MoNA SpecTcl Guide

A. Ratkiewicz & W. A. Peters
National Superconducting Cyclotron Laboratory

February 13, 2006

1 Introduction

This guide is intended to give you an idea of how to use SpecTcl to analyze data that MoNA has collected. This guide is written for the analysis of data taken from cosmic rays, thus this may not be useful to you. However, it should give you an idea of what's involved in a fairly simple application of SpecTcl, which is all it's intended to do. Please read the NSCL documentation and general user guide [1] to familiarize yourself with the basic commands; especially the `spectrum` and `gate` commands.

1.1 MoNA Parameters

MoNA has, for the TDC's and QDC's, 576 raw and 576 calibrated parameters from the PMT's. These are then used to calculate a few parameters for each of the 144 bars:

```
Xpos, Tmean, Qmean
```

Which are, in turn, used to calculate the *Hit* parameters.

1.1.1 Hit Parameters

These parameters are not as useful as one would like since the code is written to scan the whole MoNA array (starting with A0 to I15) and labels the first event with two valid TDC channels as hit 1, and so on.

```
X_hit, Y_hit, Z_hit, ToF_hit, Theta_hit, Phi_hit
```

Pseudo parameters have been made to calculate the time-ordered hits[2] and should be used for more advanced analysis.

2 Configuration files

There are plenty of files sourced by the SpecTcl code when it begins. Many are not to be adjusted in any way because they contain long lists of MoNA variable names and special `tcl` operations that are used by the underlying `cpp` code. In general, those files will get copied to your work space and never need to be adjusted or edited. There are a few files that contain calibrated variables that get set once for each experiment and then a few that contain adjustable variables to set as you see fit to best analyze the data for a particular run number.

2.0.2 SpecTclRC.tcl Make sure that the directory that contains the `spectcl.scr` file to launch SpecTcl also contains a `SpecTclRC.tcl` file and is edited to source the SpecTcl housed in a `/user/mona/mona/spectcl_shared` directory. You may also set your `WorkDirectory` and `EventDirectory` at the end of this file. The `SpecTcl_Driver.tcl` file is sourced from this file. The `Driver` runs the `Tk` graphical user interface that displays the buttons and such we call the `SpecTcl Control Window`.

2.1 MoNA Config Directory

The `~/mona/config` directory contains many of the files that control the configuration settings for SpecTcl. The most import ones are:

2.1.1 MoNA_setup_run.tcl This file contains many permanent lists and a few fitting flags and variables. The most important parts are the TDC and QDC setup lists that get read into the scriptable SpecTcl code to configure the Readout and SpecTcl codes. It also has the CFD lists of addresses and configuration file names for all 18 CFD's.

The top of the file has variables that can be edited for each experiment. Here we have a series of fitting flags:

```
QDC_thres_flag
QDCfitted
TDCfitted
Xposfitted
Tmean_indie_offset
```

These are set to `"true"` after the appropriate calibration or fitting code is completed[3].

The `tmean_offset` variable is a global time offset for all `Tmean` parameters of all bars and should be set from a MoNA gamma timing run or to some reasonable value to start (like 333.0 ns).

The `MoNA_Z_pos` variable should be set to the distance from the center of the reaction target position to the center of MoNA bar A8 (in centimeters).

The `MoNA_hits` variable is read in to create that many hit parametes and spectra. The default is 20 because the time-sorting scripts[2] use 20 hit parameters as inputs into the pseudos.

2.1.2 MoNA_hardware_run.tcl This file sources the `MoNA_setup_run.tcl` file and creates the scriptable data packets and commands to configure the VME modules (TDC's, QDC's and, Scaler's). Different QDC threshold lists are sourced if the `QDC_thre_flag` is set to `"true"`. This file should not need adjusting unless the MoNA hardware changes (like adding neutron cans).

2.1.3 MoNA_dynamic_var_run.tcl This file sources all the default variable settings and then the calibrated or fitted ones if the appropriate flags are set in `MoNA_setup_run.tcl`. This file also creates all the non-raw MoNA parameters from large lists that are defined in `MoNA_Param_run.tcl`. It does not need editing.

2.1.4 MoNA_spectcl_run.tcl This file is directly sourced by the `.scr` script file that launches SpecTcl. It's main function is to define the data packets to unpack in SpecTcl that are defined in the hardware file, for example:

```
unpack add mona
unpack add bitpattern
```

It does this by first sourcing the `MoNA_hardware_run.tcl` file and the `MoNA_dynamic_variables` files. It also sources the `MoNA_Param_run.tcl` file to create all the raw parameters from it's lists. This file is replaced by `Tandem_spectcl.tcl` for the Tandem version of SpecTcl[4].

2.1.5 MoNA_readout_run.tcl

This file is similar to `MoNA_spectcl_run.tcl` but tells the Readout code which packets to fill, for example:

```
readout add mona
readout add bitpattern
```

3 Running SpecTcl

There must be a SpecTcl executable created that can be run from your user account or an experimental account. These executables have been traditionally housed in the `/user/mona/spectcl_shared` directory and accessed through the local `~/mona/daq/spectcl` script files that source the local `~/mona/config` directory filled with relevant settings and calibration factors.

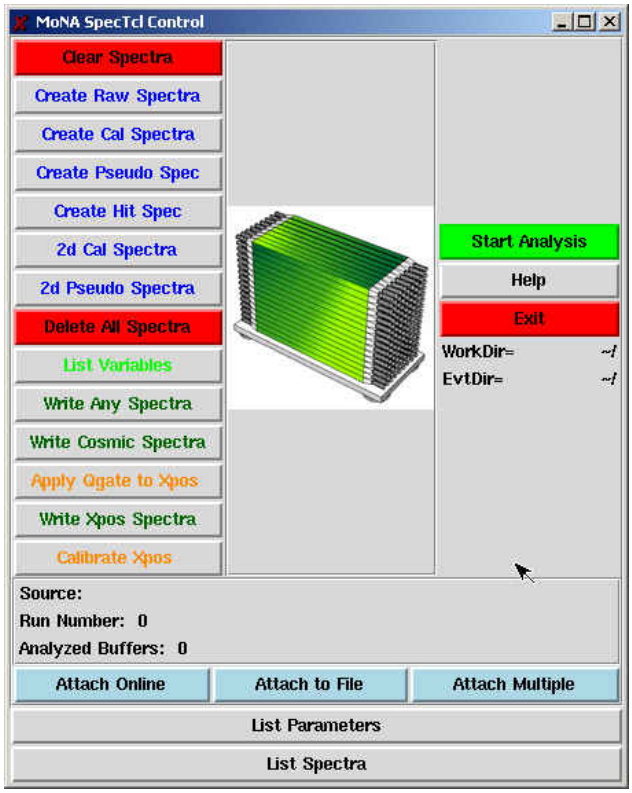
3.1 Open SpecTcl

Open SpecTcl by either clicking the button on a DAQ machine display of a current experimental account, or executing the script by typing:

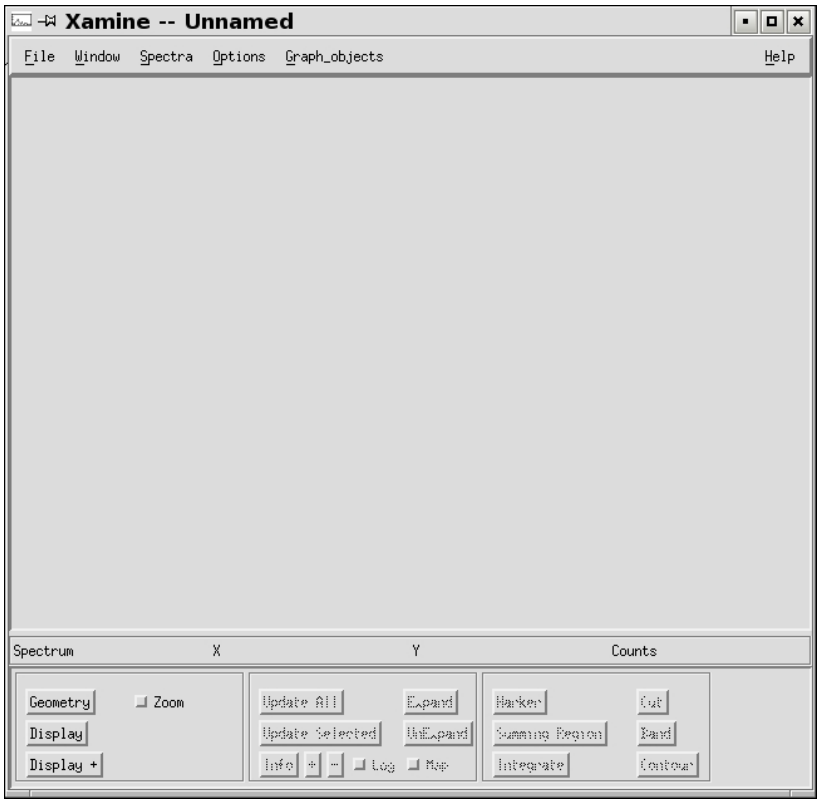
```
> ./run_spectcl.scr
```

or the equivalent script file in the `~/mona/daq/spectcl` directory. You will see the following windows:

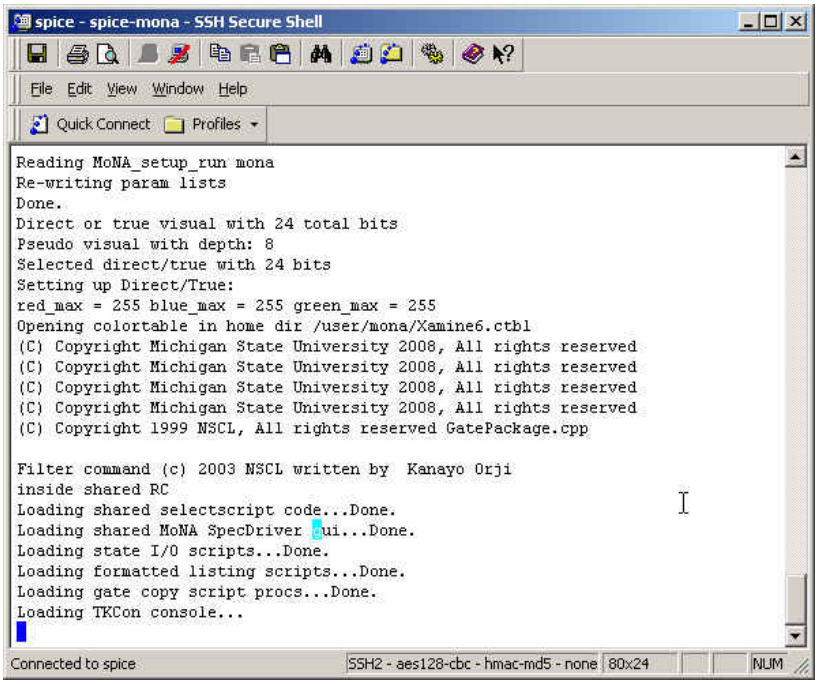
- The MoNA SpecTcl Control window:



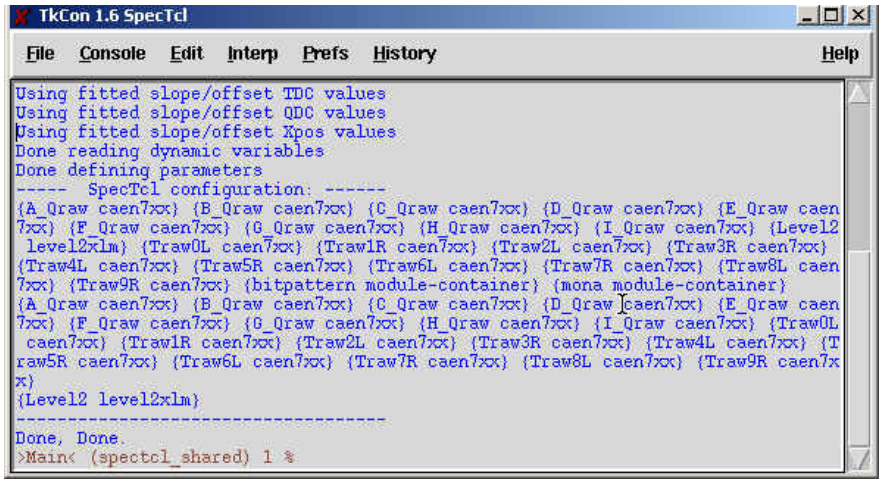
- The Xamine window:



- The Console Shell:



- The TkCon window:



The TkCon window should have listed many configuration settings including which fittings flags are set and whether or not you are using calibrated slope/offset values for QDC, TDC, Tmean, and Xpos parameters. This listing of settings finishes with `Done,Done`. The code also lists the data packets it recognizes like maybe `Level2` (when the `bitpattern` data packet is used).

Look at the right side of the SpecTcl Control window and find the `WorkDir` and `EvtDir`. The first should be set to the current working/spectcl directory, usually `~/mona/daq/spectcl`. The second should be set to the directory containing the event files you wish to analyze (maybe `~/stagearea/complete`). If they are not set, or set incorrectly, set them using the TkCon window:

```
% set WorkDirectory ~/mona/daq/spectcl
% set EventDirectory ~/stagearea/complete
```

3.2 Creating Spectra

Use the `spectrum` in-line command in the TkCon window to create spectrum from any parameters. To view a list of the parameters type:

```
% parameter -list
```

Or press the *List Parameters* button on the bottom of the SpecTcl control window. If you forget the syntax just type:

```
% spectrum
```

and the error message will display the syntax it's looking for.

When you've made a spectrum of a parameter, it must be bound to SpecTcl to view on Xamine, so type the command:

```
% sbind -all
```

There are a few pre-set buttons to create and bind many of the commonly used spectra types. These buttons are on the left side of the SpecTcl control window. For viewing the `X` position of each bar, you need to create position spectra; do this by clicking the *Create Pseudo Spec* button on the SpecTcl Control window. After you've done this, you can press the *List Spectra* button to list all the current spectra loaded into SpecTcl and verify that spectra with the group name `Xpos` exist.

3.3 Configuring Xamine

In the Xamine window, click on the *Geometry* button in the lower left hand corner of the screen. Change the configuration of the resulting pop-up dialogue to 4x4. Then click the *Display* + button and use the *Apply* command in the pop-up window to add desired spectra to the Xamine cells. Using the *Okay* command will close the pop-up window after inserting the selected spectra.

You can save any Xamine window configurations by selecting the `Write Configuration` from the `Window` menu on top. be sure to save it as a `.win` file in the `/win` directory. You may also read in previously saved configurations, but you must be sure the included spectra have been bound to SpecTcl first. For this example, select the configuration file `\pseudo_win\Xpos_A.win` to display all the Xpos spectra for layer A.

3.4 Attaching Data

In the SpecTcl Control window, click on *Attach to File* and select a run to read in. Once this is done, SpecTcl will begin analyzing the data. Let it run for a short while so that you have an idea where the peaks and valleys in your data will be.

If you do not need to read in the whole event file, and when you've got an idea where the peaks and valleys in your data are, click on the *Stop Analysis* button.

3.5 Setting Up Gates

Back in Xamine, double click on first cell (the cell in the upper left-hand corner of the window) to zoom in on it. This should be Xpos_A0. Now click the *Cut* button in the lower right hand corner. Here, we are going to tell SpecTcl that we want gate on events that hit in a certain part of the tube. For this example choose (for this bar) the region between -80.0 cm and -60.0 cm. Move the cursor to -80 cm on the spectrum in Xamine and click once, then move to -60 cm and click again. This will fill the coordinates into the appropriate box in the *Cut* pop-up window. Select the *name* box and label the gate, `gate1` and then select *Ok* to accept. Cuts can also be declared with the `gate` in-line command in the Tk Con window (see Ref[1] for details).

MoNA is set into a right-handed coordinate system with positive z in the direction of the moving beam and positive y as the up direction (toward ceiling). This means positive x is to the left when looking at the front of MoNA (layer A). Xamine displays spectra from lower value to higher value, so the range `{{-150 150 301}}` will have the correct coordinates, but left and right will be reversed on the screen. Just image you are viewing the Xpos of a bar from behind MoNA (where spdaq16 sits). This way positive x is to the right of the viewer since you are looking in the negative z direction.

In the TkCon shell, type:

```
% gate -list
```

to verify that the gate has been created. Note that if you've made a mistake in gate definitions, you can use the `gate -delete gatename` command to zero out the gate. This command does not remove the gate from SpecTcl, it just gets rid of all its arguments. This is handy; it allows you to change gates at the last minute without changing much else.

Now do the same thing for the last cell (Xpos_A15) in Xamine as you did for the first, but use different points for the cut (use the positive value for the second, since you're trying to get a feel for how the ray travels through MoNA). We will gate the middle bars to see the cosmic rays travel from top left to bottom right through layer A.

Finally, you need to set up an `and` gate. If you called the first gate for Xpos_A0 `gate1` and the one for Xpos_A15 `gate2`, then to create an `and` gate going to the TkCon shell and typing:

```
%  gate -new andgate * {gate1 gate2}
```

Here, `gate -new andgate` tells SpecTcl that you are making a new gate called `andgate`, the `*` tells it that the gate is a logical `and`, with `{gate1 gate2}` as the arguments, so that the gate is only true if both `gate1` and `gate2` are true. This is the only way to apply more than one condition on a spectrum. An `and` gate using `*` can depend on many other gates; just list all the ones you want to include inside the `{}` brackets.

3.6 Copying Spectra

Now that you've got your gates set up, you want to make a copy of all the spectra you'll be applying them to. This is mainly for safety reasons; you don't want to change your source data by gating it. Go to the TkCon shell and type:

```
%  spectrum -new xpos_a0_gate1 1 {Xpos_A0} {{-150 150 301}}
```

Here, you're making a new spectrum named `xpos_a0_gate1` from parameter `Xpos_A0` with a range of -150 to +150 cm and a resolution of 301 bins.[1] Repeat this for `gate2` using the name `xpos_a15_gate2` and for the middle 14 spectra (Xpos_A1 to Xpos_A15) with `_andgate` in the name because we will apply the new `andgate` to them all. Next you need to bind them using the `sbind -all` command(as noted above).

3.7 Displaying New Spectra

In the Xamine window, use the *Display+* button to view all the new spectra you've just bound. At this point, it's a good idea to save your configuration, name it something unique, and save it. You've just saved the setup that's currently in Xamine; you have not saved the spectra names or the gates, so if you've done a lot of work or feel like you might want to revisit the these settings for this experiment, now might be a good time to go to the TkCon shell, press h, and copy/paste the result to a text file (See Section 5).

3.8 Applying Gates

Go the TkCon shell and type:

```
% apply gate1 xpos_a0_gate1
```

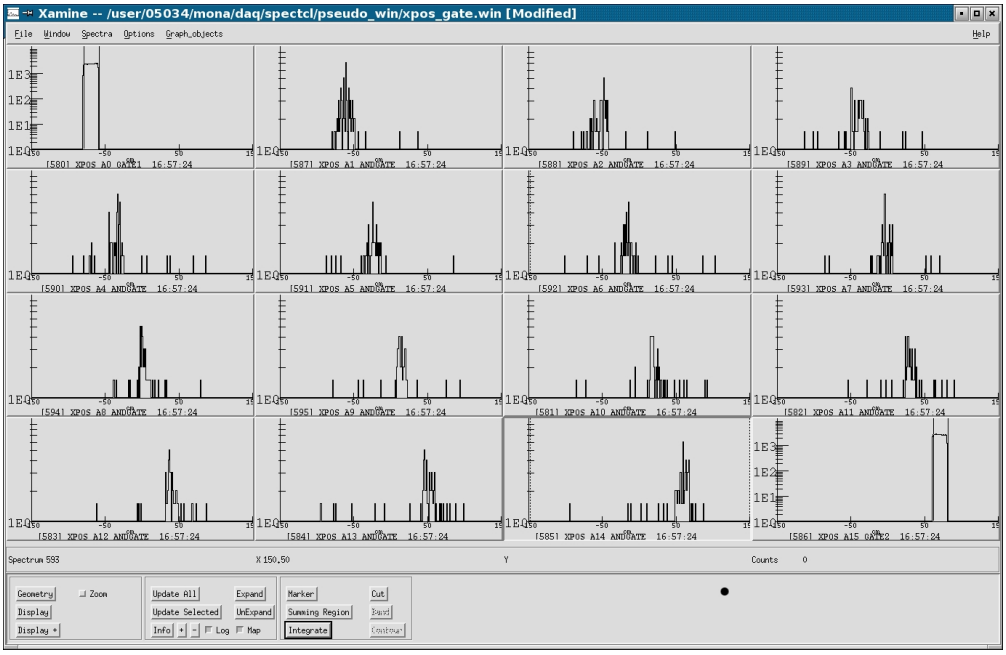
to tell SpecTcl that you're applying `gate1` to the spectrum called `xpos_a0_gate1`. Now repeat to apply `gate2` to the `xpos_a15_gate2` spectrum. The `apply` command can have a whole list of arguments so, to apply the `andgate` to all the middle 14 Xpos spectra you just made, you only need one line:

```
% apply andgate xpos_a1_andgate xpos_a2_andgate ...
```

[1] It is nice to select a bin number that is a multiple of the range you've selected plus one so the bin edges line up with the values.

4 Analyzing Data

Go to the SpecTcl Control window, click on the red *Clear Spectra* button. [2] Now re-attach your data file. SpecTcl will start analyzing the data. Now's a good time for a coffee break; a typical run can have over 1,000,000 buffers to analyze. When it's done, the Spectra will look something like this:



Notice the peaks seem to move from one side to the other as you look at successive bars. Most of these event are muons that pass through the left side of bar A15 through the right side of bar A0. So the overall pattern will look like a diagonal path through layer A.

4.1 Exporting Data

When SpecTcl has finished analyzing the data, you might want to export it for later use. Go to the SpecTcl Control Window and click the *Write Any Spectra* button on the left. Select the spectra you would like to export, and don't forget to give it a unique name. The default format is `ascii` which works well with Excel. You may select any format you want to use. This will then save the `swrite` file in the `spectra` directory.

4.2 Importing into Excel

Open *Excel*, click on `Data`, go to `Get External Data`, and select `Import Text File`. Change the file-type option to `all files (*.*)`, and select your `ascii` file. In the first frame, just click *next*. In the following box, check both the `space` and `other` delimiter boxes. In the `other` box type a closed parentheses `)`. This will turn every space and parentheses read in by Excel into a new column space, which makes life much easier for you. Now click finish and select the first cell to start the import.

Next, if you've only imported one spectrum per Excel sheet and you want to automatically turn these columns into calibrated and compressed columns, you must first set your macros security to medium. Do

[2] Note that it's very important to clear spectra whenever you make a change to the spectra or the gates.

this by selecting the `tools -> options -> security -> macro security` menus and then setting the macro security to medium. Now open the `/projects/proj1/mona/docs/spectcl-read-macro_calibrate.xls` Excel file that has the correct macros in it, and select the *Enable Macros* option. Once this file is opened in Excel go back to your data sheet and you can run the macro `test2` on your imported data. This will create many new columns and fill any missing zero's from the original histogram. One column fills with the calibrated values and then the following ones are the same data re-histogramed with `x2` and higher compression factors.

5 Simplifying

5.1 Settings Files

Now you can repeat the same gates and commands you just did for layer B. However, you're not going to do all that tedious typing again. Instead, you're going to put the bulk of your commands in a text file, and source the file from the TkCon shell. This will save you time if we ever want to revisit the analysis.

Open a text editor (use Kwrite or emacs). Emacs is prefered by people with a strong Unix background. If you are not entirely comfortable with it, use Kwrite; Kwrite's a lot like notepad, so you shouldn't have any problems.

When your text editor opens, go to the TkCon window, type `h` and then highlight the commands you used to create and apply the gates for layer A. Now select *Copy* from the TkCon *Edit* menu. In the editor select *Paste*, then edit the commands for layer B instead of layer A. You must delete the line numbers that the `h` command inserts in front of the lines. Don't forget to type `sbind -all` at the end of the file if you create any spectra within. This is also a good place to save the in-line commands to create the gates you use (they must be created before they can be applied). Save the file in the `settings` directory as a `.tcl` file. To source the file and run the commands go back in the TkCon window, type:

```
%  source settings\filename.tcl
```

This will read the file and run the commands between each `<return>` as if typed one at a time as an in-line command into the TkCon Window. If there is a syntax error in the file, the process breaks and all the commands after the error are not read in. In that case, just re-edit the file, comment out any commands before the error, and source it again.

5.2 Creating Custom Pseudos

In the course of your analysis, you may find that you need to create a spectrum that's a calculation of two or more calibrated parameters (or any calculation not done within SpecTcl already). This technique will work for any reasonable arithmetic operation. A `pseudo` parameter is really just a procedure implemented while filling the SpecTcl parameters. Now you can finally tell SpecTcl to do whatever it was you wanted it to do. It can output any value you choose into a new parameter that can be put into a spectrum like the regular parameters. Three commands are involved:

5.2.1 Parameter First, you need to define a new parameter. It's a good idea to pick a high number for the id (higher then 5000), as this command won't work properly and will cause problems down the road if you assign an id that's already in use.

5.2.2 Pseudo Now you need to define the pseudo. The syntax here is a bit more complex:

```
%  pseudo name {parameter1 parameter2 ...}
  {if {$parameter1isValid && $parameter2isValid ...}
      {return [expr($parameter1 * $parameter2)/$parameter2] }
      else {return -1}}
```

Note that the name of the pseudo should match the name of the parameter associated with it that you just made.

The syntax here is not intuitive if you're not experienced in TclTk, so let's take a minute to discuss it. First, note the `{parameter1 parameter2 ...}` part of the command. Remember when we made copies of existing spectra (section 3.6), we had to tell SpecTcl what we were copying? As you might suspect, this is the same idea. Be warned, though, any parameter you want to use in the pseudo calculation must be declared here.

The next part of the pseudo command is an `if{}` statement, followed by the expression to return if `true` and the expression to return if `false`.

The `{if {$parameter1isValid && $parameter2isValid ...}` in the example is a logical `and`; all parts must be true in order for the conditional to be true. This particular statement wants valid parameters; the $ sign indicates that we're looking at the value of the parameter, and `isValid` demands that there be something assigned to it (note that this is case sensitive, so pay close attention to your declarations).

If the terms of the conditional are met, then the `{return [expr ($parameter1 * $parameter2)/$parameter2] }` expression will be evaluated. Note that we're telling SpecTcl to return the value of an expression `[expr ...]`. Again we insert a $ in front of the parameter name to use it's value.

Notice that if the conditions of the if statement are not satisified, some other number is returned. You should make this value marginally outside the spectrum you're defining, so you don't see bad data in your spectrum.

5.2.3 Spectrum Finally, you need to make a new spectrum to display the pseudo parameter you've just defined. The syntax is identical to creating a spectrum for a gated spectrum, as above, just use the pseudo parameter's name. The command is:

```
%  spectrum -new name 1 {pseudo_name} {{-150 150 301}}
```

Some very useful pseudo scripts have been written by A. Ratkiewicz[2]. One sorts all the MoNA hits by time, within a set neutron-time gate. Another calculates the angles and velocities between internal MoNA hits (like between the first and second hit) to help with multiple neutron analysis.

5.3 Filtered Data

SpecTcl allows one to filter out only selected parameters that can then be read in to a different SpecTcl quicker than re-reading all the raw parameters and processing them. This is useful if you have a large set of data runs that take a long time to read, and if you've calibrated all your parameters and are only concerned

with a small subset of the total parameters made. Since MoNA has over 1500 standard parameters (not including the hit parameters), it may be wise to crate a filtered file for the data. This process requires a special SpecTcl code that uses the filtered file as an input, and has the filtered parameters defined. For more information about this see Ref[1].

5.4 Useful Tips

- In a unix shell, or the TkCon window, the `up arrow` key will display the last command. pressing the up arrow twice will display the command before the last, etc.
- In a unix shell, or the TkCon window, typing `h` will give you a history of past commands.
- Typing the command with no arguments will display an error message illustrating the expected syntax, for example: `% gate`.

References

[1] R. Fox, *SpecTcl - User's Guide* October 28, 2003. NSCL. http://docs.nscl.msu.edu/daq/spectcl/users_guide.htm

[2] A. Ratkiewicz, W.A. Peters, *Time Sorting Pseudos* 2005. NSCL. `/projects/proj1/mona/reports`

[3] J. Miller, M. Strongman, L. Elliott, D.B. Hecksel, M.M. Kleber, P.J. Voss, T. Pike, R. Pepin, A. Ratkiewicz, W.A. Peters, *MoNA Calibration* 2005. NSCL. `/projects/proj1/mona/reports`

[4] W.A. Peters, *Tandem SpecTcl Guide* 2006. NSCL. `/projects/proj1/mona/reports`