Splitting, Cleaning, and Spikesorting
Post-Recording Tools For The 128 Channel DAQ2 Recorder
by
Dale Shuman

This document provides instructions on how to use a set of tools to process recordings made by the 128 channel DAQ2 recording system.

Some existing tools have been modified and new ones created. In general, modified utilities retain the same name as the older ones but have a "2" added to the name. The existing utilities remain available as legacy tools for processing older files. The new utilities do not have a "2" as part of the name since there never was a "1" version.

The tools include:

split_all.sh	Split recordings into individual channel files
daq2_split	Program that does the actual splitting, called by split_all.sh, can be used standalone
make_chan.sh	Make the chanlist files used for cleaning, clean_rec.sh needs these
clean_rec.sh	Clean all of the chan files for a recording
make_label.sh	Make the label file that the spikesorter requires
do_clean_data.sh	Workhorse script, called by clean_rec.sh, can be used standalone
do_clean_data2.m	Octave script used to clean chan files, called by clean_rec.sh
do_noclean_data.sh	Script to process chan files that do not need to be cleaned

This document assumes you have created one or more experiment recordings. Instructions for running the daq2 recording software are provided in a separate document.

One of the fundamental assumptions is that the initial name of the recording files conforms to a fixed format. Many of the down-stream tools expect this and will not work correctly if this is not the case. The term "basename" is used throughout this document to refer to this. The basename format is:

YYYY-MM-DD

which is year-month-day.

Additional fields are added and removed to the basename at various stages in the processing.

A good way to explain this is to use a specific basename as an example. Assume you have a set of three recordings for 2012-02-21:

2012-02-21_001_1-64.daq
2012-02-21_001_65-128.daq

2012-02-21_002_1-64.daq
2012-02-21_002_65-128.daq

2012-02-21_003_1-64.daq
2012-02-21_003_65-128.daq

The recording files have the recording number and channel fields added to the basename.

SPLIT THE FILES

- ☐ In your directory of choice, typically in /raid/datamax on cisc3, create a dir with the base name of 2012-02-21. Copy the .daq files to this dir by your preferred method.
- ☐ Open a command line window and cd to the dir with the .daq files.
- ☐ Run the split_all.sh script to split the .daq file(s) into separate channels.

In our example, you you would type this:

```
split_all.sh  001 002 003
```

This will create split dirs if they do not exist and then split the data to separate chan files. Some user feedback is sent to the console, such as percent complete.

If you prefer to do this manually and have more control over the split operation, you can invoke the program that the script uses. If, for example, you just wanted to extract channel 23 from the 001 recording, you would type this:

```
daq2_split 2012-02-21_001_1-64 23
```

This command will print help information if you type in the program name with no arguments.

After the script completes, you will have three new dirs with individual chan files in them. In our example, you will have:

```
split.001/
split.002/
split.003/
```

A typical file in split.001 would be: 2012-02-21_001_r_45.chan. The _r_ field means this is a raw file that has not been processed.

The daq2_split command is backwards compatible with .daq files from earlier recordsings. These did not have the 1-64 and 65-128 fields. If these fields are missing, it assumes that it is dealing with a 1-64 channel file.

CLEAN THE FILES

Once we have the individual chan files, we have to prepare them for spikesorting. Some files require running through a cleaning process that removes noise from the files. Others require no processing and are ready for the spikesorting stage.

- ☐ Before cleaning, you need to create the initial channel list files for each recording that the cleaner uses. To do this using our example files, in the same directory as the .daq files, type:

  make_chan.sh 001 002 003

This creates these files, with default values:

chanlist_001_1
chanlist_001_2
chanlist_001_3
chanlist_001_4
chanlist_001_5
nocleanlist_001

chanlist_002_1
chanlist_002_2
chanlist_002_3
chanlist_002_4
chanlist_002_5
nocleanlist_002

and so on.

The clean lists chanlist_*_6 and chanlist_*_7 are not created by this script since the corresponding array does not exist yet. The code to do this is in the script, but has been commented out.

You may want to edit these to exclude some channels or you may want to delete some of them completely if you know you will not need them. Note that you have lists for each recording, so you can alter them without affecting processing for other recordings.

Some channels should not be cleaned. These are in the nocleanlist_* files. "Cleaning" for them means they are copied from the split dir to the clean dir.

Each channel file listed in the chanlist files must exist. The cleaner software will exit with an error if it cannot find a file.

Each chanlist file should contain a single line of channel numbers and two additional numbers that will be the name of files the cleaner creates. The default chanlist_001_1 file is this:

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 199 198

The last two items will contain pass one and pass two noise data that was extracted from a raw file to create a clean file. The convention is that the chanlist file number is the prefix for the 99 and 98, so, for example, in the chanlist3 file, they are 399 and 398.

- ☐ To process all of the chan files, you need to run the clean script for each recording. In our example, to clean the 001 recordings, you would type:

  clean_rec.sh 001

The script looks for the chanlist* files and starts a cleaning process in the background for each set of channels in the chanlist files. The script expects to find chanlist1-chanlist7 files, but will not consider it an error if any of these files do not exist.

These run concurrently and the OS will take advantage of multiple cores to keep all of the CPUs busy. Output is sent to a log file named after the recording number and the channel list, such as chanlist_001_1.log. You can view updates to one of these in real time by typing:

```
tail -f chanlist_001_1.log
```

The files in the nocleanlist file are copied as-is to the destination directory. Since it seems that the split directories are generally deleted later, it seems safer to decouple the split and clean dirs and not have the contents of one depend on the contents of the other.

Once the 001 recordings have been cleaned, you can move on to the next one and type:

```
clean_rec.sh 002
```

After that completes, type:

```
clean_rec.sh 003
```

It is possible to create a script that would iterate on a set of recording numbers. This is probably counterproductive because cleaning is a very CPU-intensive operation and if you give the CPUs too much work, they spend a lot of time switching between the jobs. You almost certainly want to wait for the "clean_rec.sh 001" command to finish before starting the next one. On the other hand, if you just want to start a cleaning process and let it run unattended for a few hours, you can issue all three clean_all_sh commands in sequence.

If the clean_rec.sh script does more than you want, you can get a copy of this from /usr/local/bin and edit your local copy to only do a subset of the operations. You can also use this file to cut and paste commands to the command line if you want to clean a single channel by hand. The clean_rec.sh script calls do_clean_data.sh, which in turn runs an Octave script. The Octive script will create dirs if they do not exist, so you don't have to do that by hand. If you use variations of the clean-rec.sh script, don't forget to type:

```
./clean_rec.sh 001
./clean_rec.sh 002
./clean_rec.sh 003
```

to run your local copy.

The cleaning operations in our example will create three new dirs:

```
clean.001/
clean.002/
clean.003/
```

A typical file in clean.001/ would be: 2012-02-21_001_45.chan.

For large files, the cleaning can take a long time. You can view progress by viewing the .log files. For example, to follow the progress of the cleaning of chanlist1, type:

```
tail -f chanlist_001_1.log
```

SPIKESORTING

Once we have clean files, it is time to spikesort. It requires an input file in the directory where the .chan files reside. So, if you want to spikesort clean.001, go to a command line window, cd to the clean.001 dir and type:

- make_label.sh

This script will look for a .chan file in this dir and if it finds one, it extracts the basename and creates a label file from this. If there are no .chan files, the script prints an error message. In our example, the file is:

2012-02-21_001.lbl.

You can also type "make_label.sh some_other_name" to create a label file with a name you supply. Note that if this name does not conform to the naming conventions that the spikesorter expects, it may not be able to find .chan files unless you also rename them.

As this point, the recording data files and label files are in formats that the down-stream tools expect.

To start spikesorting for recording 001, run spikesort_control_panel, giving the complete path and name of the .lbl file without an extension, like so:

- spikesort_control_panel /raid/datamax/2012-02-21_001/clean.001/2012-02-21_001

Consult the spikesorting documentation for more information on spikesorting.

THE SHORT FORM

- ☐ 1. Create YYYY-MM-DD dir.
- ☐ 2. Copy .daq files to it.
- ☐ 3. Start a command shell and cd to dir with .daq files.
- ☐ 4. type: split_all.sh 001 002 003
- ☐ 5. type: make_chan.sh 001 002 003
- ☐ 6. Edit or delete chanlist* files as required
- ☐ 7. type: clean_rec.sh 001, wait for completion (or not)
  type: clean_rec.sh 002, wait for completion (or not)
  type: clean_rec.sh 003, wait for completion
- ☐ 8. cd to clean.001 dir
- ☐ 9. type: make_lbl.sh
- ☐ 10. edit .lbl file as required.
- ☐ 11. spikesort_control_panel /raid/datamax/YYYY-MM-DD/clean.001/YYYY-MM-DD_001
- ☐ 12. Go to 8 for 002, 003, etc.

While the order of some of the operations is fixed, they apply to a single recording. You could, for example, do all of the steps for just recording 001, and then repeat the entire procedure for each subsequent recording, or do all of the splitting first, then clean and spikesort 001, then repeat for 002, etc.