


Branch: master ▾ [E12014Planning](#) / [e15507](#) / [helpDocumentation.md](#)[Find file](#) [Copy path](#) [anthoak13](#) Update helpDocumentation.md

93267f2 yesterday

[1 contributor](#)

166 lines (109 sloc) 9.36 KB

[Raw](#) [Blame](#) [History](#)   

# Manual for E15507

---

- [Running the DAQ](#)
  - [Readout Shell](#)
  - [SpecTcl](#)
  - [Scalers](#)
  - [Signal Switcher](#)
  - [PulserGUI](#)
  - [HV Control](#)
  - [Hornet Control](#)
  - [ELog](#)
  - [Venting, pressure, and PanelMate](#)
- [FAQs](#)
  - [Firefox not working](#)
  - [Restarting the DAQ](#)
  - [Problems with slow controls](#)
    - [Refused socket connection](#)

## Running the DAQ

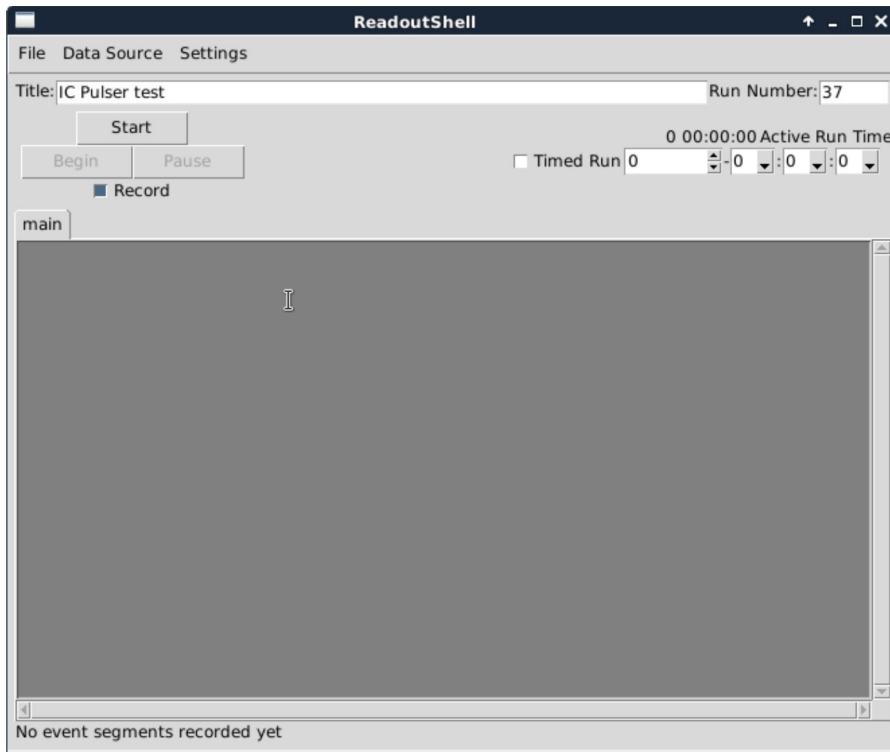
---

The DAQ is setup to be run from any computer on the DAQ network, including U3PC's and any spdaq computer. The general recipe for getting the DAQ up is:

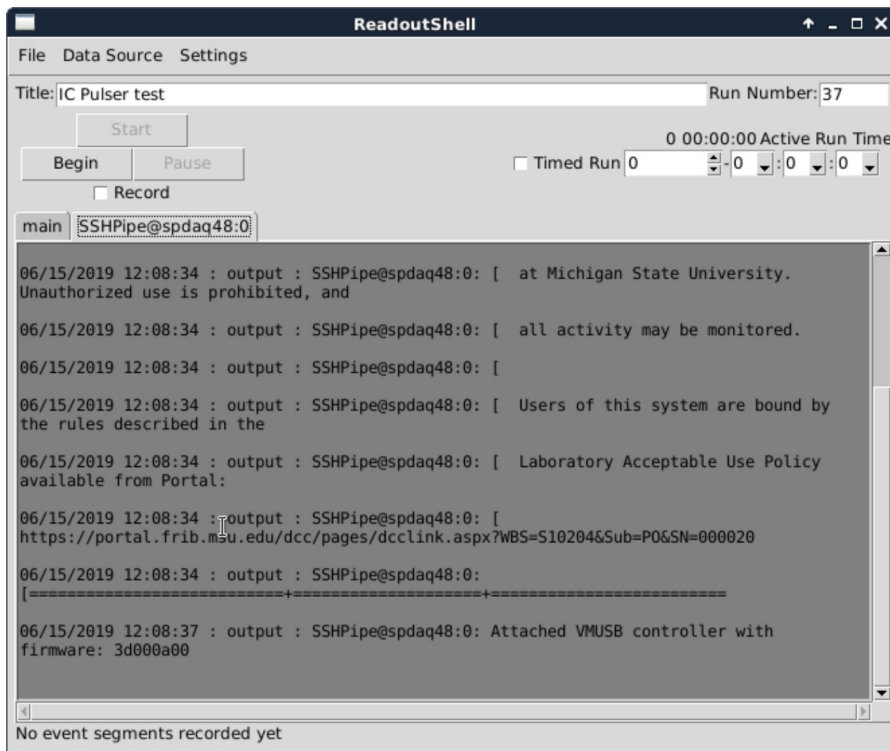
1. Run [Readoutshell](#)
2. Run and connect [SpecTcl](#) to the ring buffer
3. Take data with [Readoutshell](#)

### Readout shell

From the home directory of account e15507, run `./ReadoutShell`. This should open a window like



Then press start and you should see the VME crate attach like



If this doesn't happen, make sure the VME crate is on and look at the section for [restarting the DAQ](#).

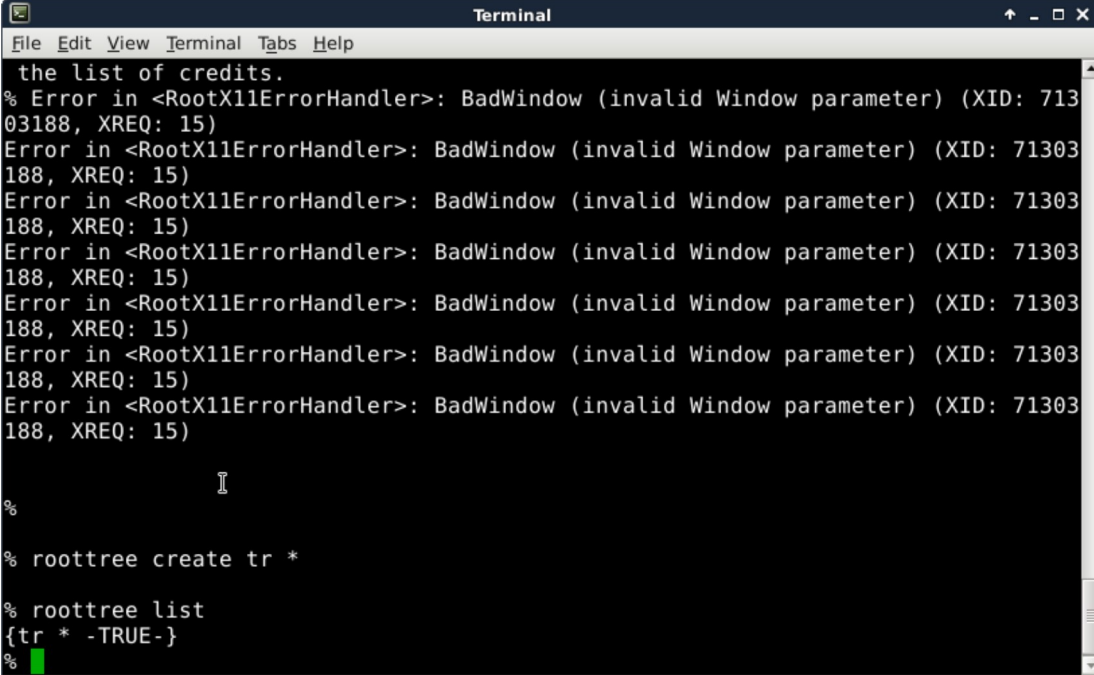
You are now ready to take data. If you want to record data, make sure the record button is ticked. You can verify data is coming in by running `./dumper` from the home directory. This just outputs everything being read in by the DAQ. You can also pass valid dumper command arguments through this wrapped. For example `./dumper --count 10` will only output the first 10 items. A full list of valid arguments can be found with `man dumper`.

## SpecTcl

Right now, TTree generation is not completely automated in SpecTcl. This means there is an additional step that has to be done after each startup.

### Launching SpecTcl

To launch SpecTcl, `cd specTcl` and then `./SpecTcl`. It will take a few seconds for it to fully load. After it is fully loaded, go to the terminal you used to open SpecTcl and type the command `roottree create tr *`. You can verify it worked with the command `roottree list`. For those curious, you can find the documentation for this command [here].(<http://docs.nslc.msu.edu/daq/newsite/spectcl-5.0/cmdref/r3059.html>) At this point that command line should look something like



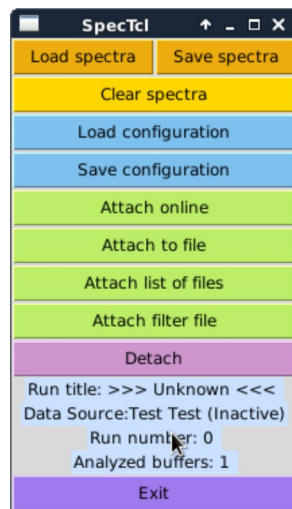
```

the list of credits.
% Error in <RootX11ErrorHandler>: BadWindow (invalid Window parameter) (XID: 71303188, XREQ: 15)
Error in <RootX11ErrorHandler>: BadWindow (invalid Window parameter) (XID: 71303188, XREQ: 15)
Error in <RootX11ErrorHandler>: BadWindow (invalid Window parameter) (XID: 71303188, XREQ: 15)
Error in <RootX11ErrorHandler>: BadWindow (invalid Window parameter) (XID: 71303188, XREQ: 15)
Error in <RootX11ErrorHandler>: BadWindow (invalid Window parameter) (XID: 71303188, XREQ: 15)
Error in <RootX11ErrorHandler>: BadWindow (invalid Window parameter) (XID: 71303188, XREQ: 15)
Error in <RootX11ErrorHandler>: BadWindow (invalid Window parameter) (XID: 71303188, XREQ: 15)
Error in <RootX11ErrorHandler>: BadWindow (invalid Window parameter) (XID: 71303188, XREQ: 15)
Error in <RootX11ErrorHandler>: BadWindow (invalid Window parameter) (XID: 71303188, XREQ: 15)
Error in <RootX11ErrorHandler>: BadWindow (invalid Window parameter) (XID: 71303188, XREQ: 15)
%
%
% roottree create tr *
% roottree list
{tr * -TRUE-}
%

```

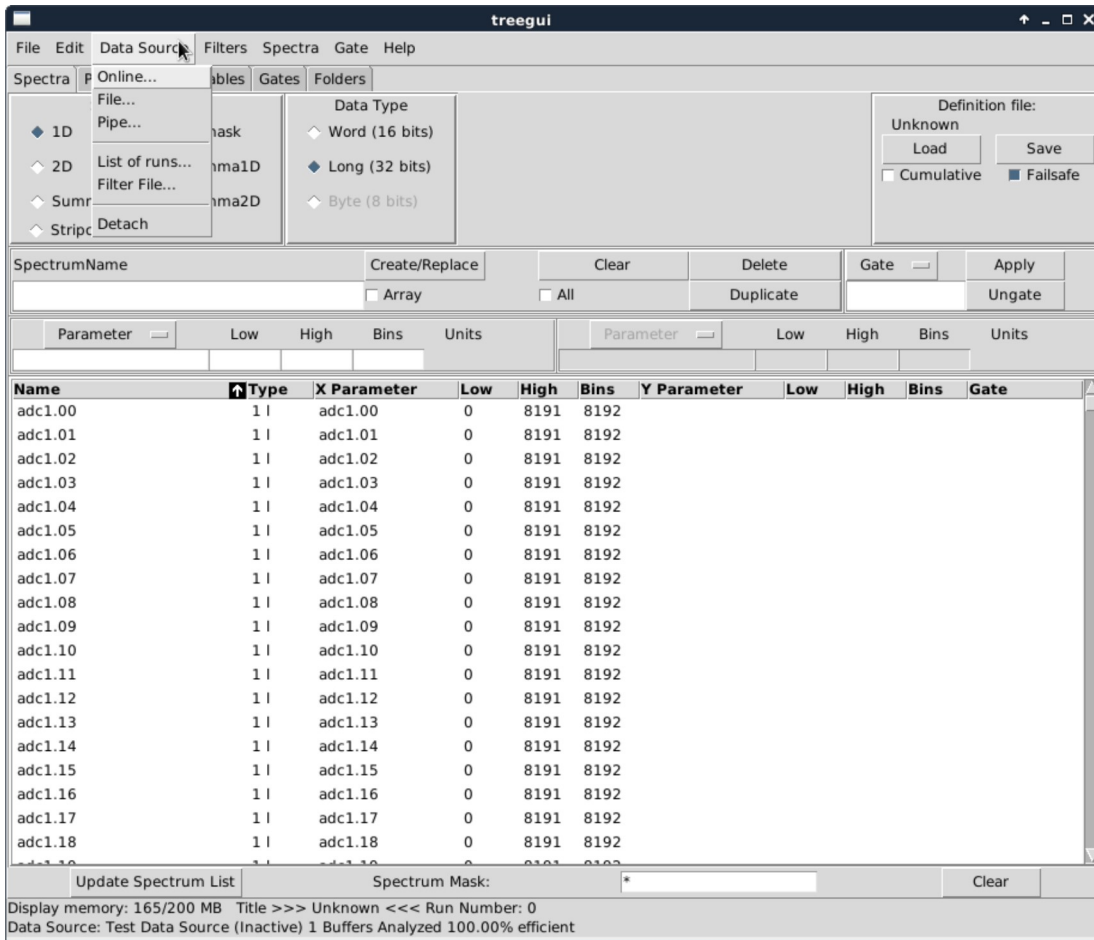
### Closing SpecTcl

To close SpecTcl, you should always use the purple **Exit** button on the bottom of this window

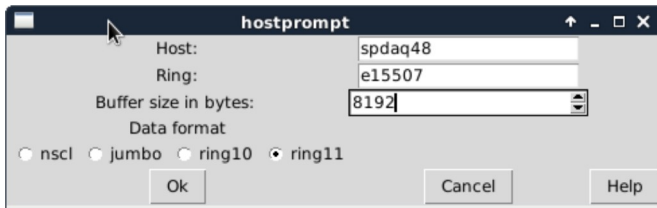


### Using SpecTcl

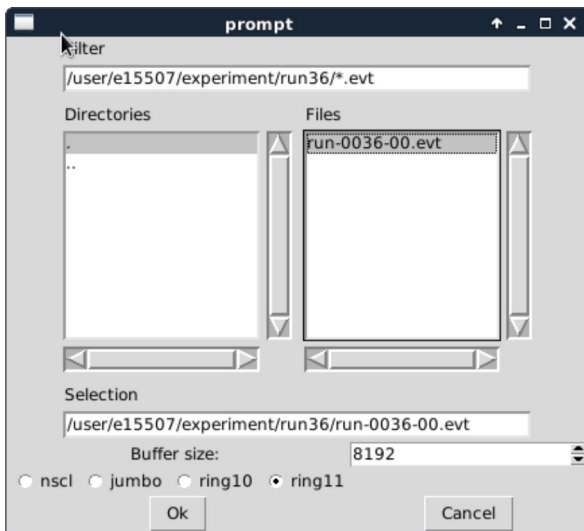
In order for SpecTcl to be of any use, you have to attach a data source. Usually, this will be an online data source (a ringbuffer) or a .evt file. The data source is selected from the data source tab. Make sure that ring11 is selected and the buffer size is set to 8192.



When connecting online, make sure the settings match those below



For offline, the .evt files are located in ~/experiment/run#/



Data should now be coming in. At list point, you will want to load in and def-files, making sure Cumulative is checked. Now window files can be

Every parameter defined in SpecTcl will be written to a TTree with the name `run-#.root` in `~/SpecTcl` . After each run, this should be moved out of this directory to the analysis directory with

```
scp ~/SpecTcl/*.root e15507@fishtank:/mnt/analysis/e15507/SpecTcl1TTree/
rm ~/SpecTcl/*.root
```

In order for a TTree to be generated with online data, SpecTcl must be connected to the data source before the run starts. Otherwise it misses the `start_run` ringbuffer item.

## Scalers

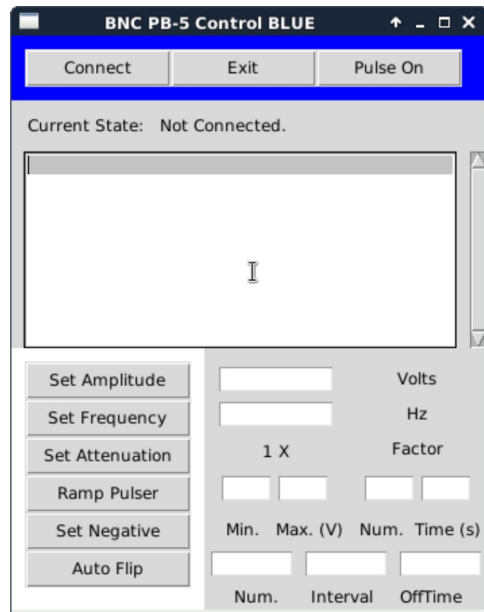
Coming soon!

## Signal Switcher

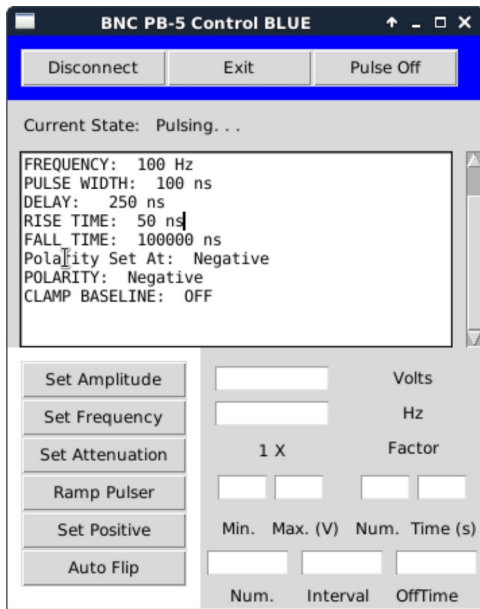
Coming soon!

## PulserGUI

To run the pulser GUI `cd ~/pulser` and run `wish pulserGUI_BLUE_fission.tcl` . You should see a window like



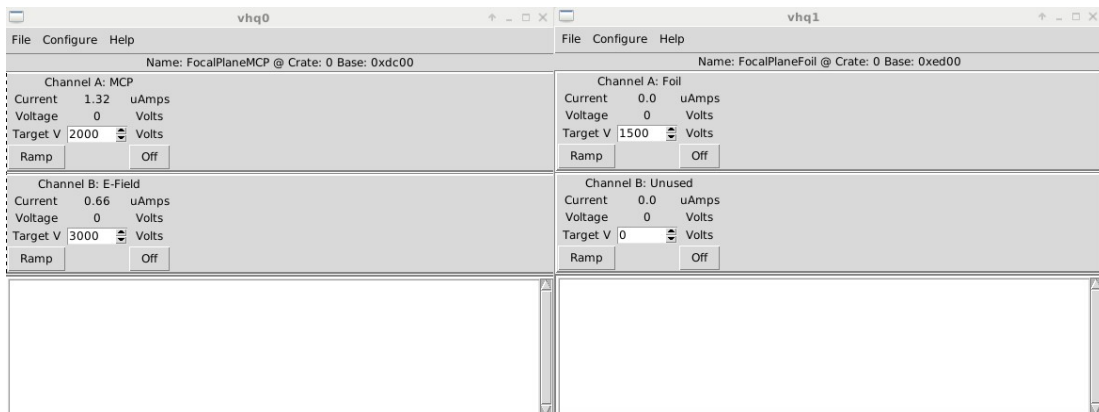
To connect to the pulser press the **Connect** button, and then push **Pulse On** to start the pulser. The window should look like this now



## HV control

The HV control GUI used is a modified version of the standard NSCLDAQ vhgControl. It was changed to add more descriptive channel names. This program works through communication over the fiber optic interface between the DAQ computer and the VME crate. This means that the program has to be run on the DAQ computer that is physically attached to the VME crate. The launch scripts `~/goHVUS` and `~/goHVDS` handle this all for you. The actual program being called is `~/HVcontrol/vhgControl` with the appropriate configuration scripts passed. All of the configuration scripts are located in `~/HVcontrol` if they need to be modified. DO NOT try to change the configuration scripts through the GUI interface!

To launch the voltage control of the focal plane MCP, run `~/goHVUS`. You should see 2 windows pop up that look like



The downstream HV control works the same way as the focal plane HV control, but there are three windows. The additional one is for control of the ion chamber bias.

## Hornet Control

To launch the ion gauge control run `~/goHornet`. This launches a Tcl GUI written by Adam to monitor the pressure in the MCP and Ion Chamber sections of the chamber. When it launches, you should see a small window with two tabs. It seems to like to crash after being open for a bit, if that happens just close it and reopen it

If the IC is on, it will display and update the pressure every two seconds and look like



If the ion gauge is off, then you should see something like



The **On/Off** button toggles the power status. It will take up to 15 seconds to turn on, so be patient. You can also read the power status and any error from the status bar below the pressure reading.

## ELog

Coming soon!

## PanelMate Control

Coming soon!

## FAQs

---

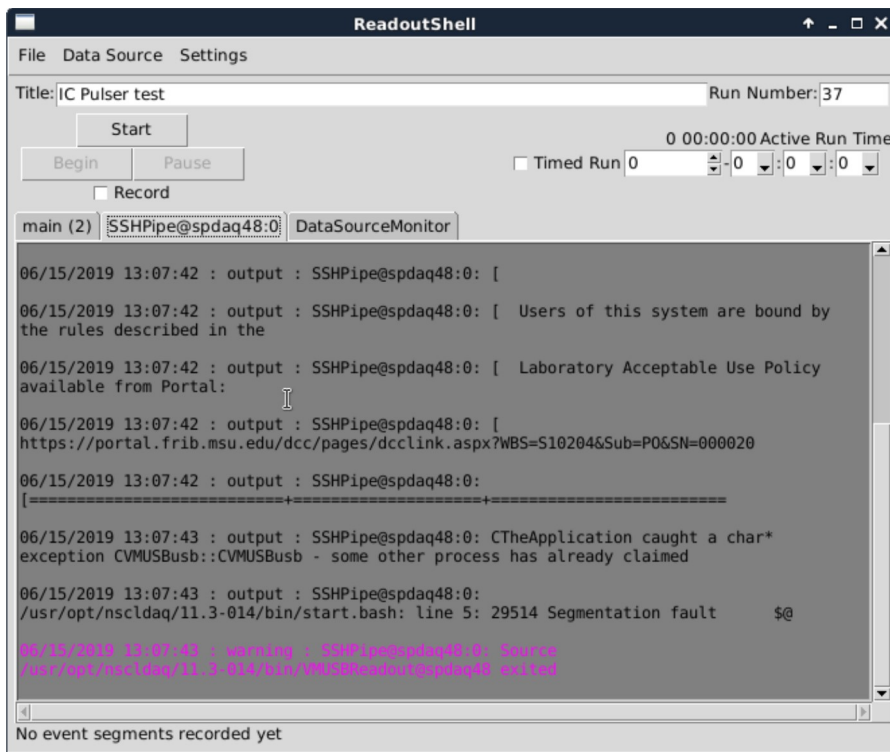
Below is a list of common problems I've encountered and how to fix them. As we find more problems, I'll add them to the list

### Firefox

Because all of the computers share a home directory, I've found Firefox can often screw up its locks. To fix this problem `cd ~/.mozilla/firefox` and delete the files `lock` and `.parentlock` from the profile directory. I've found that the right profile to try is `2rbkukgx.Kyle`.

### Restarting DAQ

If you see something like this



then you'll need to, probably, fully restart the DAQ. This often occurs when there is already a process registered as the producer for the e15507 ringbuffer. Either because the DAQ crashed, or there is a readoutshell open somewhere else connected to ringbuffer e15507.

To restart the DAQ, start by `~/GoSpdaq`. Then type `ringbuffer status`. You should see some output like

```
<spdaq48:~ >ringbuffer status
+-----+-----+-----+-----+-----+-----+-----+-----+
|Name  |data-size(k)|free(k)|max_consumers|producer|maxget(k)|minget(k)|client|clientdata(k)|
+-----+-----+-----+-----+-----+-----+-----+-----+
|e15507|8194        |8167   |100           |29294  |26       |26       |-    |-            |
|-     |-         |-     |-           |-     |-       |-       |29308|26          |
+-----+-----+-----+-----+-----+-----+-----+-----+
```

Kill the process that is registered as the producer `kill 29294` and delete the ringbuffer `ringbuffer delete e15507`. You should then be able to use ReadoutShell to attach to the ringbuffer.

## Slow Controls

### Refused socket connection

Most problems with the slow controls stem from another copy of the program being open elsewhere. This is difficult to track down if you do not know where the program was opened last. This is particularly true of the PuslerGUI and Hornet Control as they both connect to a terminal server that only allows one connection at a time. When you try to connect to the terminal server if there is an existing open connection you will see an error like

```
<u3pc2:~ > ./goHornet
Error in startup script: couldn't open socket: connection refused
  while executing
  "socket $terminalName.nsl.msui.edu 2001"
   (procedure "start" line 6)
   invoked from within
  "start $server"
   (file "/user/e15507/hornetIG/hornetGUI.tcl" line 47)
```



The challenge is tracking down the open process. The best way I've found is to ssh into each computer it might be open on (s2pc2, u3pc2, u3pc3) and kill any offending processes. You can find potential process IDs with the command `ps aux | grep wish`. This should fix the problem.