Scaler 2 Operations Manual

Overview

Each scaler board in the system consists of a PCB scaler board and a corresponding Linux receiver machine. The PCB scaler boards are housed in standard PC boxes and powered via the PCI slot but do not use the PCI bus besides powering the board. Currently, boards 1-3 are housed in one box and boards 4-6 are housed in another box. The status of the Scaler 2 system can be monitored at :

https://online.star.bnl.gov/L2TimingPlots/scaler2mon/scaler2mon.html

Scaler Boards

Each scaler board is controlled via its ConnectCore Ethernet interface. To control a board, ssh into the corresponding board at hostname "scalerbdX.scaler.bnl.local" using the "root" account. Note that the boards are connected to the "scaler.bnl.local" network and connection must come from another machine on that network. The "root" account must be used because the control code is run as a system process.

A process runs on each board that communicates between the FPGA and the STAR run control system. When this code starts, it reprograms the FPGA from its flash memory so if the board gets into a bad state, restarting this process will initialize the FPGA and control code to its startup state. While logged into the corresponding scaler board, run `ps` and look for the `sca_control_node` process. To kill this process run `kill -9 <PID>` where <PID> comes from the `ps` command. To restart the process, run `/etc/init.d/S97scaler_control`. It will take about 30 seconds to reprogram the FPGA and reinitialize all registers. The code will report `Setup Done` when configuration is complete. After the code starts and is running, it will print a short series of messages every 10 seconds that indicate the reading and writing of registers. These messages are related to the monitoring code and indicate that the control software is running correctly. Note that AFTER a board is reprogrammed, the Linux receiver code should be restarted also, as described in the following section.

Note that direct communication with STAR run control comes from a central process running on the "scaler48" machine so restarting an individual board will not affect the operation of a current STAR data-taking run but the corresponding board will no longer be configured or running until the STAR run is restarted. This allows scaler boards to be reconfigured without affecting STAR data-taking.

If errors are seen on the monitoring page for a particular board, or "Dropped Blocks" or "Data Stream Errors" are seen in the receiver section for a board, the FPGA should be reprogramed via the procedure described in this section and then the Linux receiver process should be restarted as described in the next section.

Linux Receivers

Data is streamed from each scaler board to its corresponding Linux receiver machine via gigabit Ethernet. Streamed data is stored locally on each receiver machine and is then histogrammed at the end of each STAR run. To control a Linux receiver machine, ssh into the corresponding machine at hostname "scalerrcvX.scaler.bnl.local" using the standard "trg" account. Note that these machines are also on the "scaler.bnl.local" network.

To connect to the control process, run `su` to become "root" and then run `screen –x`. Messages related to the operation of this receiver machine will stream while connected to this process. To stop the receiver code, press `ctrl-c` while connected to this process. To restart the receiver process, make sure you are "root" and run `systemctl start scaler_receiver.service`. No message will be printed when this control code is started but the process can be monitored using `screen –x` as described above.

Note that AFTER the FPGA on a board has been reprogrammed (described in the previous section), the receiver code should be restarted as described in this section.

Also note that as in the case of the boards themselves, direct communication with STAR run control comes from a central process running on the "scaler48" machine so restarting the code on an individual receiver machine will not affect the operation of a current STAR data-taking run but that the corresponding receiver will no longer be configured or running until the STAR run is restarted. This allows the receiver machine code to be restarted without affecting STAR data-taking.

Network Power System

All scaler boards and Linux receivers are powered via network-connected power supplies and can be controlled remotely.

Scaler boards and receivers 1-3 are connected to "nps2.scaler.bnl.local" which uses the password "!nps2" (no account name) and can be connected to via telnet.

Scaler boards and receivers 4-6 are connected to "nps1.scaler.bnl.local" which uses the username "apc" and password "apc" and can also be connected to via telnet.

In addition to network-connected power supplies, each Linux receiver machine can be controlled via its IPMI interface. Sometimes when a machine is powered up, it doesn't start automatically and needs a push of its physical power button. This IPMI interface is a way to do that remotely, as well as directly monitor the console output from the machine and respond if it gets stuck in its startup procedure.

These IPMI interfaces can be controlled by starting a web browser from a machine that is connected to the "scaler.bnl.local" network and going to the address "http://scaleripmiX.scaler.bnl.local/". The IPMI interface can be logged into using username "ADMIN" and password "ADMIN". Note that the IPMI interface does not currently work on all machines because not all of the machines are the same and some do not support this feature. The IPMI interface is also broken on one of the receiver machines.

Configuration and Run Control

All configuration and direct communication with STAR run control is done on the "scaler48" machine. This machine is connected to the "trg.bnl.local" network so that it can communicate with run control and to the "scaler.bnl.local" network so that it can communicate with the scaler boards and receiver machines. The standard "trg" account is used on this machine.

A central controlling process runs on "scaler48" called "scaler_control_master". This process runs as "trg" and the output can by monitored by running `` `screen -x` `` when logged into the "trg" account. This process communicates directly with STAR run control so if there is a problem with the "scaler48" node when taking STAR runs, this process should be inspected. When this process receives a command from run control (ie CONFIG, RUN_START, RUN_STOP, etc), it reads the Scaler configuration file and takes care of configuring each of the scaler boards and receiver machines in the system. The individual boards and receivers don't communicate with STAR run control directly but only indirectly through this process. If there is a problem with one of the boards or receivers, this process will time out quickly on that board and continue on to the others so any problems with individual boards or receivers won't hang up STAR data-taking.

This master controller process starts automatically on system boot but can also be stopped and started by running :

```
`systemctl stop scaler_controller.service`
`systemctl start scaler_controller.service`
```

The configuration file for the Scaler2 system is located at :

"~trg/code/scaler2/cfg/scaler_config.txt"

This file contains configuration settings for each board as well as "mode", "coarse delay", and "fine delay" settings for each channel of each board. It is re-read at each run configuration time and is archived for each run. In this file, the first few lines configure the board and then there is a line for each channel. For the channel configuration line, the first field is "SC_MDP" to identify that it is a channel setting. After this identifier, there are four numbers separated by white space. The first is the channel number (starting from "0"). The second is the "mode" setting (0 = level-triggered, 1 = edge-triggered). The third is the "coarse delay". The fourth is the "fine delay".

The master control process also contains a "backdoor" mechanism for stopping, re-configuring, and starting JUST the scaler system during a STAR run without affecting the data-taking run. This is used to make scanning delay settings easier while setting up the system at the beginning of a new year. The two main commands used in this procedure are :

```
`~trg/code/scaler2/bin/local_config_start`
`~trg/code/scaler2/bin/local_force_stop`
```

Running `` `local_force_stop` `` will communicate with the master control process which will "stop" each of the scaler boards and receivers without interrupting the STAR run. Running `` `local_config_start <local_run_number>` `` will re-read the configuration file and re-configure each of the scaler boards and receivers using the supplied "local_run_number". Note that the run number supplied is NOT the STAR run number but a local run number to use for this "local" run. While using this feature to scan delay settings, it is useful to use a "local_run_number" that is very different from the STAR run numbering system to avoid

confusion. Standard procedure has been to use a "local_run_number" following the format YYYYXXX where YYYY is the year and XXX just counts the runs used (ie 001, 002, 003, etc) but if it is not important to keep the data, a dummy "local_run_number" can be used repeatedly such as "999".

Note that this local "config_start"/"stop" procedure is only able to change register settings on the boards but it cannot change the actual STAR trigger system Run_Stop signal which goes to all the scaler boards as well as the rest of the full STAR trigger system. Therefore, when doing a delay scan for the scaler system using this "local" start/stop procedure, a STAR run must be in progress (ie trigger system in "run_mode") throughout the entire scan so that the scaler boards will run after they receive a local "start" command.

Also, this local start/stop procedure is done in software via the master control process to start/stop boards so they will not be in sync with each other or the rest of the STAR trigger system in regards to the number of RHIC ticks since the start of the run. Hence, the "RS Count" on the monitoring page will not be uniform for all boards and the number of counts in the data files will vary depending on when the local run was started/stopped.

The use of this local start/stop procedure will be described more in the following section discussing timing-in bits.

Timing-In Bits

To collect data to time-in bits, the procedure is as follows :

1. Make a copy of the scaler configuration file on "scaler48" to use for delay scanning. Soft link this file to "scaler_config.txt" so that it will be used when configuring the boards. To start, leave all "coarse_delay" settings the same as in the original file and set the "fine_delay" setting to "0" for all bits to be timed-in. Note that any number of bits can be scanned at once as long as the delay setting is changed for each one in this file. For the "mode" setting, a value of "0" (level-triggered) should be used for logic-level bits (ie bits coming from the output of QTs or DSMs). The procedure for discriminator bits will be described later in this section.
2. Start a STAR run so that the trigger system is in "run mode"
3. On "scaler48", in one window connect to the master control process by running `` `screen -x` `` from the "trg" account. This window is used to observe when the local run starts/stops.
4. Run `` `local_force_stop` `` as described previously to stop the scaler system. Watch the master control process window and wait until the run has "stopped". The STAR system will continue to run even though the scaler system is locally stopped.
5. Run `` `local_config_start <local_run_number>` `` as described previously to re-configure and start the scaler system using the "local_run_number" with the current "fine delay" setting. Watch the master control process window and wait until the local run has "started".
6. Let the system run for 5-10 seconds to collect scaler data for this delay setting.
7. Change the "fine_delay" setting in the configuration file to the next scan value (ie 1,2,3,...)

8. Return to step 4 to stop/start a local run with the new fine delay setting. Note that the STAR system should run continuously throughout this part of the procedure.
9. After data has been collected for all fine delay settings, run `local_force_stop` to stop the scaler system for this final scan run. The STAR run control system can be stopped at this point.

This procedure should be taken WITH BEAM so that all bits are being exercised as they will be used during regular data-taking.

At this point, there will be data files for each of the "local_run_number" runs which each correspond to one of the "fine_delay" settings.

For logic-level bits (ie coming from the output of QTs or DSMs), the analysis procedure is as follows :

1. After reading the header, read the data chunk for each "channel" (ie each distinct set of bit combinations) in the histrogrammed data file.
2. Keeping track for each of the 32 bits, if that bit is on, increment the count for that bit based on the "count" for that "channel". Don't increment the count if the bit is "off"
3. Also keep track of the total number of RHIC ticks based on the "count" for each "channel".
4. After reading all "channels" in the file, use the information from above to determine the frequency that each bit fired (ie Counts for the channel / Total RHIC ticks in the run).
5. Repeat this procedure for each of the "fine delay" settings

If you look at the frequency for the bit being timed in for each of the fine delay settings, most frequencies will be relatively close over most of the settings but will have one or two fine delay settings where the frequency is drastically different. This is where the "edge" is. The different frequency could be much higher or lower but the main thing to look for is a very different value. Depending on what the bit is, the frequency may change slightly over the scan of the delays if, for example, it has something to do with beam luminosity but the drastically different frequencies should still be evident. For the "fine delay" setting, choose a value that is far from the edge.

To determine the "coarse delay" setting, use the data file from the previously determined "fine delay" setting and look at the frequency for the bit in question for each of the 0-119 bunch crossings. The abort gaps should be apparent and should be aligned using the "coarse delay" setting to correspond with the rest of the system.

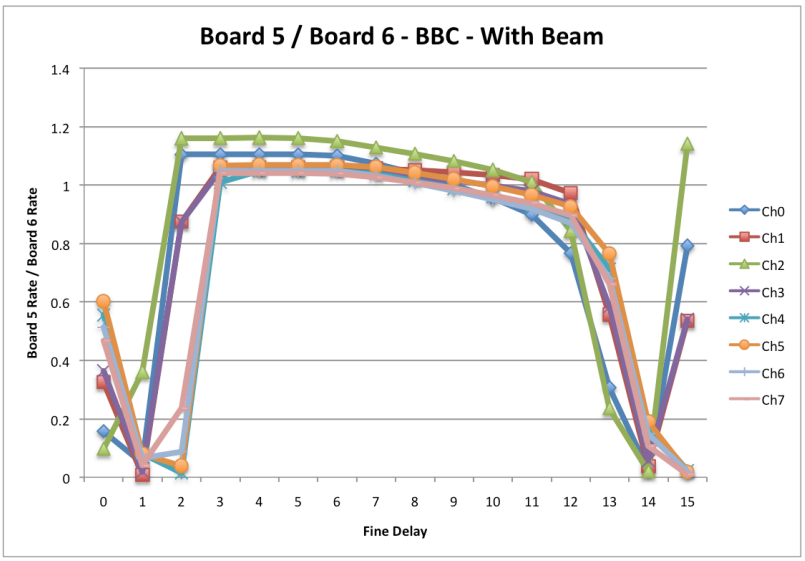
Note that some scaler bits are not simply lone bits indicating a hit but may indicate a channel number or may be combined with other hits (ie ZDC-SMD). For these bits, a special procedure may be needed to find hits which may use more than one bit in the data but the general procedure is the same, ie counting the frequency for something and looking for the edge/abort gaps.

The procedure for timing-in bits from discriminator channels is slightly different. The general procedure is to route the discriminator signals to TWO different scaler boards by re-programming the RAT board. When following the procedure above for taking the "fine delay" scan data, set the "mode" setting on one of the boards to "1" to use edge-triggering. Set the

“mode” delay on the other board to “0” to use level-triggering. The “fine delay” should then be scanned on the board set to “0” (level-triggering). When the board is set to edge-triggering, the “fine-delay” is not used and thus should not be scanned for that board.

The general procedure for analyzing the scan data for the discriminator channels is to look at the relative frequency between the level-triggered board and the edge-triggered board for each “fine delay” setting. The general theory is that the edge-triggered board will catch “all” of the hits while the level-triggered board will catch some fraction of the hits depending on where the “fine delay” setting is set. Looking at the relative frequencies between the level-triggered board vs edge-triggered board will show where the “edge” is for these channels.

An example is shown below of a plot of the relative frequency between level-triggered channels vs edge-triggered channels for all fine delay settings. This plot is for BBC discriminator channels.



In the final configuration file, a “fine delay” setting should be chosen after the edge where the ratio is stable but not too long after the edge so that all pulses are caught. In the above plot, a fine delay setting of “6” was used but a slightly smaller fine delay setting could also be used.

The reason that the level-triggered channels catch more hits than the edge-triggered channels (ie the ratio is > 1) is not fully known but may have to do with late hits or noise in the system. It was determined that the ratio goes above “1” because the edge-triggered channels are missing some hits as opposed to the level-triggered channels counting too many hits. The plot for these channels is consistent every year, however, and the fine delay setting was originally chosen at a point on the curve to match the “old” scaler boards when the transition was made to the “new” scaler boards. The ratio for VPD discriminator channels is much cleaner and plateaus out at close to exactly “1”.

The procedure for determining the “coarse delay” setting for the discriminator channels is the same as for the logic-level channels once a “fine delay” setting is chosen.

Troubleshooting

On the Scaler2 monitoring page, if the “FIFO Overrun” is set for a board, or if there are non-zero “Dropped Blocks” or “Data Stream Errors”, the first (and usually only) thing to do is to first reprogram the scaler board FPGA and then restart the receiver process (both procedures described previously).

If there is a problem in the connection to STAR run control (ie scaler48 node is red or stops the run), the master control process should be checked on “scaler48”. Occasional problems have also been seen in previous runs when one of the NFS mounts gets corrupted on the “scaler48” machine.