# 6/29/2011

The existing NMR program uses LabVIEW 6.1 datalog files to store baseline and event data. Datalog is a proprietary format recommended for use only within LabVIEW. To this day we have stored all of our data in this format and used a custom in-house C program – polcalc – to extract and analyze the data offline. We wish to migrate to an alternative data format for several reasons:

* The datalog format has potential to change with every new LabVIEW release, making upgrades difficult. The format has changed at least once between LabVIEW 6.1 and 2010, and we are uncertain whether our existing tools work properly with the new format.
* The existing method of data storage is unnecessarily complex. A one line description of the system is: files of records of arrays of clusters of arrays of comma-separated lists of values represented as text. Since the existing system already stores numbers as text, wrapping these values into datalog records carries no benefit.
* The datalog files require an in-house program to extract and analyze data. Using a standard ASCII file format will allow us to import data into Excel, gnuplot, Top Drawer, and many other standard data analysis packages.

We wish to convert this system to a more modern event logging system. While NMR is running, it will record data as text in CSV files. A data analysis VI will allow us to easily browse through data from multiple event files, and the comma-delimited format should also be easier to analyze with external programs.

To start investigating how the new system might work, I created a VI that converts our old event files to CSV. TCL Convert Datalog to CSV performs this conversion for one or multiple files.

# 6/30/2011

I next wrote a new version of NMR that writes events directly to CSV files instead of datalog files.

## TCL Init

This VI reads the paths of the event and base files from the config file. It also reads the list of fields (called “fragments”) for each event. I changed this VI to read two lists from the config file: (1) the list of fragments, which have one value per event, and the list of signals, which contain many y values per event. Examples of fragments include Polarization, Area, and CalConst. Examples of signals include RawSignal and PolySignal.

I also removed the option to specify a separate base file name. The user now only supplies one path and one name, and additional files will be labeled appropriately. For example, if one specifies “events” as the event file name, the following files will be used:

* events.csv
* events-RawSignal.csv
* events-PolySignal.csv
* events-base.csv
* events-base-RawSignal.csv
* events-base-PolySignal.csv

This assumes two signals named RawSignal and PolySignal. The extension “.csv” is appended to indicate a comma-delimited text file.

## TCL Log Event to Disk, TCL Log Base to Disk

I combined these two VIs into one, since the only difference was what global variable they were using for the file name. One can now use this VI for both event and base logging by specifying a different file name. Given a single event path and file name (such as “events” as in the example above), the VI writes to one CSV file for the fragments and one CSV file for each signal. One must also specify a fragment as a primary key (usually EventNum), which associates signals with events.

# 7/8/2011

## TCL

This is the main VI that receives messages from other components, decodes them, and performs the proper action (create new file, acknowledge ping, return address, and log event). A value of “0” for the Baseline fragment indicates that an event should be copied to the baseline file. I modified this VI to properly record signals along with fragments for each event. Note that signal data points do not have an associated time stamp; therefore, data may become corrupt if events do not arrive periodically, all containing the same set of signals. We may wish to improve this in the future.

The old TCL system was case-insensitive to fragment names. The new system is case-sensitive.

Strangely, the existing TCL VI has a “stop” state that never executes. A hander exists for the “Stop” message, but it does nothing except flip the state of a hidden front panel variable. I changed this such that a “Stop” message really does stop running the TCL VI. I believe this was the original intent.

# 7/11/2011

## TCL Open BaseFile, TCL Close

These VIs are rather useless, as they contain nothing but a call to another VI. I removed them.

## TCL Baseline Lookup, TCL Get Valid Baseline

The TCL Baseline Lookup VI returns the values of specified fragments for a list of specified events. Despite its name, the VI works fine with event files as well as baseline files. The TCL Get Valid Baseline VI is similar, though it looks up a signal (as an array of doubles). This VI contains code that checks whether the RF frequency of a baseline matches the current RF frequency; other than this code, the VI is not specific to baseline files.

While I hate to make changes that propagate through other components of NMR, I hate even more seeing a VI whose scope is limited only by a poorly-chosen name. Therefore, I renamed TCL Baseline Lookup to “TCL Get Fragments,” and TCL Get Valid Baseline to “TCL Get Signal.” Both VIs will work equally well with event and baseline files. The RF checking code is in a new VI called “QCA Verify Baseline,” which is part of the QCA system.

# 7/12/2011

I encountered difficulties making the TCL Get Fragments and TCL Get Signal VIs completely generic. In the existing NMR program, the current event and baseline file paths are stored in TCL global variables. Other sections of NMR should not access these variables. Therefore, they cannot tell the TCL Get Fragments and TCL Get Signal VIs which file to use.

While the optimal solution would do away with global variables altogether and place a responsibility on VIs using TCL to call TCL Init and keep track of the event and baseline paths it returns, implementing such a system at this point would require significant changes to the structure of NMR. I do not wish to make such changes; therefore, I created TCL Get Current Signal and TCL Get Current Fragments, which uses the global variables to access the current *event* (not base) files. I expect that this change will not be noticeable to a typical user, as all baseline events are already stored in the event file. The new VIs will have no trouble finding baselines there; the only feature ever used to distinguish a baseline was and is the presence of a “0” in the Baseline fragment.

The net effect of this change is that baseline events and signals will still be copied to a separate baseline file, but NMR will not use this baseline file for any purpose. Therefore, let this be a proof of concept of the superfluity of the baseline file. Should no one discover a useful purpose for storing baselines separately, we may wish to consider removing the feature entirely, so as to avoid unnecessary complexity in the program. At this point in time, I see no disadvantages to simply storing baselines and events in the same file.

Note that TCL Get Fragments is now simply a special case of the new and improved TCL Select Fragments (see below) that uses no criteria, therefore, returning the desired fragments for all events in a file.

# 7/13/2011

## TCL Baseline Match

This VI acts as an SQL select statement, returning specified fragments from events where other specified fragments match specified values. Again, this VI is more generic than its existing name implies: I renamed it “TCL Select Fragments” and provided a “TCL Select Current Fragments” version that uses the current event file. I also extended the functionality of this VI, adding the ability to select various Boolean operators (is equal, is not equal, is greater…etc) for each criterium.

Strangely, the existing VI discards criteria fragments that do not exist in the event file. I consider this unconventional behavior, as one would expect “does not exist” to count as no match, though this VI counts it as a match. Despite the illogical behavior, to avoid breaking existing systems, the new VI mirrors the functionality of the old. If the user relies on this functionality, the VI injects a warning into the error stream indicating that the behavior is deprecated.

# 7/20/2011

I created a VI called “TCL Convert Datalog to CSV” that converts old event files to the new CSV format. New files will have the same name as the old files, though with the .csv extension, and separate files for the signals, which the user must specify when running the converter. The converter is somewhat slow, though the user can select multiple files for batch processing.

# 8/1/2011

The “TCL Event viewer” VI allows one to browse event files in the new format. One can filter events by any basic criteria, plot a specified fragment across multiple events, and review signals for single events. The VI also has a “real-time” option that automatically refreshes to show the latest events as they arrive from TCL. I added a notifier to TCL to facilitate this functionality.

# 8/3/2011

I optimized the Select Fragments and Get Signal VIs. Everything now runs significantly faster, especially the event viewer.

# 1/9/2012

After a few cooldowns, I discovered that searching through the entire event file to find the baseline for every data point becomes impractical as the event file grows large. I switched back to using separate event and baseline files for now. Hopefully this will fix the memory leak and baseline selection problems. TCL Get Base Fragments, TCL Select Base Fragments, and TCL Get Base Signal now do the equivalent of their TCL \_\_\_ Current \_\_\_ counterparts for the baseline files.

# 1/13/2012

I rewrote TCL Get Signal to read in files one line at a time, vastly reducing memory requirements. Use of a single file for events and baselines now works fine. Accessing a signal at the end of a file (worst case) takes about 2/3 second for an event file containing four days of data. Hence, at our usual rate of data acquisition we can continue to use the same file for several weeks before this delay becomes a problem (although such files are not recommended, as they would be gigabytes in size and annoying to browse through). I reverted the changes from 1/9; reading baselines from event files is once again standard procedure, and baseline files are once again deprecated.

# 1/17/2012

For consistency, all TCL VIs now accept a single “event file” pathname. This should be the pathname of the files storing the fragments. When accessing signals, the pathname will be correctly dissected to specify the pathname of a signal file.

# 1/14/2013

We are back to reading baselines from separate files. Apparently, operators find it useful to split an experiment into multiple event files while still referring to the same baseline.