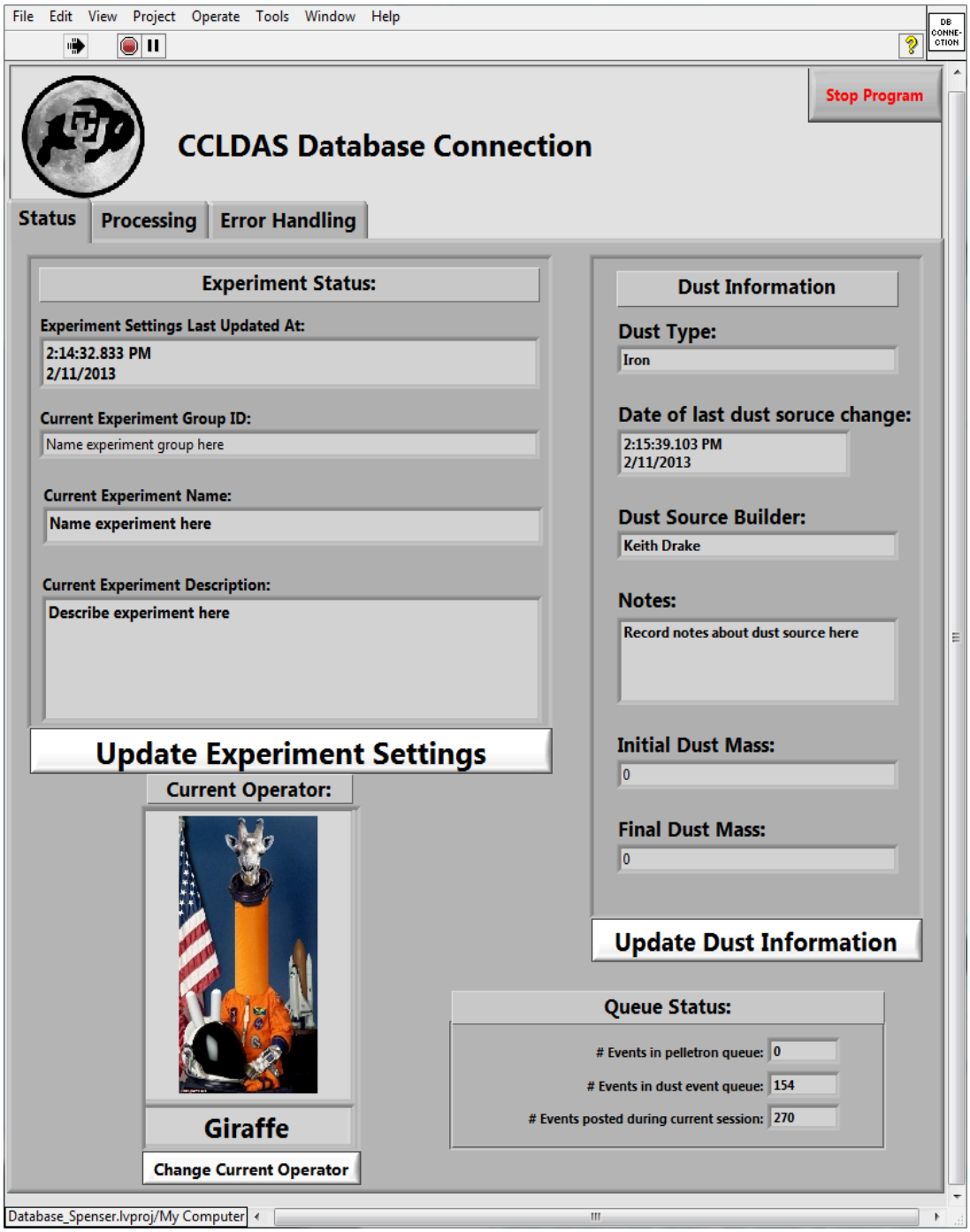
FInally, the **dust_type** table stores information about the types of dust that are used in the accelerator. Given how rarely additions are made to this list, it is updated manually in the MySQL workbench.



Note on future upgrades to the MySQL database model:

When creating a new schema in the database, the **Forward Engineer** functionality is used in the MySQL workbench. However, using this functionality to update an existing database **WILL DESTROY DATA!** To update or change an existing schema, the **Synchronize model** functionality in MySQL workbench should be used instead.

Database Connection.VI: Data enqueuing and experiment management



Use:

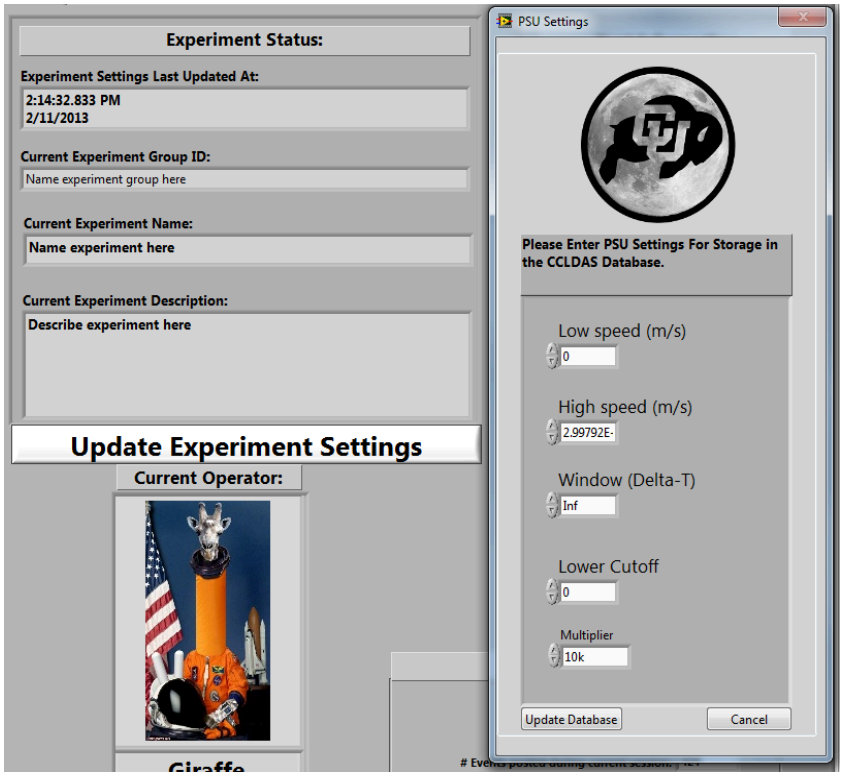
The **database connection.vi** program serves two main functions. First, this VI allows users to change a variety of experiment parameters in the database. Second, it takes

dust events out of a queue, saves individual detector waveforms, and then creates a database entry.

Status tab:

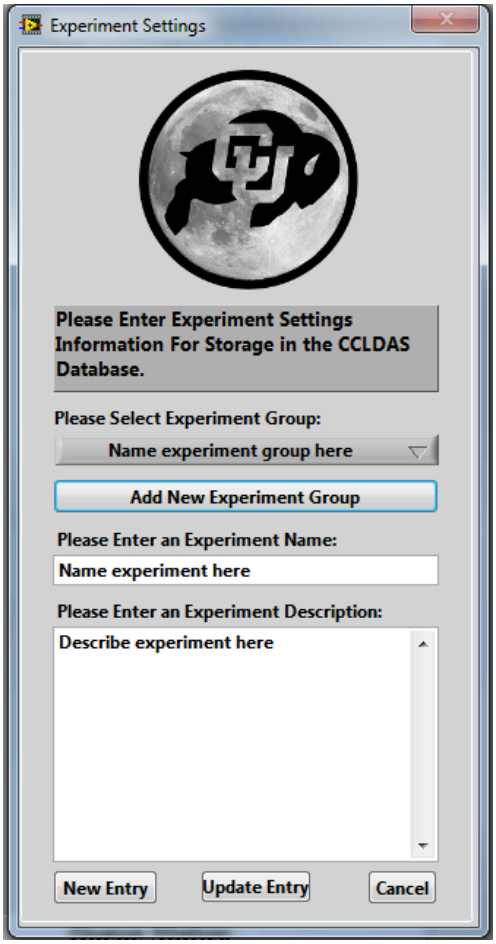
The status tab is where all controls for the database settings are located and where the queue indicators are displayed. For normal operations, this tab is the only one that is of interest to the user.

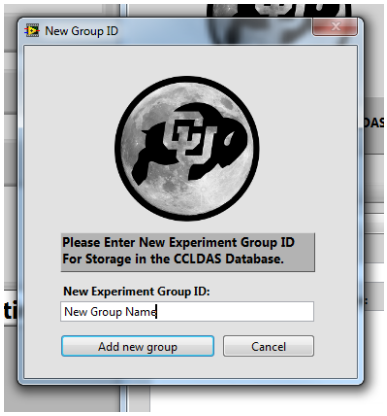
PSU/Experiment Settings:



The front panel of the **database connection.vi** allows the user to see the settings for the current experiment being run and allows the user to update them as needed. Pressing the **Update Experiment Settings** button first brings up a **PSU Settings** pop-up window.

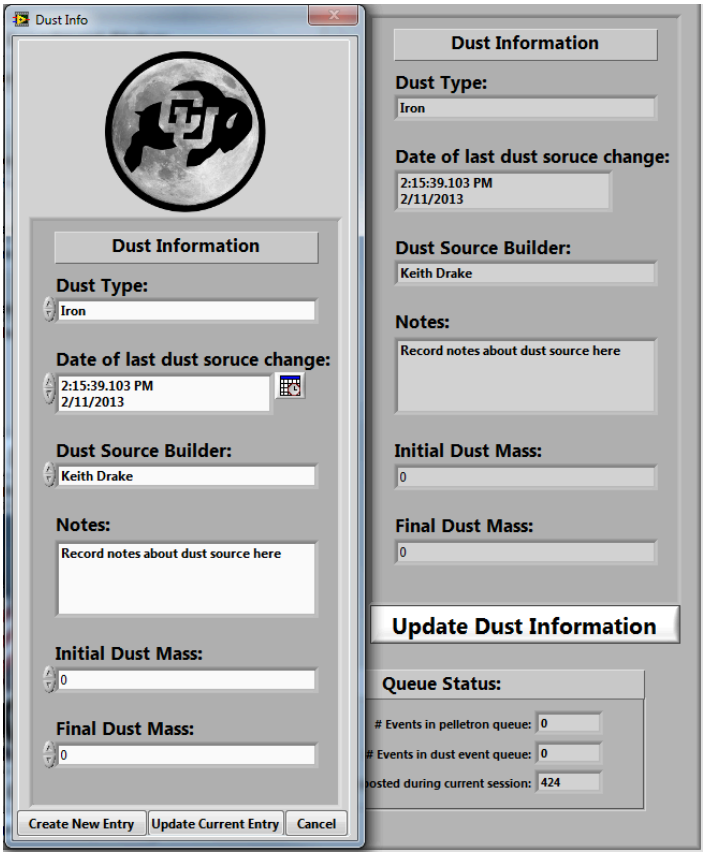
This window allows the user to calculate the settings to enter into the PSU and then updates the database with the velocity limits for the experiment. The user can then define either the desired velocity limits to calculate PSU settings, or enter PSU settings to calculate velocity limits. Pressing the **Update Database** button saves the velocity limits, while pressing the **Cancel** button discards any changes.





Pressing either button in the **PSU Settings** window brings up an **Experiment Settings** popup. This pop-up menu allows the user to select the experiment group, the experiment name, and write a brief experiment description. If the experiment group desired is not yet in the database, the **Add Experiment Group** button brings up a popup that allows the user to add the group to the database. Once the appropriate fields have been filled out, the user can either create a new database entry or update an existing one. The timestamp stored in the database with an experiment settings entry is the time the entry was created.

Dust Information:

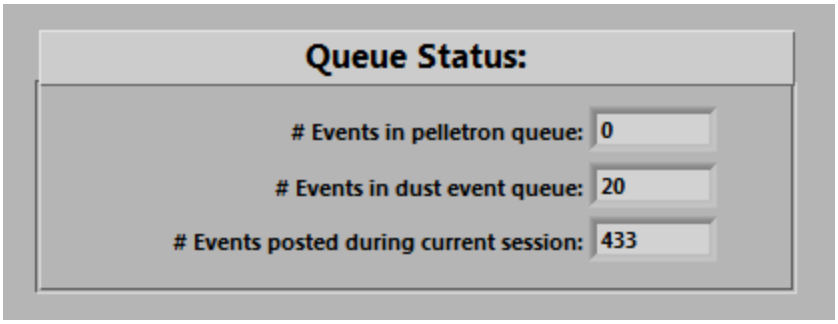


Information about the current dust source being used is also entered and displayed in the **database connection.vi** program. Pressing the **Update Dust Information** button launches a pop-up menu that allows the user to define type of dust being used in the current dust source, the date of the last dust source change, the builder of the last dust source, the initial and final dust mass in the dust source, and any notes about the dust source. Pressing the **Create New Entry** button creates a new database entry, pressing the **Update Current Entry** button updates the current entry, and pressing the **Cancel** button discards changes.

Current Operator:

The front panel of the **database connection.vi** program displays the current entry for accelrator operator and allows this entry to be updated. The picture and name for the current operator is displayed. To change the current operator, the **Change Current Operator** button is pressed. This brings up a pop-up window that allows the user to select the current operator from a list of approved operators (currently, Giraffe, Andrew Collette, Keith Drake, Anthony Shu, and Evan Thomas) by selecting their name and photograph. Clicking the **Update Operator** button changes the current operator to the one selected in the popup, while clicking **Cancel** aborts without changing the operator.

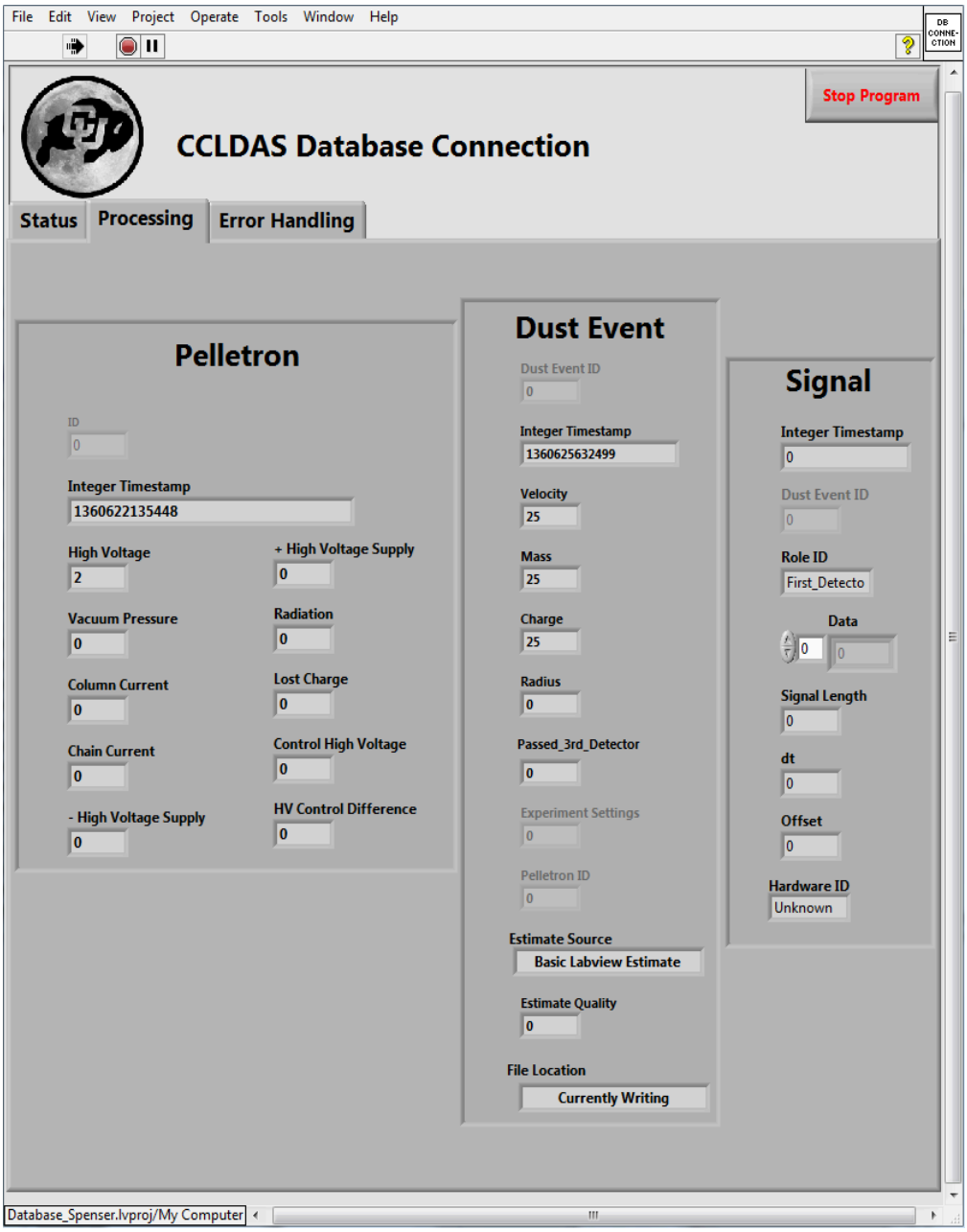
Queue Status:



The **database connection.vi** program also features a **Queue Status** indicator cluster.

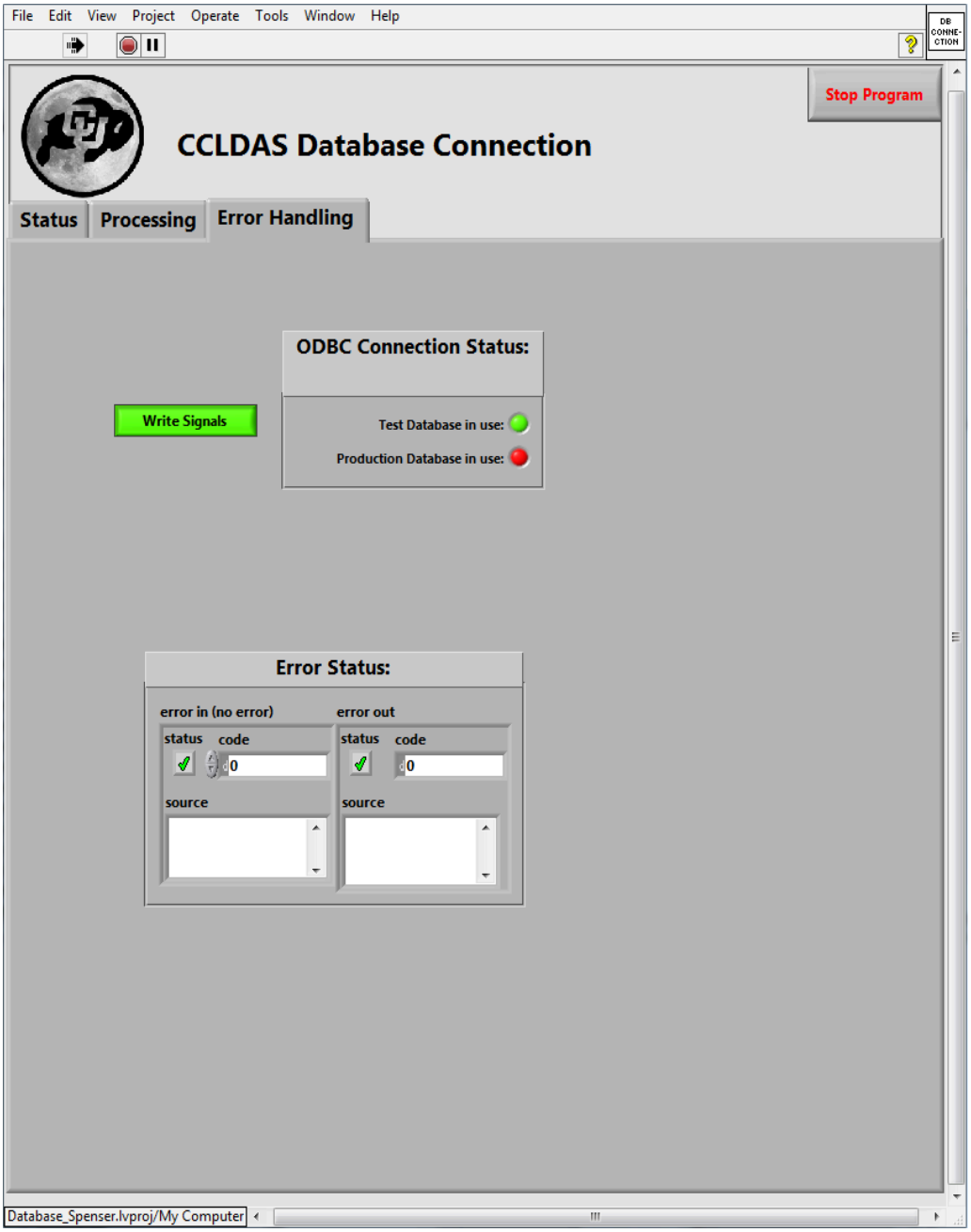
These indicators show the number of events in the pelletron queue, the number of events in the dust event queue, and the number of events that have been posted since the program was opened.

Processing Tab:



The **Processing** tab displays the information currently being posted to the database. This tab is mostly a vestigial remnant of early debugging processes and is not of use during normal operations.

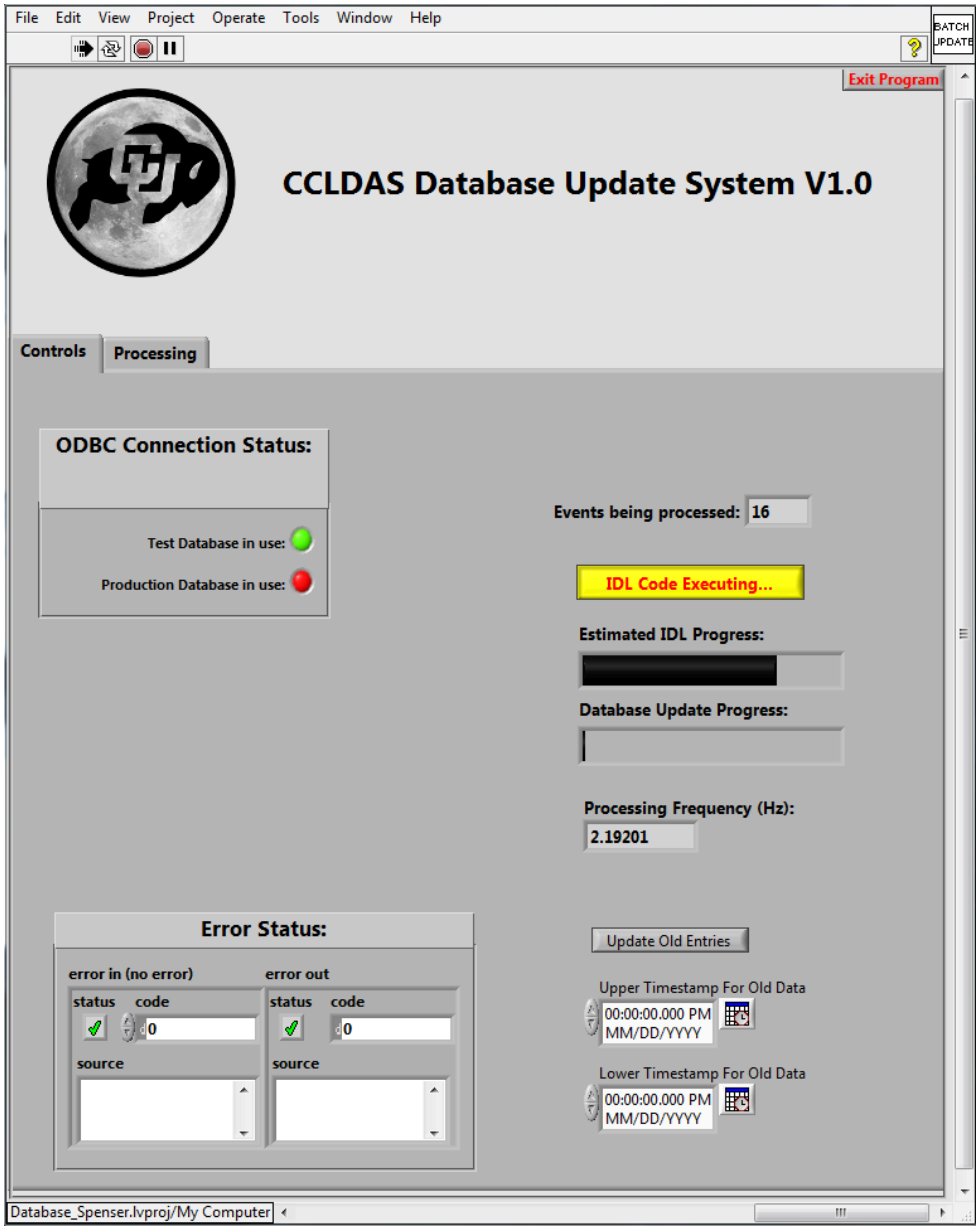
Error Handling Tab:



The **Error Handling** tab displays error information, which ODBC connection is being used, and has a **Write Signals** button that can enable or disable detector waveform

writing. This tab is also mostly vestigial and used for debugging.

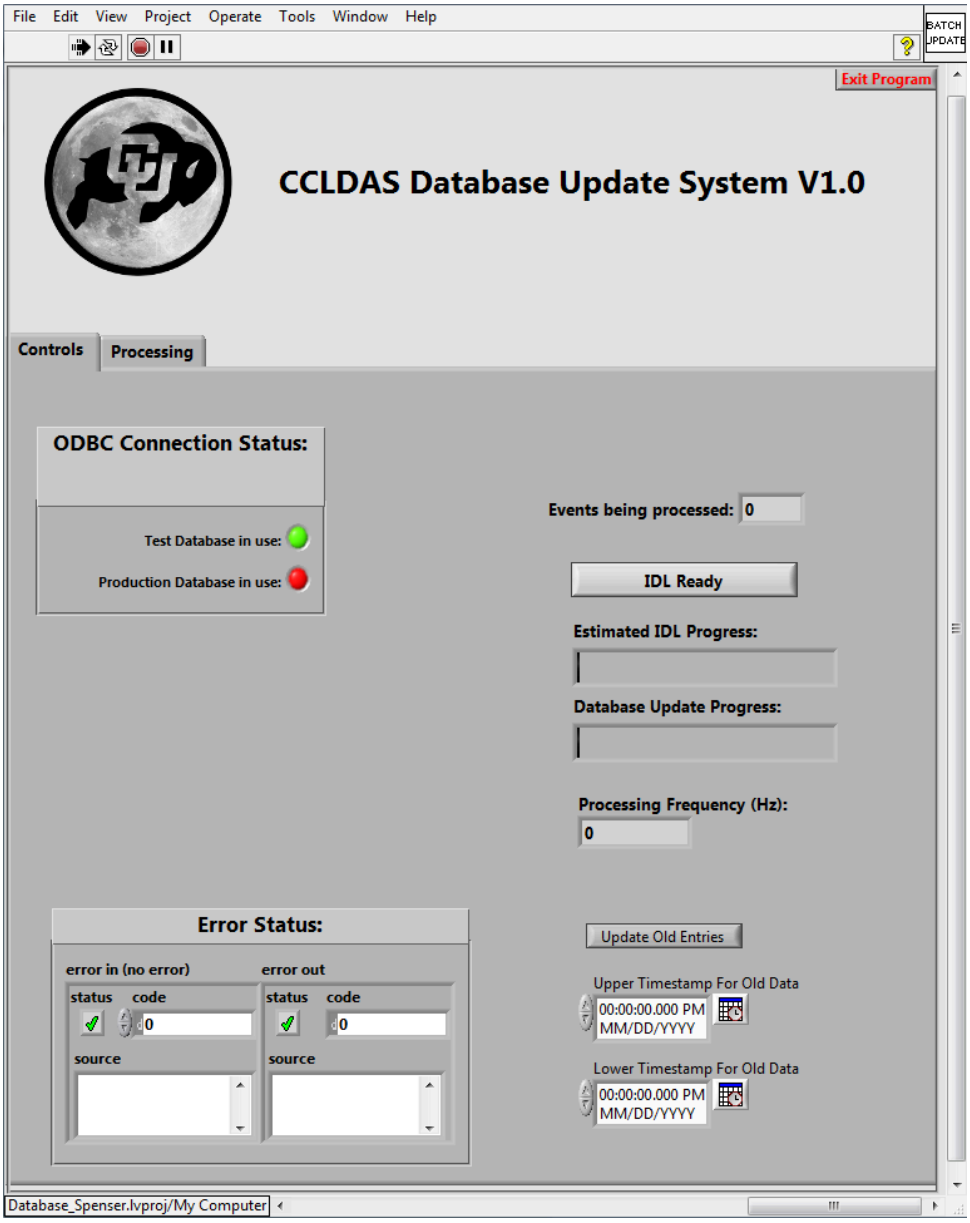
Batch processor.VI: Data analysis



Use:

The program **batch processor.vi** is used to conduct data analysis of waveforms

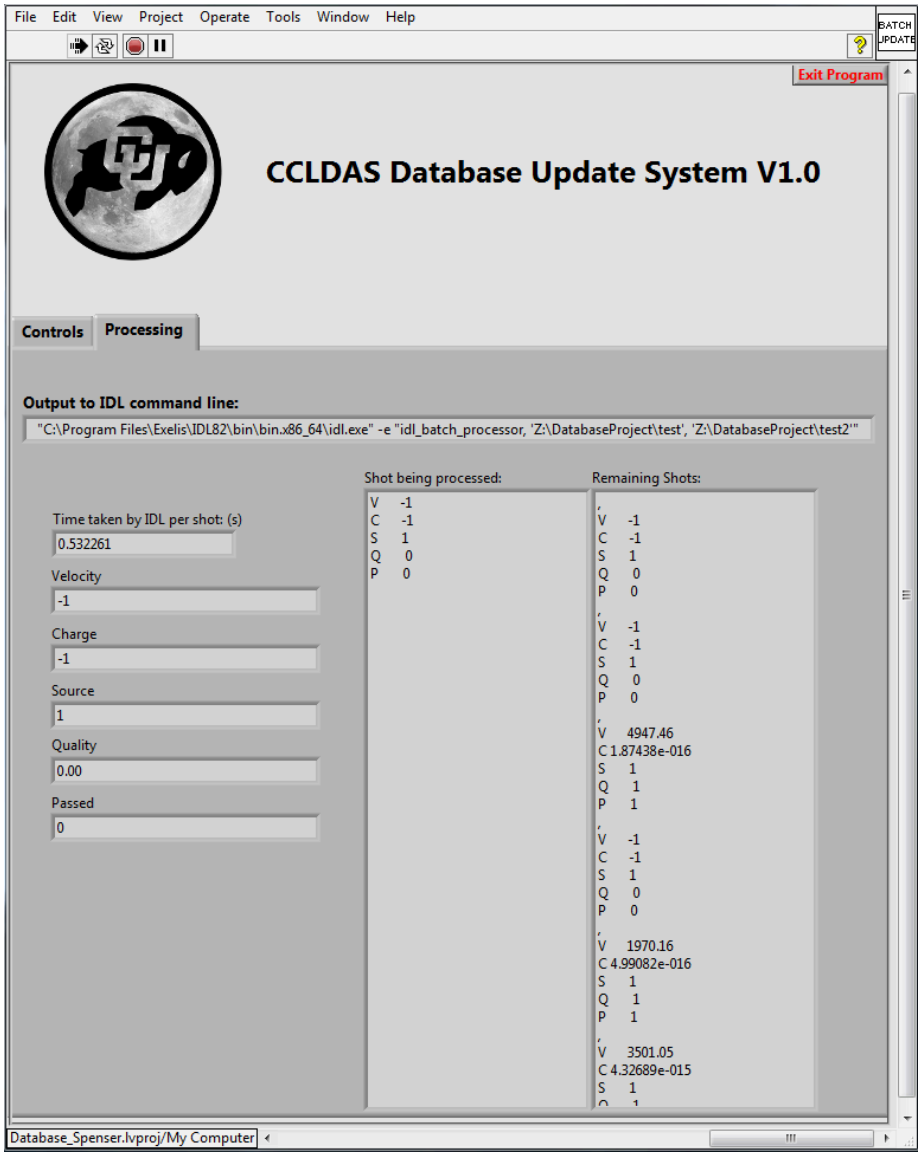
captured by the beamline dectetors and to manage waveform storage on the CCLDAS server. It concurrently queries the database to find new binary labview waveforms, converts these waveforms to .hdf5 format files on a local hard drive, and uses an IDL data analysis program to evaluate the waveforms. Waveforms that have been analyzed are then moved from the local drive to the server.



Controls Tab:

The controls tab is the main user interface for the batch processor and allows the user to verify that the batch processor is operating. The **Events being processed** indicator shows how many dust events are currently being run through the IDL data analysis program at any one time. The **IDL Ready/IDL Code Executing...** indicator shows the

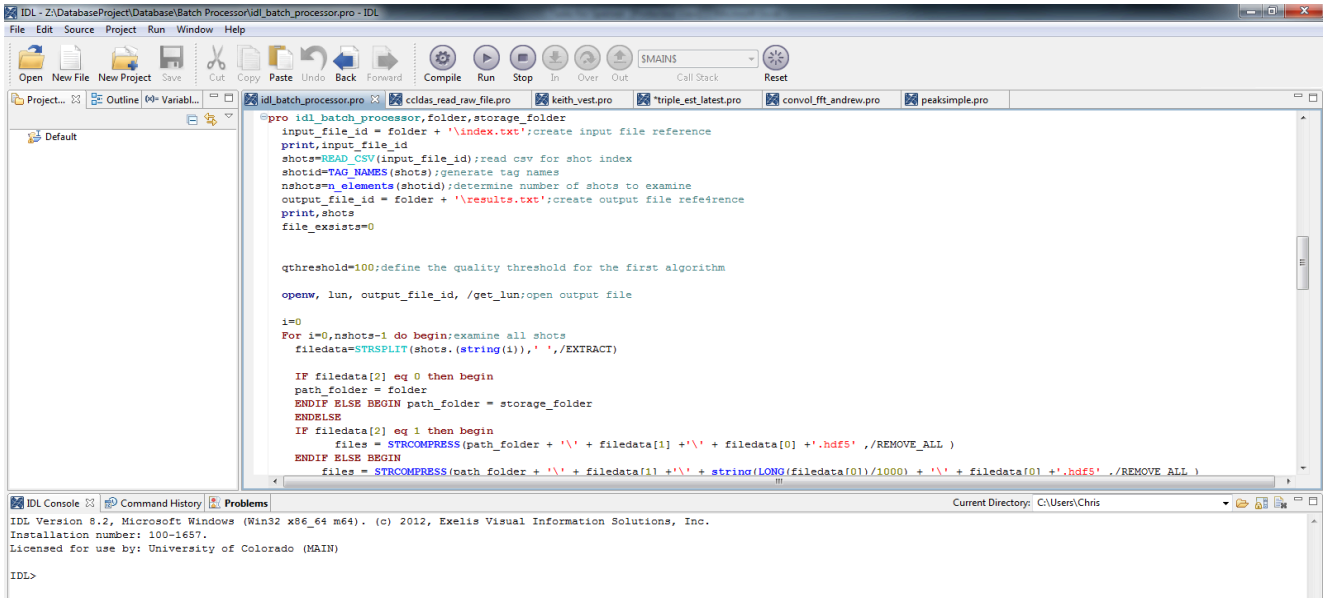
state of the batch and indicates if the batch processor is currently waiting for IDL to finish examining a dataset. The **Estimated IDL Progress** indicator shows the estimated progress of the current batch based on the processing frequency of the last batch. The **Database Update Progress** indicator shows the progress of the database update with information generated by IDL analysis. The **Processing Frequency** indicator shows the calculated processing frequency of the last batch in Hz. The **Update Old Entries** button allows old entries to the database, constrained by the **Upper Timestamp For Old Data** and **Lower Timestamp For Old Data** controls, to be run through the batch processor.



Processing Tab:

The processing tab shows the current command line output to IDL and shows the raw data list generated by IDL. It is mostly a vestigial remnant of early debugging processes

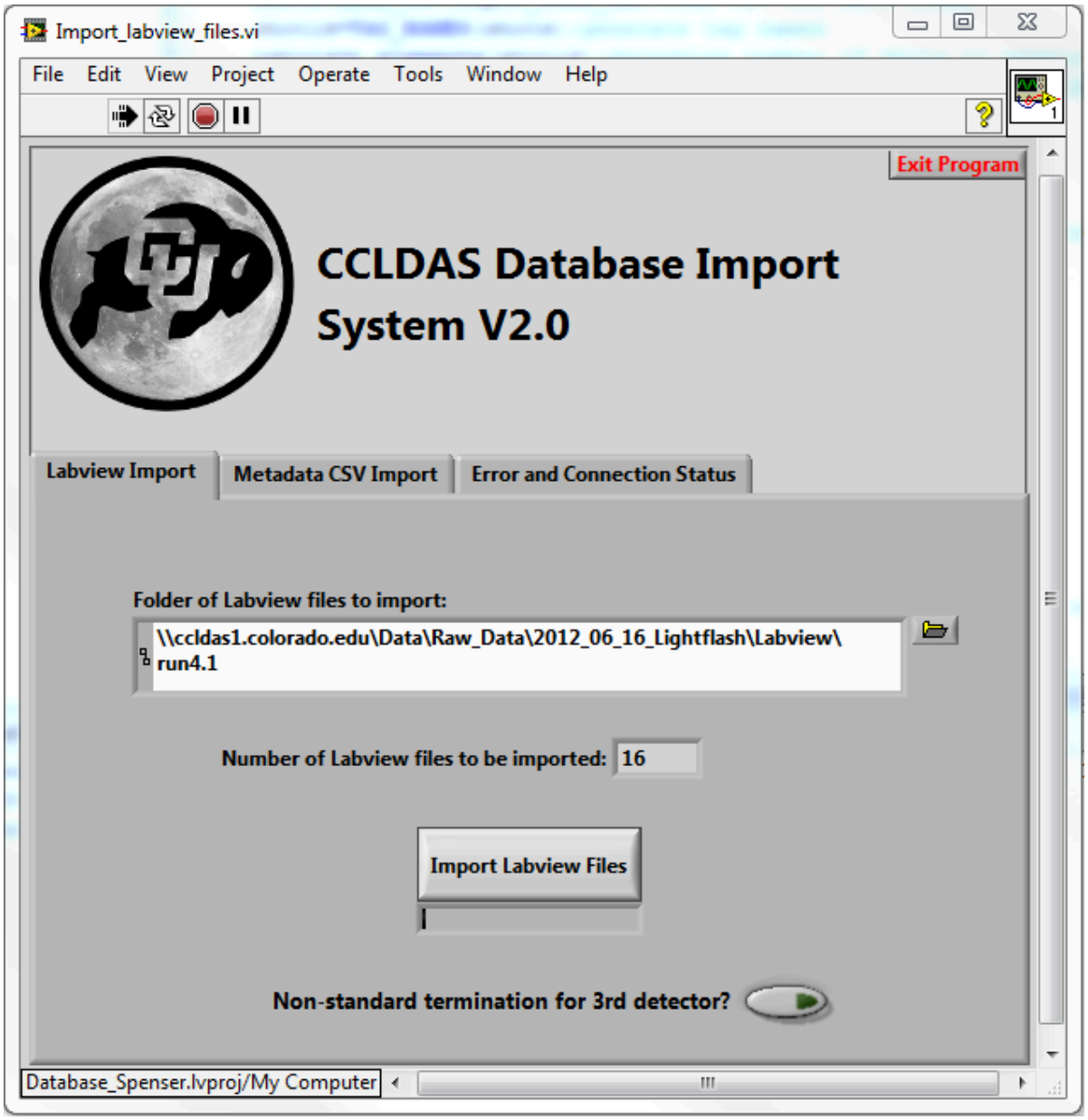
and is not of use during normal operations.



IDL Code Overview:

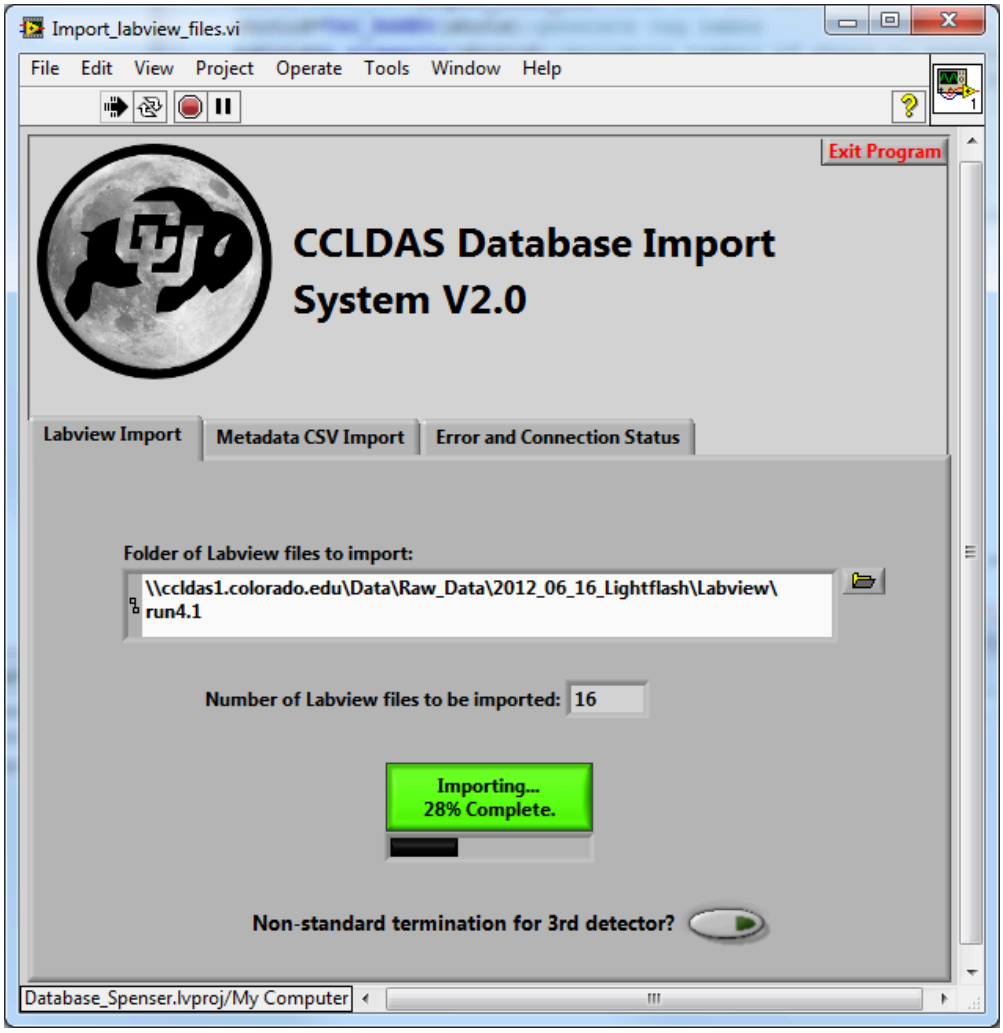
From the IDL command line, the **Batch Processor.vi** program calls an IDL program named **idl_batch_processor.pro.** This program first looks at a text file generated by labview to determine the location of the various waveforms to be examined, then loops over the number of shots to examine. For each shot, the IDL function **ccldas_read_raw_file.pro** is called to open the .hdf5 waveform file and extract the waveforms. The waveforms are then run first through a program written by Keith Drake named **keith_vest.pro** to determine which shots are valid particles and what the charge and velocity of the valid particles are. Waveforms where this algorithm fails to detect a particle are then passed through a second program written by Andrew Collette named **tripple_est_latest.pro,** which has two internal dependancies named **convol_fft_andrew.pro** and **peaksimple.pro**. Waveforms that this program cannot find a valid particle in are then considered false triggers. Estimates for velocity and charge, information about the validity of the particle, and the estimate source are then written to a text file which is then read by labview and used to update the database.

Import_labview_files.VI: Data Import



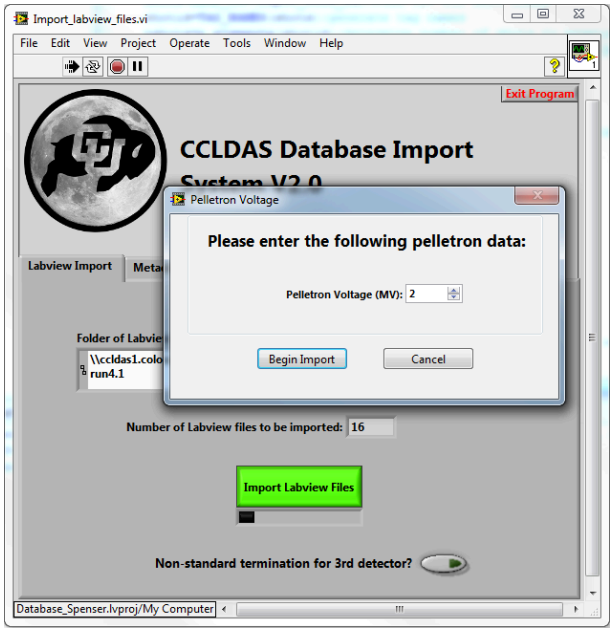
Use:

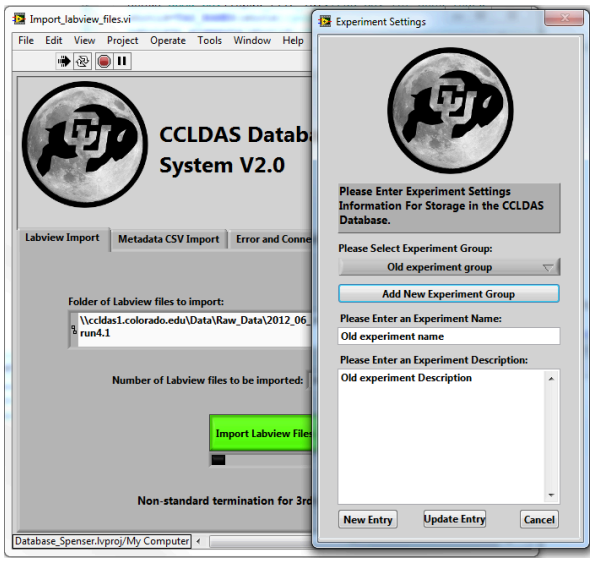
Import_labview_files.vi allows the user to import old data that predates the implementation of the database system into the database. It can take old labview binary files or old metadata CSV files and import them to the current database system.



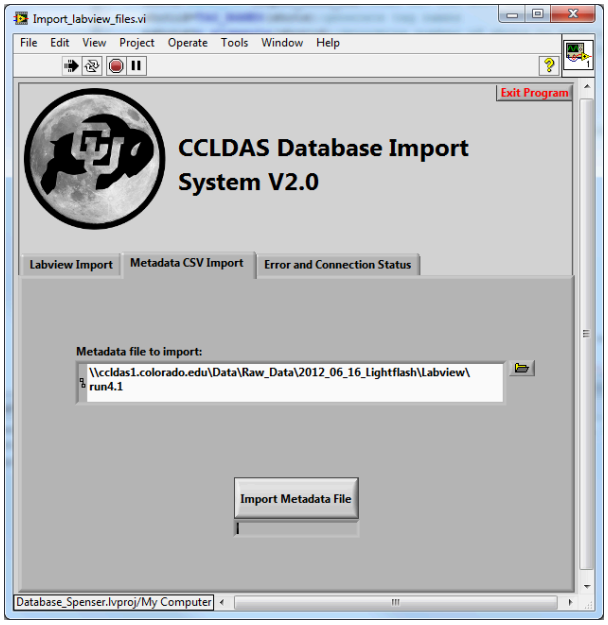
Labview Import Tab:

The labview import tab is used to import old labview binary waveform files generated by the DAQ program written by Keith Drake. These files were used for data storage before the database system was deployed in the summer of 2012. These files also sometimes had a non-standard termination scheme for the third detector. File probably only is with non-standard termination should be imported with the **Non-standard termination for 3rd detector** button on. To import files, the **Folder of Labview Files to import** control is used to select a folder of labview files. The **Number of labview files to be imported** indicator shows how many labview files are in the selected folder. The **Import Labview Files** button is used to begin an import. This launches an experiment settings popup window, followed by a dust information popup window just like the **Update Experiment Settings** and **Update Dust Information** buttons in **Database Connection.vi** to allow the user to imput the experiment name and the dust information for the old experiment.



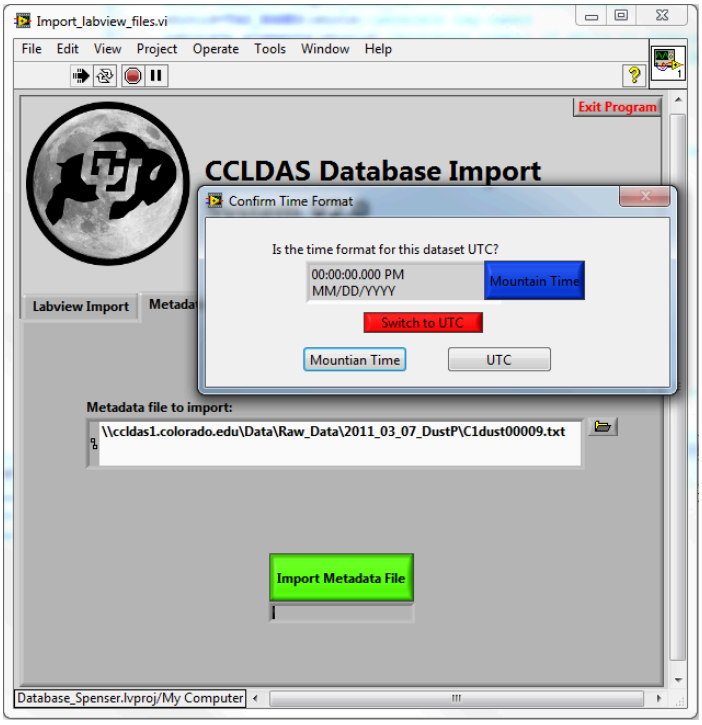


Once this information is entered, the pelletron voltage for the experiment is entered on a last popup window. When this is done, the **Begin Import** button begins the database importation.



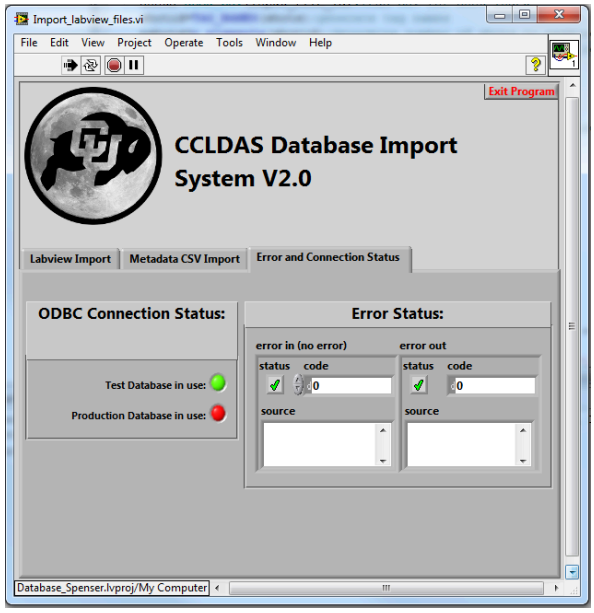
Metadata CSV Import Tab:

The **Metadata CSV Import** tab is used to import old .csv spreadsheet files that predate Keith Drake's DAQ vi. These old files are selected one at a time using the **Metadata file to import** control. Pressing the **Import Metadata File** button launches an experiment settings popup, followed by a timestamp format popup. Whether or not a file is in UTC or local time can be determined by looking at the first entry and seeing if it makes sense as an entry that would take place during normal operating hours in Boulder Colorado.



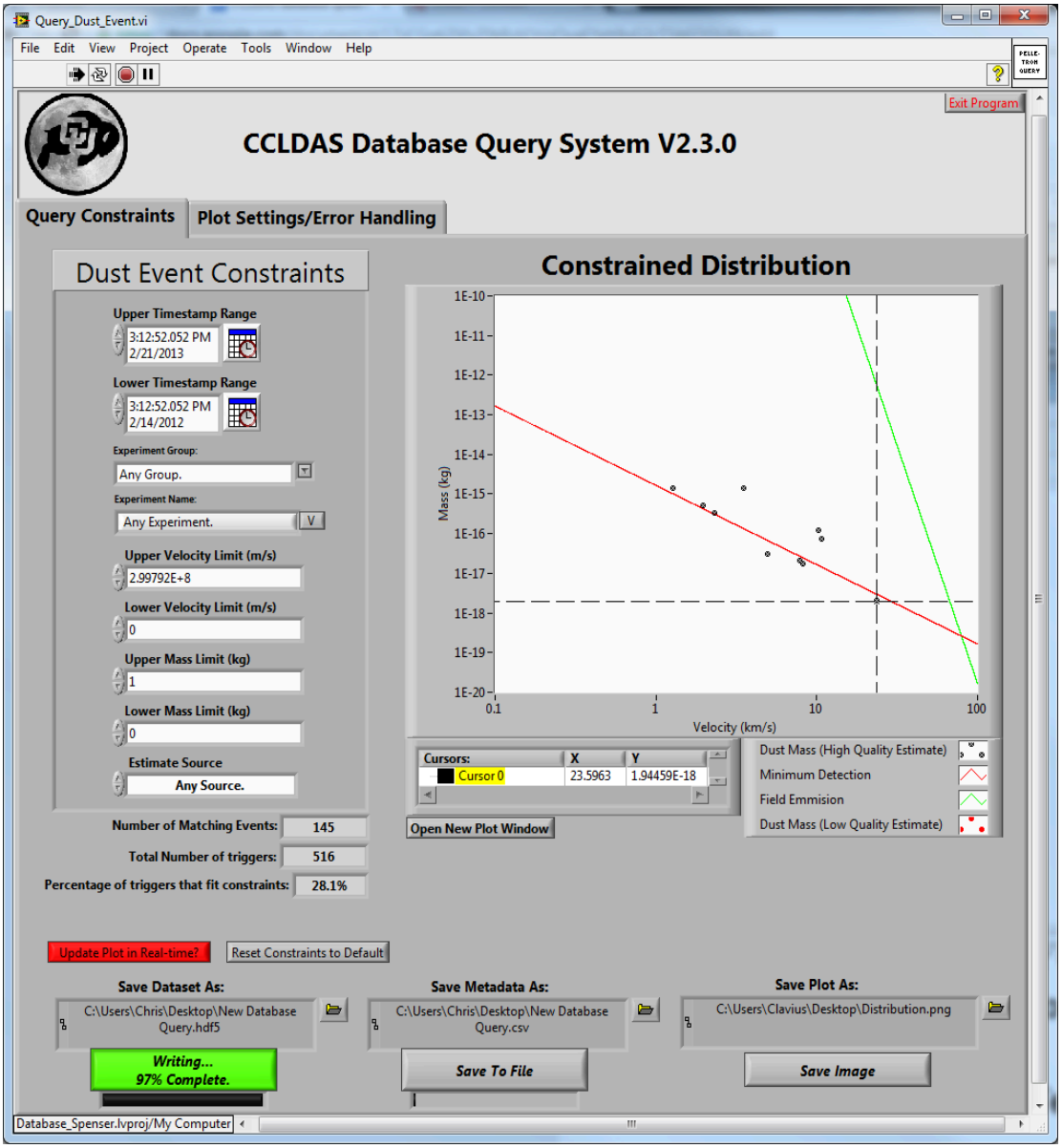
Once a time zone is selected, a dust info popup appears. When dust information is filled out, the data import begins.

Error and Connection Status Tab:



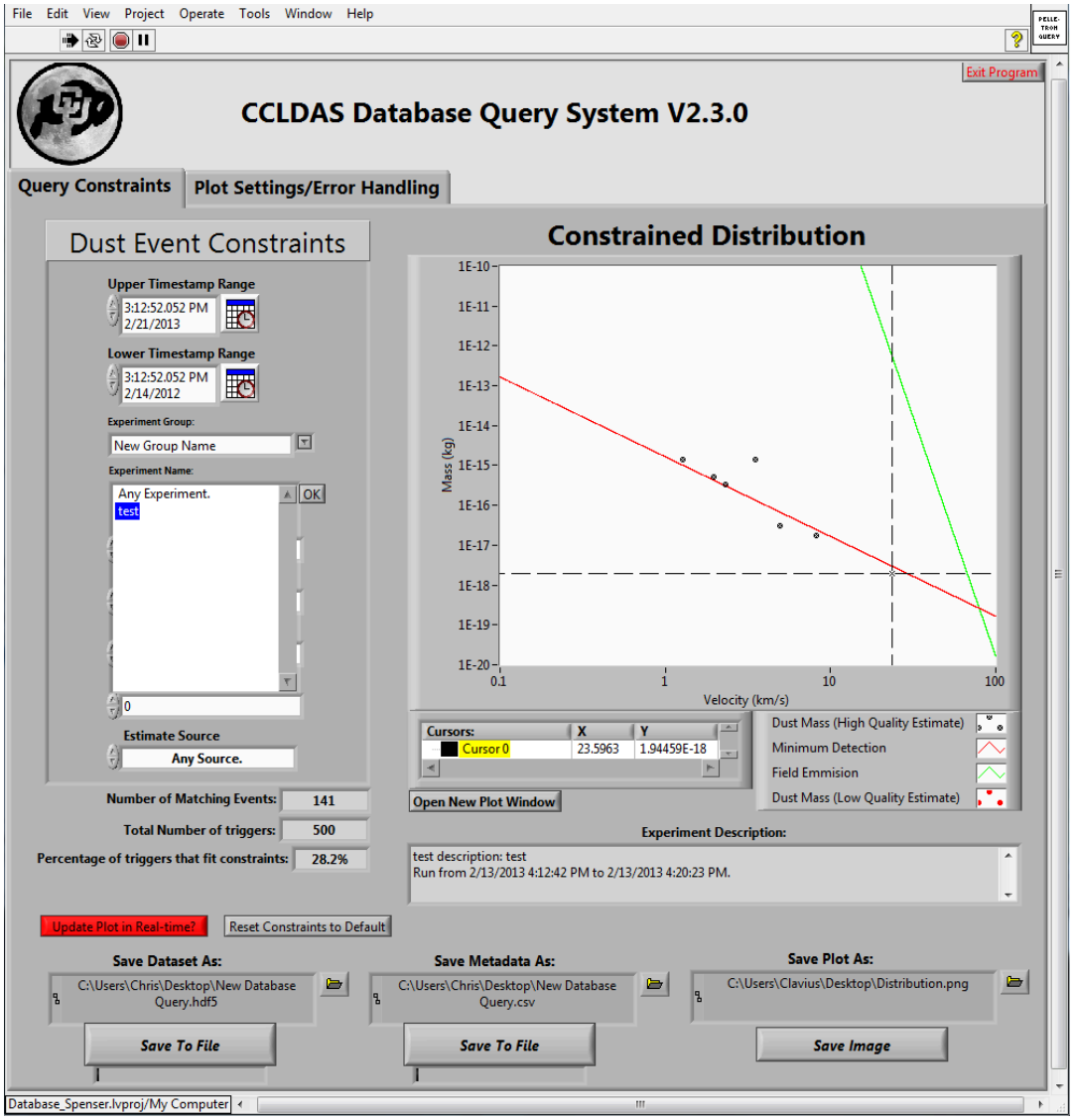
The **Error and Connection** tab shows error and ODBC status and is again mostly a vestigial remnant of early debugging processes and is not of use during normal operations.

Query_Dust_Event.VI: Data Retreval



Use:

Query_Dust_Event.vi is used to retreive dust event information and to display real-time information to the machine operator about what kind of dust events are being recorded by the detectors. Plots, waveform/metadata binary files, and metadata files can all be generated from datasets constrained by experiment group, experiment name, timestamp range, velocity, mass, and estimate source constraints.



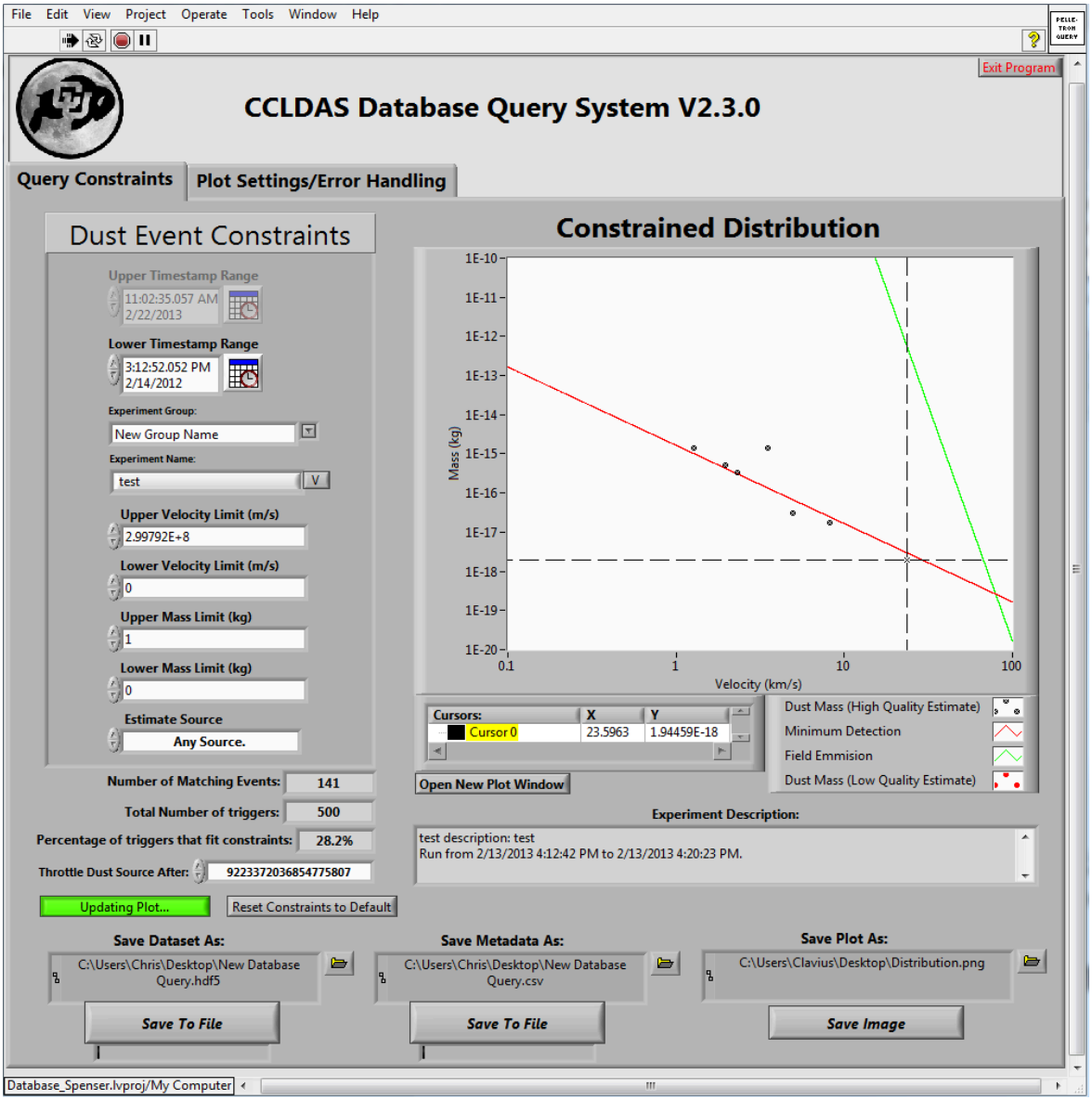
Query Constraints Tab:

The **Query Constraints Tab** is the primary user interface for getting data out of the database. The **Dust Event Constraints** box is used to select a dataset from the database. The **Experiment Group** and **Experiment Names** fields are updated to show the groups and names that exist within a given time range defined by the **Upper Timestamp Range** and **Lower Timestamp Range** controls. The two fields are also linked; if a certain experiment group is defined, only the experiment names within that group are displayed and if an experiment name is defined only groups containing that experiment are shown. When a group and one or more experiment names are selected, an **Experiment Description** box appears that shows the experiment description text string and the timestamp range that the experiment was run at.

Once the time range and experiment are defined, the dataset can be further constrained

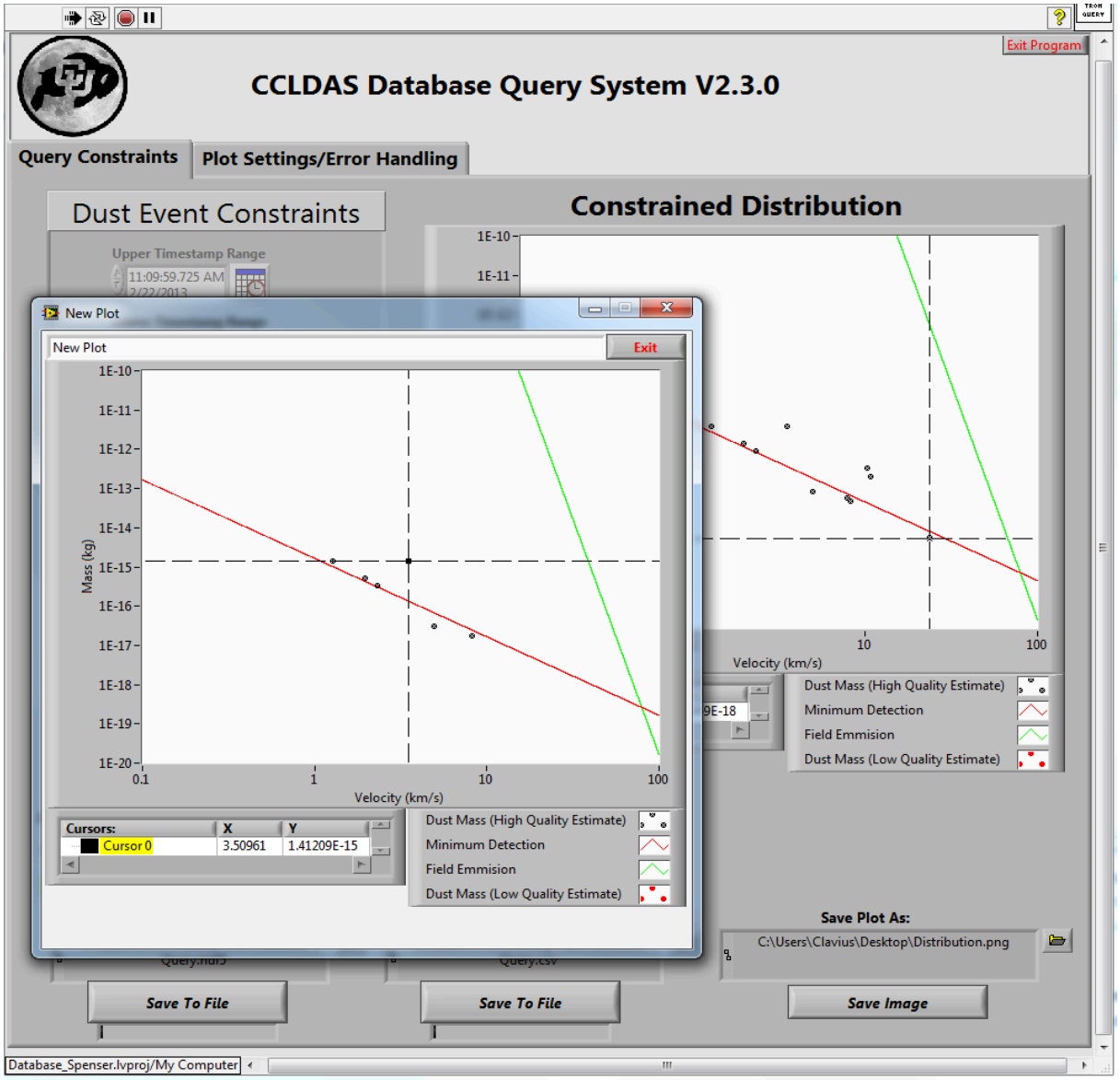
by using mass, velocity, and estimate source controls. Every time a constraint changes, the **Number of Matching Events** indicator and the **Percentage of triggers that fit constraints** indicator are updated. The **Total Number of Triggers** indicator shows how many triggers fit the time range and experiment and is the same number used to calculate the percentage of triggers that fit the given constraints.

Whenever the constraints are changed, the **Constrained Distribution** plot is also updated. Dust events that have not yet been analyzed by the batch processor are shown as red dots on the plot, while valid dust events from the batch processor are shown as black dots. Field emission and minimum detection limit lines are also plotted as green and red lines respectively. A cursor on the plot can be used to find the mass (in kg) and velocity (in km/s) of any particle plotted.



The constraints can also be set to update in real-time during an experiment by pressing the **Update plot in real-time** button. This locks the **Upper Timestamp Range** control and increments upper timestamp limit to the current time every few seconds. This updates both the plot and the dataset selected for writing to files.

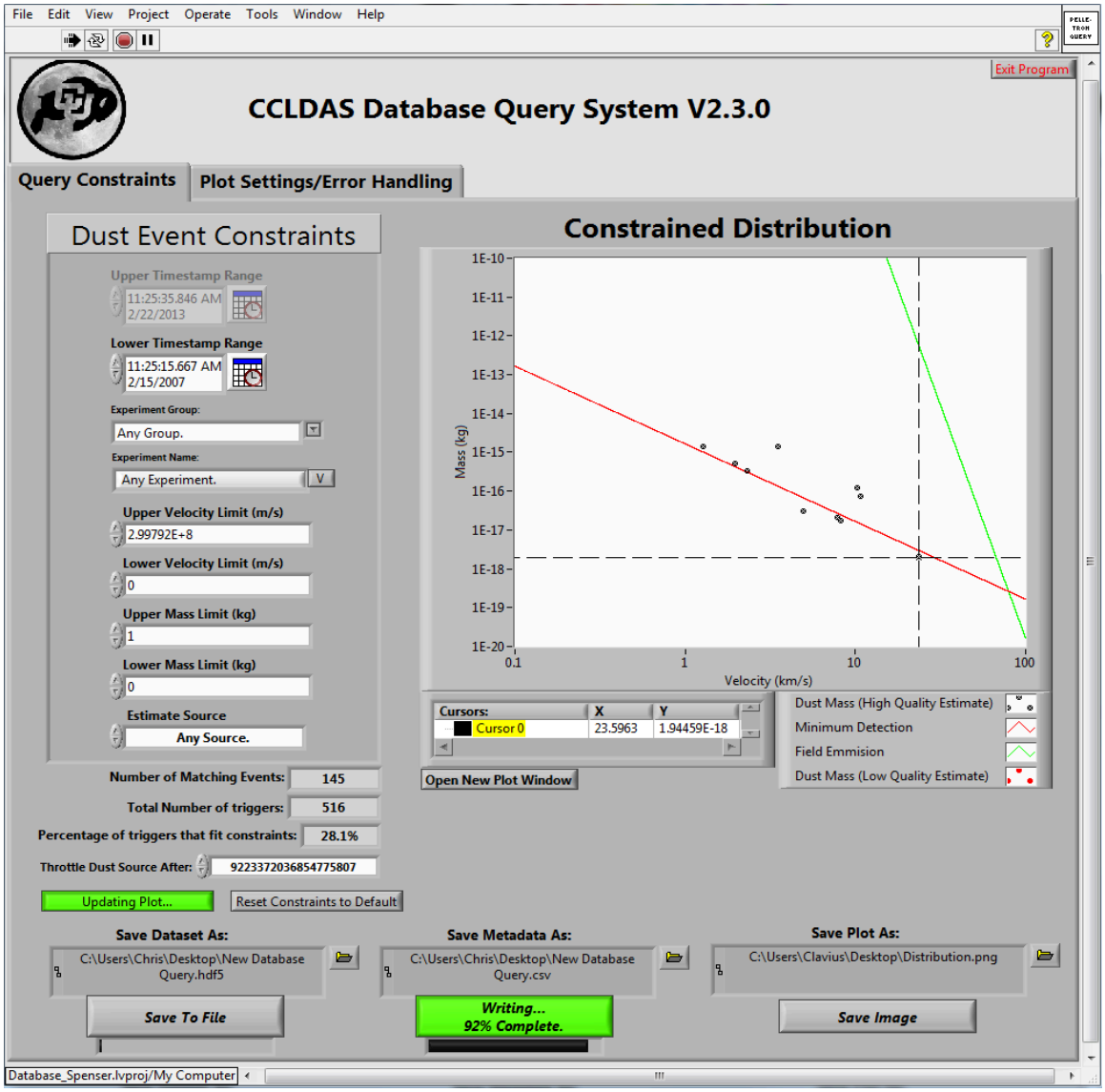
When the plot is being updated in real-time, there is also an option to use a throttling functionality. The **Throttle Dust Source After** control can be used to select the desired number of particles in a dataset. If the number is set, the system will continue shooting until the desired number of valid particles is reached. Once this happens, the super giraffe vi will stop shooting automatically.



In addition to the main plot, additional plots can be generated using the **Open New Plot Window** button. This creates a popup window that replicates the current plot at the time

the button was pressed.

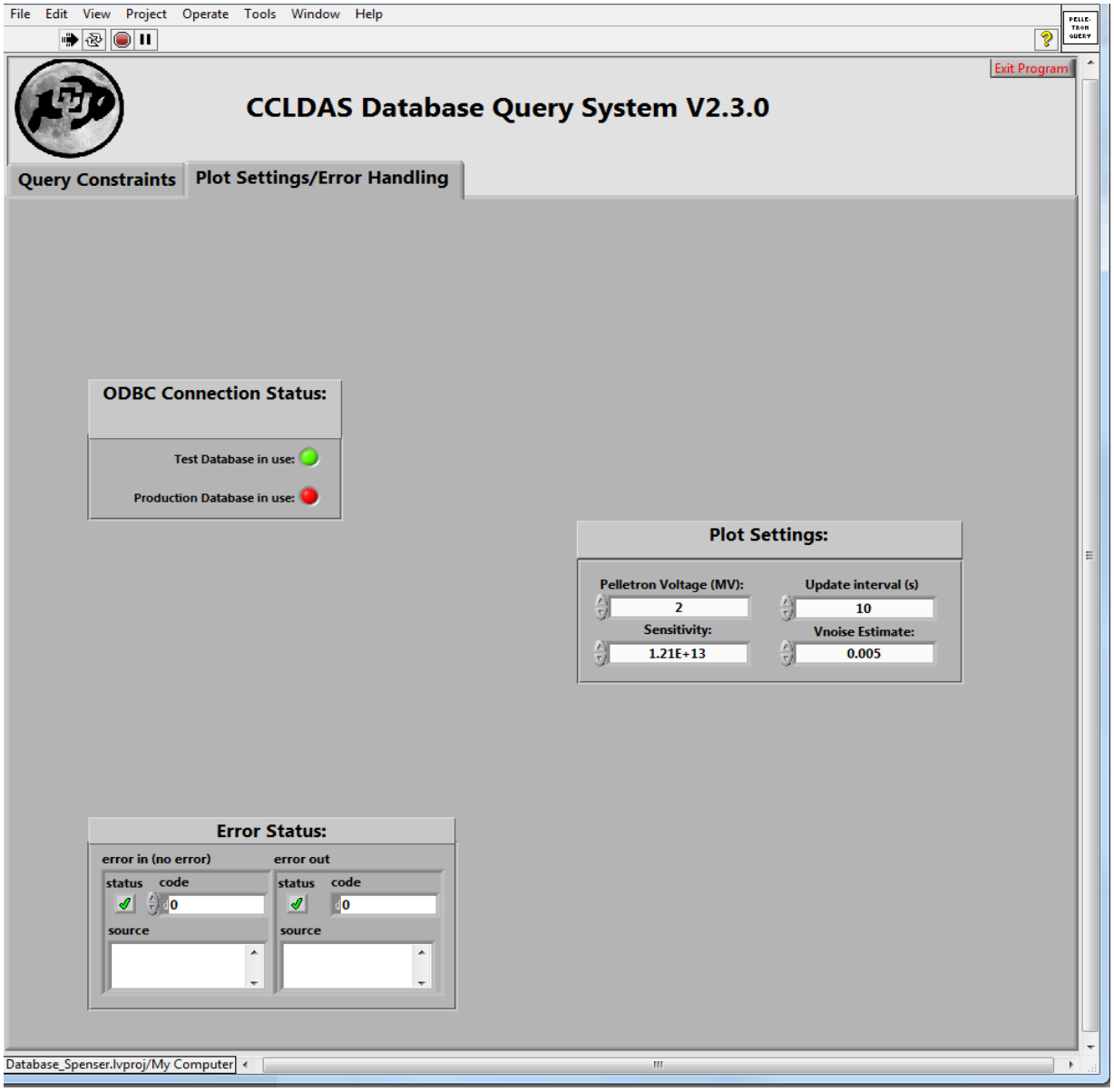
Datasets can be saved in one of three ways. The first, clicking on the **Save Image** button after selecting an image location in the **Save plot as** control, simply creates a .png image file of the current plot.



A .CSV metadata spreadsheet file can be created by pressing the center **Save to file** button after selecting a .CSV file in the **Save Metadata As** control. This file contains the Dust Event ID, a human readable timestamp, an integer timestamp, velocity, mass, charge, radius, a flag to indicate if the dust particle passed the 3rd detector, the pelletron voltage, the experiment group, the experiment name, the estimate source, the estimate quality, and the dust type for each dust event.

A binary .hdf5 file can also be generated by pressing the left **Save to file** button after

selecting an hdf5 file in the **Save dataset as** control. This file contains all the metadata for each dust event and all associated waveforms.



Plot Settings/Error Handling Tab:

The **Plot Settings/Error Handling** tab displays ODBC and error status information for debugging and plot settings such as the plot update interval and controls to define the constants (pelletron voltage, noise, and sensitivity) used to plot the field emission and minimum detection limit lines on the graph.

Dust Control Super Giraffe.VI: Accelerator control

Use:

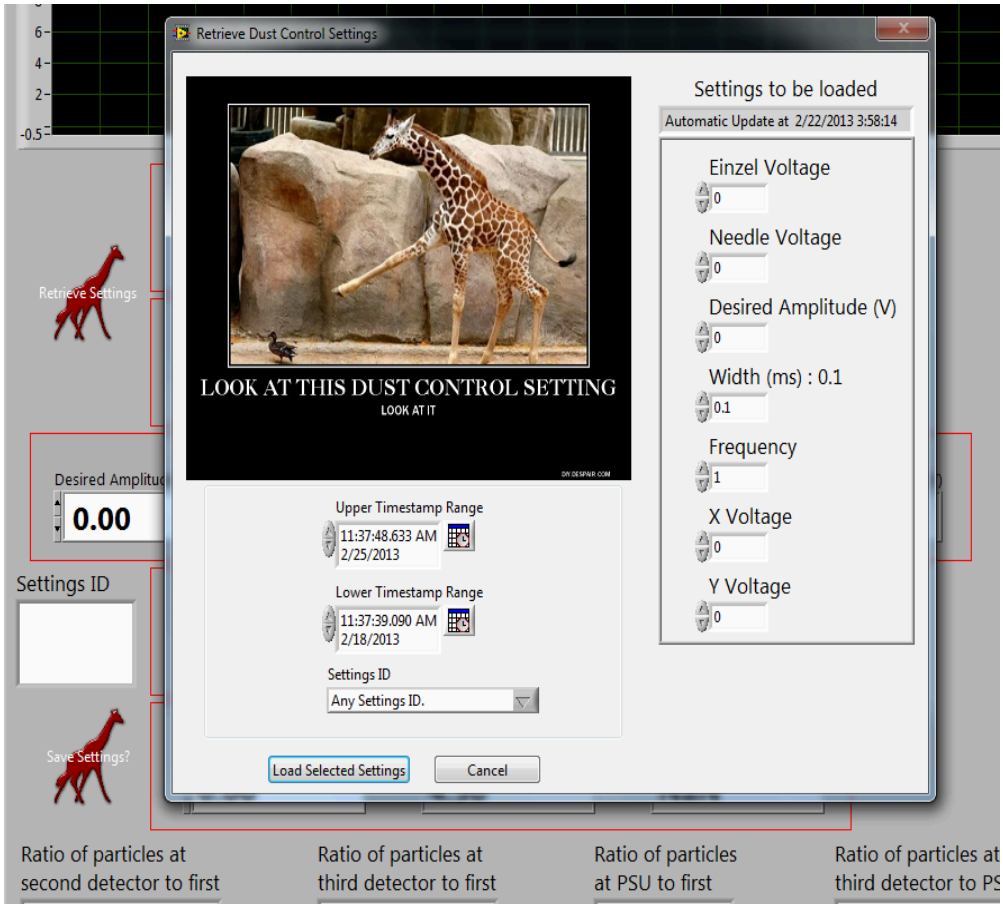
Dust Control Super Giraffe.vi is the program used for actual dust firing. It controls and monitors the pulser in the pelletron, the einzel lens, the XY steerer, and interfaces with the Dust Coordinate Sensor (DCS) and dust counters. It interfaces with the database to store and retrieve dust control settings and DCS data.

Controls Tab:

The **Controls** tab is the primary interface for running the accelerator. It houses the

Enziel Lens controls, the Needle Controls, the Tube Drive controls, and the Steerer controls. It also features monitors for everything that it controls, including a output and actual waveform display for the pulser. Counter ratios are also displayed on this tab to give an indication of how well the accelerator is aligned by monitoring the beamline throughput.

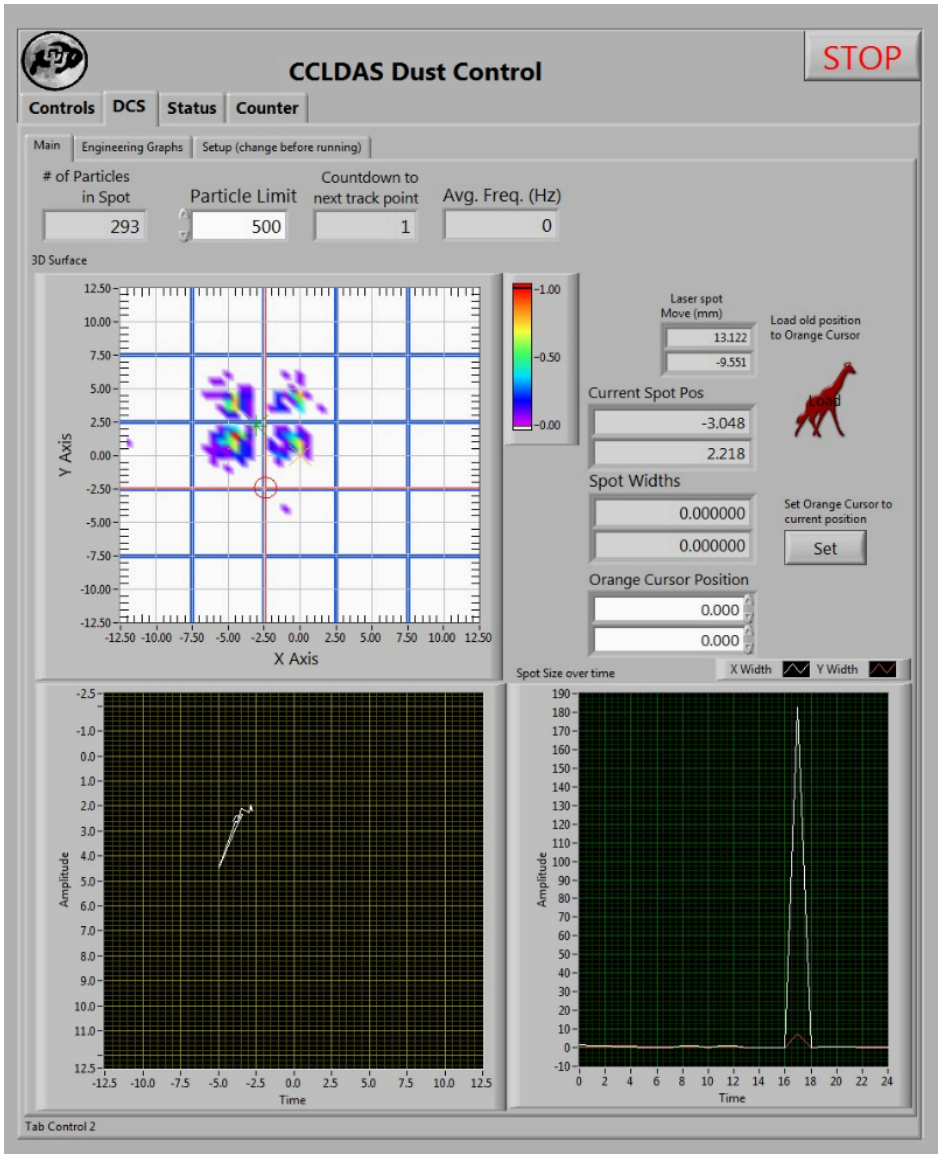
When any of the controls are changed, the system gradually ramps up to the desired value for safety. Settings can be saved by either entering a name in the **Settings ID** field and pressing the **Save Settings** giraffe, or by using the system's automatic settings updater. This saves settings automatically on exit or when a setting is changed.



Retrieve Settings Giraffe:

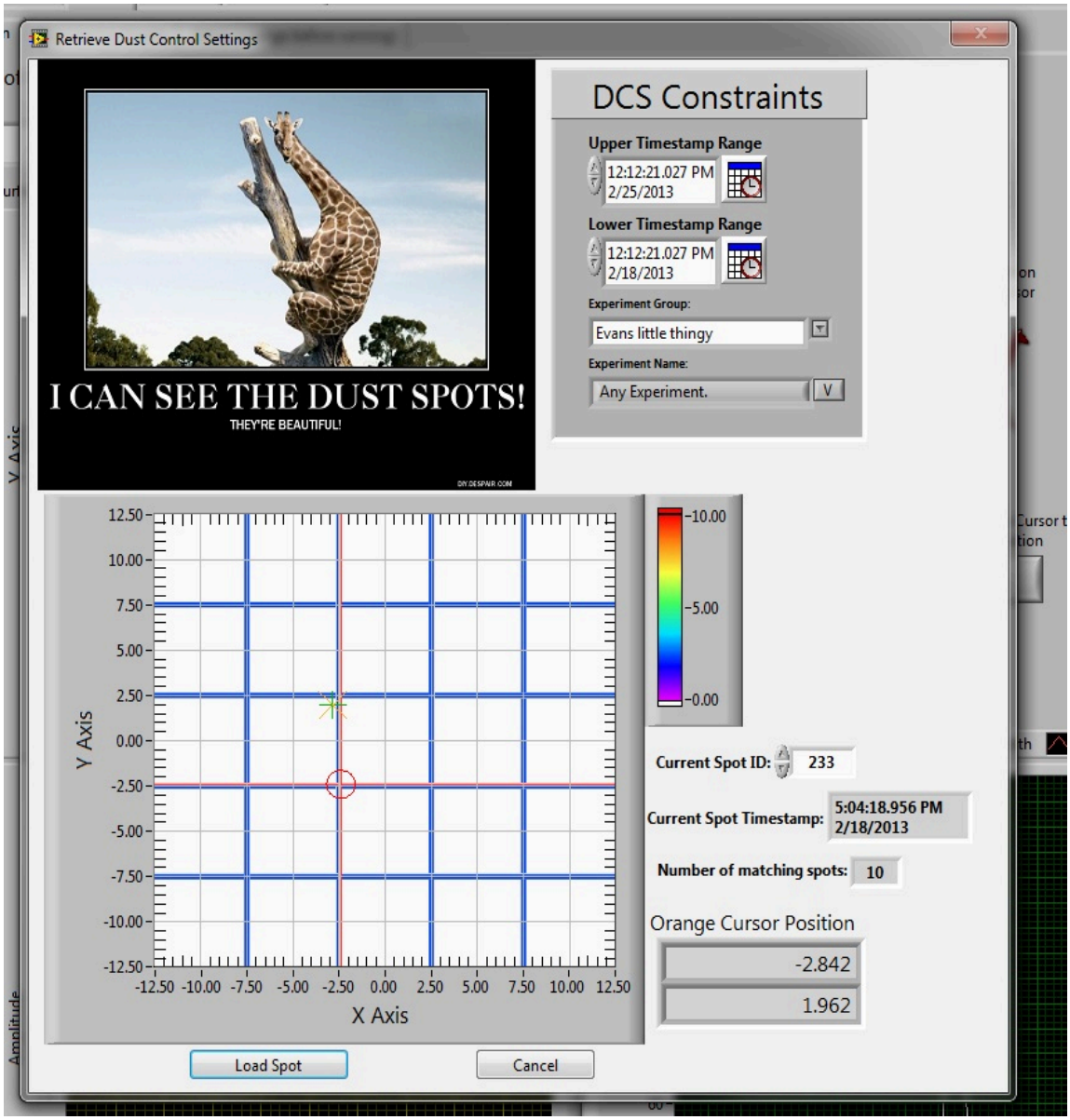
Saved settings can be recovered using the **Retrieve Settings** giraffe. This launches a popup window that can be used to select and load saved settings based on timestamp information. The saved settings within the timestamp range are selected by the **Settings ID** control. The **Settings to be loaded** controls show settings that will be loaded to the machine and can be used to edit settings before loading them. When the desired settings are seen in **Settings to be loaded,** the **Load Selected Settings** button

will load them.



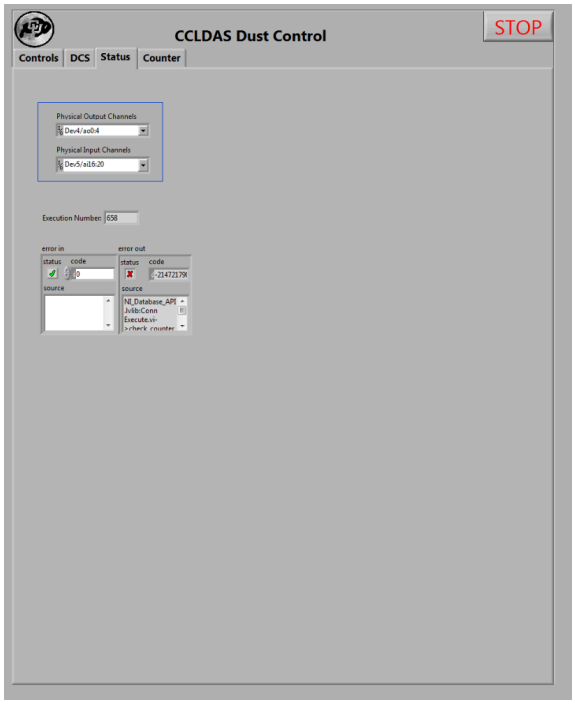
DCS Tab:

The **DCS** tab is used to monitor the dust coordinate sensor and to see where the dust beam is actually pointing. A number of dust particles controlled by the **Particle Limit** control is used to create a dust spot, which is then saved to the database and can be retrieved later to determine where the beam moves over time. Individual dust particles are also saved to the database. Several subtabs also exist for engineering and debugging use.



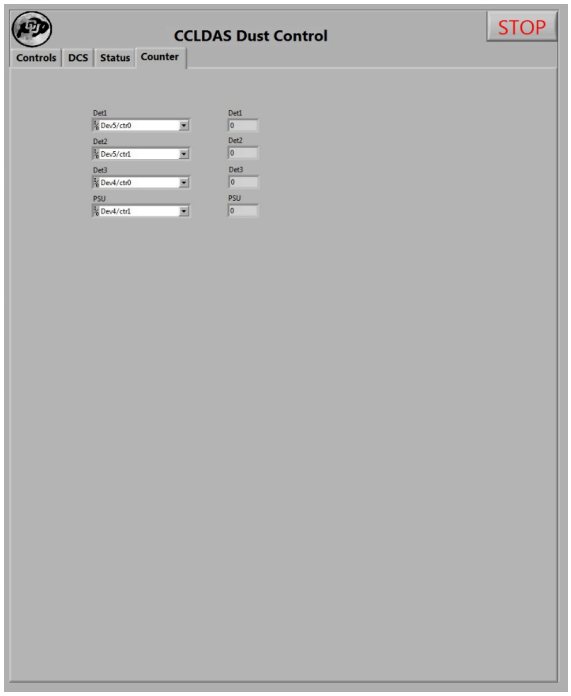
Load Old Position Giraffe:

The **Load old position** giraffe is used to load DCS spots. It launches a popup window that allows old spots to be displayed. Spots are constrained by timestamp, experiment group, and experiment name. The **Current Spot ID** control can be used to select the exact spot desired. Pressing the **Load Spot** button loads the desired spot into the graph.



Status Tab:

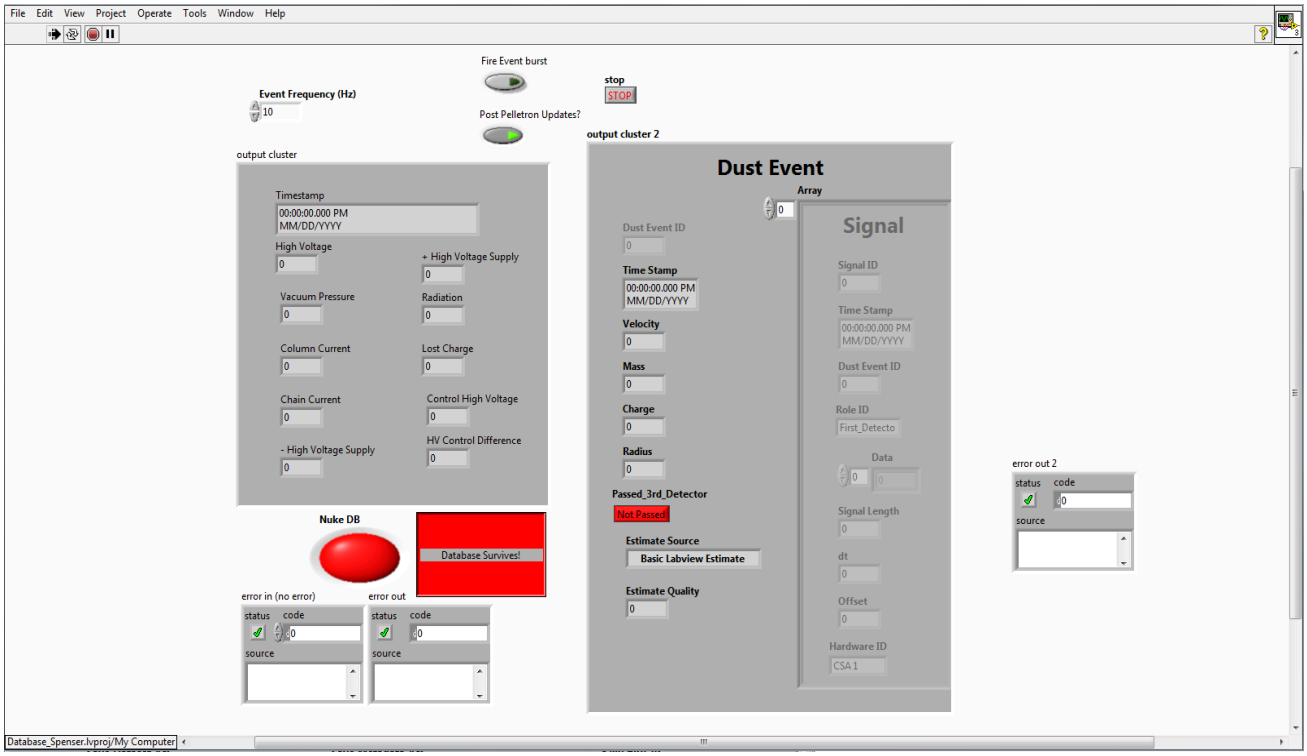
The **Status** tab is used for error handling and to define input and output channels for the dust controls and monitors. It is of limited use during normal operations.



Counter Tab:

The **Counter** tab is used to define the counter input and output channels and to monitor their raw output.

Test VI v3.VI: Testing



Use:

In order to simulate actual operation of the dust event part of the database, a test program named **Test VI v3.VI** can be used. This often-changed program has a crude interface, but can mimic the input to the database system that is normally provided by the pelletron control VI and the detector DAQ VI. Old waveforms, some valid and some false triggers, are used to create realistic signals for the database system to process. Pelletron updates can be toggled on and off using the **Post Pelletron Updates** button, and the **Fire Event Burst** button creates dust events for the database. The **Nuke DB** button can clear old database entries in several tables including the dust event table.

This VI should NOT be used when connected to the main database system. It should only be used on the labview test machine with test databases that lack real data.

About the author/contact info for when something inevitably goes horribly wrong

Spenser Burrows graduated in 2012 from the University of Colorado at Boulder with bachelor's degrees in both engineering physics and biochemistry. He worked from 2010-2013 as a research assistant at the Colorado Center for Lunar Dust and Atmospheric Studies (CCLDAS) designing and building circuits, control systems, and the CCLDAS database system. He has extensive experience in circuit design, CAD design, and programming. He also enjoys flying and has a private pilot's license.

Cell #: 303-253-2634
Email: Spenserjb@comcast.net or burrowss@colorado.edu