AMORE Beam Test Documentation

Michael Phipps

Prior to this fall, our implementation of AMORE was designed and used primarily for monitoring and analysis of the MTS and small-scale detector tests. As our lab transitions to larger scale beam tests, the DAQ process poses extensive challenges and the readout code demanded an overhaul. The purpose of this documentation is to detail the challenges and implemented solutions for scaled-up, more general applications of AMORE. Pay special attention to the mapping sections. By making use of three types of inversions, the orientations of the APVs within particular planes and across different detectors can be easily synchronized during readout.

I. Monitoring

The monitoring process for beam tests is similar to the process performed in the lab at FIT. Raw data plots and higher level monitoring plots are checked at the beginning of each run and continuously throughout the duration of the run.

The biggest challenge to monitoring during the FNAL beam test was implementing the proper mapping and rewiring AMORE to handle multiple mapping types across detectors and within particular APVs – and to do so in a timely fashion. This process was not finished by the time we started the beam tests, and the only monitoring performed was with raw data plots.

For future beam tests, the current version of AMORE should be sufficient for most monitoring needs.

II. Mapping Overview

Mapping refers to the process of aligning the data from particular readout channels from the APV to physical strips on the readout board. This is done locally across each channel in the APV and then globally to arrange the channels from multiple APVs across each plane of our detectors.

The DAQ process begins with the individual copper strips on the readout board. From there, data is routed to individual pins on the Panasonic connector, and each pin is affiliated with a particular channel in the APV. Once the data arrives at the APV, it undergoes an analog multiplexing process that further scrambles the order of our initial strips. When it finally arrives in AMORE, we have to rearrange the order of the data to match the physical layout on the strips. Doing so is complicated, but the purpose of this note is to describe that process in detail.

Note: please see the Appendix for the C++ mapping implementation.

III. Mapping Configuration File

During the October beam test, there were four distinct mapping periods. The differences included: the order in which APVs were connected to ADC channels, the number of APVs assigned to particular detectors, the type of readout boards attached to detectors, the number of disconnected FEC channel and the number of APVs displaying NAN (not a number) data. To simplify the analysis process, we created four different configuration files, so that the only change made on a run-to-run basis was the name of the appropriate configuration file in the shell script. The files themselves are located within ~/amoreSRS/configFileDir, and have the naming scheme /mappingFNAL[date].cfg.

The first section in the config file defines the detectors used and the basic properties of each: number of planes, plane size, number of APVs, mapping type, local and global inversions, pitch size and number of strips. The next section contains the alignment shift parameters. The section after that lists each APV and assigns it to a particular detector plane and FEC channel number, as well as a column indicating whether that FEC channel number is active or inactive. Active or inactive, in this context, refers to FEC channels that are somehow empty. This can happen if a master APV is connected to the ADC port without a slave APV.

The following three sections explain local, global and FEC channel inversions. If you apply these inversions correctly, the shouldn't have to use more than one inversion for any APV or detector orientation.

IV. Local Mapping

The algorithm that routes a strip number on the readout board to a channel number of AMORE is essentially the same for readout boards of the same type. The difficulty comes when the readout board is installed on opposite sides of the GEM board or when the Panasonic connectors are installed on opposite planes of the readout board. These cause 180° inversions of the channels, either locally, globally or both.

A local inversion means that the channels in each APV are inverted. For example, if there are two APVs along one readout plane, the channels in each APV would receive a 127 - strip number local inversion (there are 127 + 1 channels in each APV). When you consider this globally across the entire plane this causes the strips in the center of the board (strip numbers 127 from APV 1 and 128 from APV 2) to be flipped so that they flank the edges of the plane (strip numbers 0 and 255).

An example is shown in Figure 1. In the initial configuration, the APVs in the Y plane are configured on the left side of the readout board. This causes APV channel 127 to be associated with strip number 0 (and so on). In order to correct this, all you need to do is designate this plane for a local inversion in the configuration file.

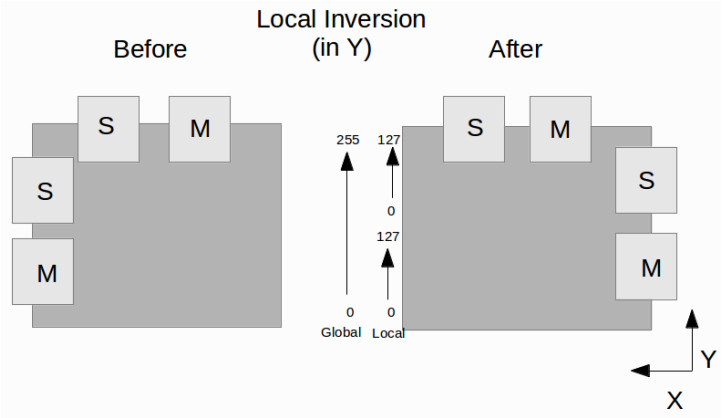

Figure 1: Local Inversion in Y. Master and slave APVs are labeled on each axis. Remember that both APVs are connected to the same ADC channel, with the master then routed to the first FEC channel and the slave to the second.

One important note: the current, default configuration in AMORE is a single local inversion. So a NO in the configuration file actually denotes one local inversion; a YES denotes two local

inversions. This is done because the 0 pin on the APV is on the right side of the chip. In order to make the left-most strip as 0, we need to locally invert the channels of the APV.

Within AMORE, these local corrections are implemented within SRSAPVEvent::StripNoCorrection. They are implemented within all DAQ processes: raw pedestal, pedestals and analysis. Within analysis you can find this by following SRSPublisher to SRSProcessor to SRSDetectorPlaneEvent:SetHitList to SRSAPVEvent::ComputeListOfAPVHits and finally to SRSAPVEvent::ComputeTimeBinCommonMode. Within the pedestal process it is found by following SRSPublisher to SRSPedestal to SRSAPVEvent::ComputePedestalData to SRSAPVEvent::ComputeTimebinCommonMode.

V. Global Mapping

A global inversion means that the channels across the entire plane are flipped. Using the previous example, the two strips in the center would merely flip positions under a global inversion (strip number 127 would become 128 and 128 would become 127). For a two APV detector plane, this amounts to a 255 - stripNo global inversion.

An example is shown in Figure 2. The problem in this configuration lies in the Y plane. The 0 channel of the master APV is oriented with the 255th strip on the board (and so on). In order to correct this, all we have to do is make a global inversion.

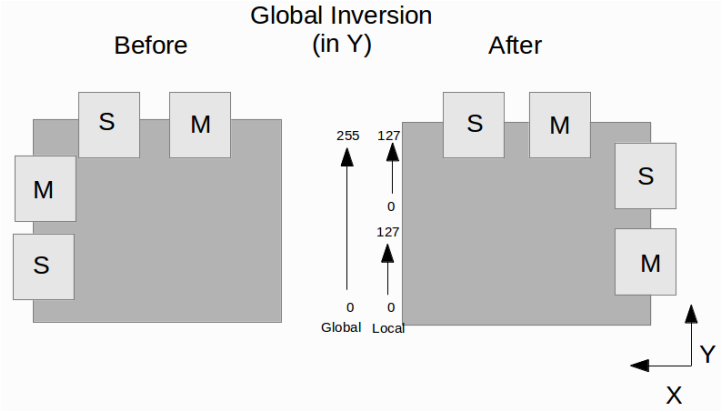

Figure 2: Global Inversion in Y. Master and slave APVs are labeled on each axis. Remember that both APVs are connected to the same ADC channel, with the master then routed to the first FEC channel and the slave to the second.

Within AMORE, global mapping is done in the SRSHit header file within the functions: SetStripNo() and ComputePosition(). Global inversions are made within the function: SetGlobalInversion().

Global mapping is done during all DAQ processes: raw pedestals, pedestals and analysis. But – and this is important – global inversions are not; they are only made in analysis, after pedestal subtraction, common mode suppression and zero suppression.

To find the strip number mapping, follow SRSPublisher to SRSProcessor to SRSDetectorPlaneEvent::SetHitList to SRSAPVEvent::ComputeListOfAPVHits. To find the positional mapping, follow SRSPublisher to SRSProcessor to SRSDetectorPlaneEvent::GetZeroSuppressionData to SRSCluster::CalculatePosition (or SRSCluster::CalculateCenter).

VI. FEC Channel Inversion

A FEC channel inversion is done by swapping the FEC channels of the master and slave APVs. Due to the routing from the ADC to the FEC, the master slave is always sent to the first FEC channel and the slave to the second. So if the only problem with your configuration is the order of the master and slave APVs on the readout board, all you have to do is swap their FEC channel number within the configuration file. In other words, if the master APV FEC channel is actually channel 0, change it to 1, and change the slave channel to 0. An example is shown in Figure 3.

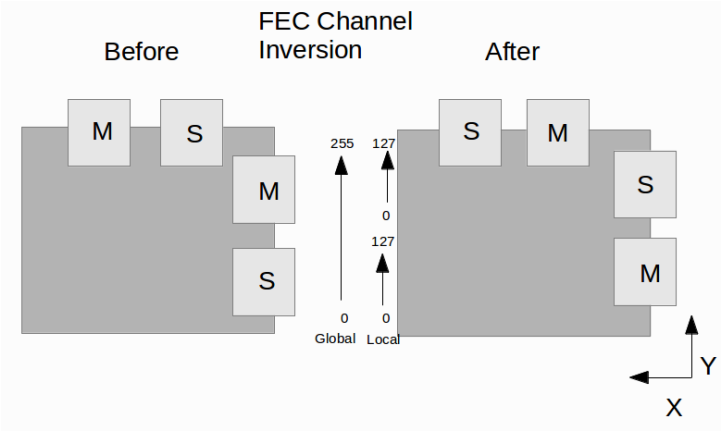

Figure 3: Inversion of the FEC Channels in X and Y. Master and slave APVs are labeled on each axis. Remember that both APVs are connected to the same ADC channel, with the master then routed to the first FEC channel and the slave to the second.

VII. Correlation Plots

To make sure your mapping is correct, you can produce correlation plots, in which the data from the same event of the same plane of different detectors is plotted against each other. A positive slope indicates consistent mapping across both planes; anything else is indicative of problems. An example is shown in the follow figure:

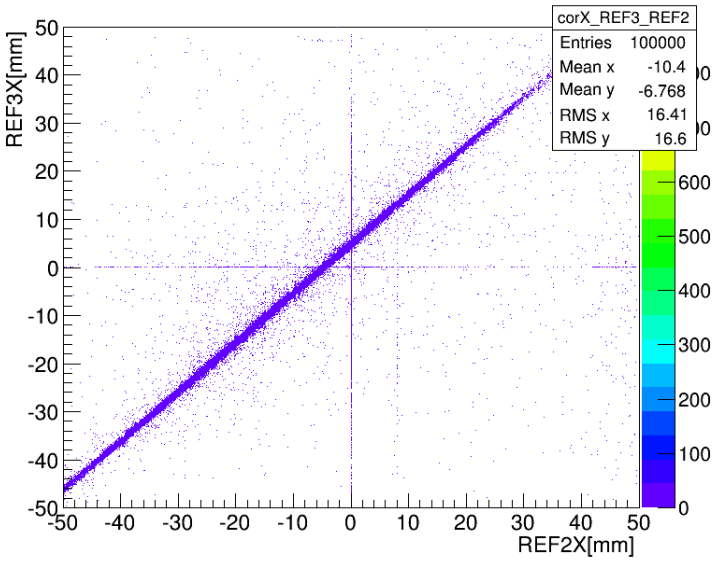

Figure 4: The position of events occuring in coincidence on both detectors are plotted against each other. The positive slope indicates consistent mapping.

VIII. Accommodating Empty FEC Channels

Our APV25 chips are typically grouped in pairs, with one set of channels set as the master and the other as the slave. The two APVs share a single HDMI cable that passes both sets of analog data to the ADC via a single input channel. The data is then digitized and broken up into two separate channels, one for each APV, and sent to the FEC for compression and storage. However, the new zigzag readout boards require fewer APVs than the cartesian readout and this sometimes results in an odd number of APVs on a particular detector. If the boards are located close enough along the beamline, it is sometimes possible to configure the master/slave setup across boards. If they are not, and a master is left without a slave, then one of the channels in the FEC reads out empty. This has to be accounted for within AMORE.

Doing so requires updates in both the decoding and encoding portions of AMORE. I initially experimented with skipping these FEC channels completely during the decoding process. Doing so required a complicated rewiring that was ultimately unnecessary. The easier way to handle this was to read in these channels normally during the decoding process and simply skip them during encoding. The encoding process happens within SRSEventBuilder, SRSAPVEvent and SRSDetectorPlaneEvent, and currently this feature is hard-coded to skip particular channels during analysis. This will be updated during the next round of updates and be set within the mapping config file.

The other subtlety with the encoding process is setting the mapping file correctly. If a FEC channel is skipped during readout, it should not be assigned within the config file. Just skip that channel and the analysis proceeds properly.

IX. Accommodating NAN Data

If a FEC channel is empty, it sends NAN data to AMORE. To account for this, we added a simple check within the modules: SRSRawPedestal, SRSPedestal and SRSDetectorPlaneEvent. The check is merely:

$if(data! = data)continue;$

The variable "data" is declared of type float, and a value is assigned to it before it reaches this step. This equality check works because a nan value cannot be referenced, and this is the only instance in $c++$ in which a variable does not equal itself.

X. Accommodating events with Multiple readouts for a particular APV

This problem was encountered at the beginning of the beam test, as we tried routing through an SRU storage device in parallel to the FECs. This caused a seg fault during the decoding process, since the implicit assumption is that there is only a single readout per APV per event. To fix this a check was written into SRSFECEventDecoder to count the number of readouts, and if it exceeds one it outputs an error in the terminal. It then continues analysis by skipping additional readout.

After we simplified the readout procedure along the beamline and removed the SRU, this problem disappeared. If it happens again, it should be dealt with immediately, since it significantly compromises the data.

XI. TTree Output

We have two forms of analysis-related output within AMORE: histograms and TTrees. Histograms are static ROOT objects that allow only minor changes (formatting, best fit lines, stats boxes, etc) after they are initially produced. TTrees, on the other hand, can be dynamically accessed. Even after they have been written to file, cuts can be easily made to a particular plot or the data within a particular TLeaf can be accessed and plotted against other TLeaves. Enabling TTree output allows us to quickly produce plots without reprocessing an entire data set.

TTree output was incorporated into AMORE and can be turned on or off through the shell script. In order to allow TTree data to be easily read back into AMORE for tracking analysis, we decided to directly write entire SRSCluster and SRSHit objects to file for each detector plane. The final result can be seen in Figure 1.

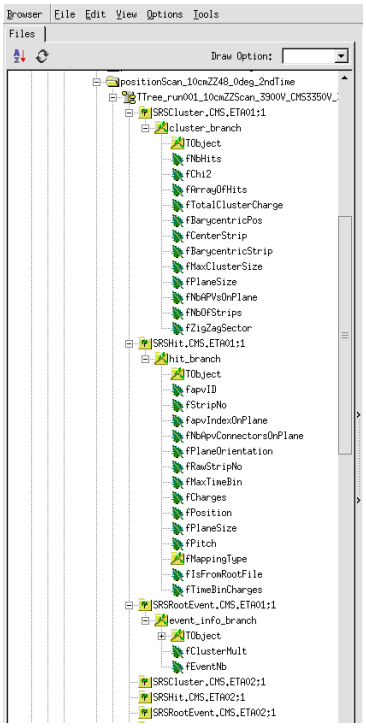

Figure 5: TTree readout structure for beam test data

These objects give us access to all post-processed data within AMORE. In other words, pedestal subtraction, common mode and zero suppression have already been performed and the data available here comes from the clusters created from the remaining data channels. An explanation of each of these variables is given below:

- Branch
  - SRSCluster: All clusters from every event are kept, so total entries = number of events * cluster multiplicity. Because of this, these plots are not simply copies of the histograms we manually produce. Those histograms contain cuts; these don't.
  - Leaves
    - fNbHits: Cluster size for all clusters.
    - $fChi^2$: This contains the χ^2/NDF measurement from the fit of each cluster's position and charge.
    - fArrayOfHits: This is not accessible in histogram form but it is of type "TObjArray *".. It's an array of the SRSHit * for each cluster. In other words, it contains the data from each strip in our cluster that ends up forming the SRSHit branch discussed below.
    - fTotalClusterCharge: This is the charge spectrum from ever strip in every cluster. Again, this is different from the normal histo output because it has no cuts.
    - fBarycentricPos: This is the barycentric position of each cluster – with no cuts applied.
    - fCenterStrip: This is the un-weighted middle strip of each cluster – no cuts.

    - fBarycentricStrip: This is the barycentric strip of each cluster – no cuts applied.
    - fMaxClusterSize: This is the max cluster size allowed in AMORE. Clusters larger than this are discarded. Right now it is hard-coded and set conservatively high at 30 strips.
    - fPlaneSize: plane size in mm.
    - fNbAPVsOnPlane: number of APVs in the plane.
    - fNbOfStrips: number of strips in the plane.
    - fZigZagSector: This is currently deprecated but it was created by Mike Staib to divide each zigzag sector into different parts and then map each part with either coarse or fine mapping.

- Branch
  - SRSHit: Information from each strip in each cluster (with no cuts for cluster multiplicity) is kept here.
  - Leaves
    - fapvID: This has the APV ID numbers for each APV in the plane. If the plane has multiple APVs, the number of entries for each is dependent on the number of hits that APV produced.
    - fStripNo: This is the same as the strip occupancy plot but it has no cuts for multi-cluster events.
    - fRawStripNo: Be careful with this plot. In some ways, its the analog of the fStripNo plot but it's at a lower level within Amore and probably not helpful for analysis purposes. This plot contains no global mapping, and when a plane only has one APV it will be identical to the fStripNo plot. However, when a plane has multiple APVs the information from each APV is overlayed onto one 128 bin plot. So if the beam profile for this run shows up between two APVs, this plot will have a U shape with the strips near 0 and 127 having the highest count.
    - fMaxTimeBin: Again, this plot is similar to the plot we output in manually produced histograms but it includes timebins from each strip in each event. So the number of entries = nEvents * (Average Cluster Multiplicity) * (Average Cluster Size).
    - fCharges: Be careful with this plot as well. It is not the charge distribution from the barycentric strip charge. It is a plot of the charge from every single strip of every single cluster from every single event.
    - fPosition: This plot should closely match the fStripNo plot. It is the positional analog of that plot.
    - fPlaneSize: Plane size in mm.
    - fPitch: Pitch in mm.
    - fIsFromRootFile: A boolean variable that tells us whether this data came from raw data input or root file input.
    - fTimeBinCharges: This is similar to the fCharges plot but it contains the charge from every single time bin of every single strip from every single cluster of every single event. This data comes after ped-subtraction, and common-mode and zero suppression.

- Branch

- SRSRootEvent: I created this class in order to store aggregate information about each event that would allow us to accurately reconstruct each event from just ROOT files (not raw data).
- Leaves
  * fClusterMult: Cluster multiplicity for each event.
  * fEventNb: This saves the event number for each event written to file. We do this to ensure no events were skipped.

XII. SRSProcessor

Previously, the process of forming clusters and accessing hit information was done in multiple sections of the code, whenever this data was needed. To cut down on computing time, I consolidated these two processes within a unique module called SRSProcessor. It is instantiated and called within SRSPublisher only once during each analysis run, and the cluster and hit lists it creates are available through an SRSProcessor object. This can be used in histogram production, root file output, track selection or any other analysis application.

XIII. Interpreting Raw Data

In order to diagnose higher level problems in monitoring and analysis, it may be necessary to sift through the unaltered raw data. As an example, during analysis this fall, we noticed that packets of data for a particular APV were occassionally empty inside the SRSAPVEvent module. This manifested itself through an empty container whose size (zero) was being used in division operations, resulting in nan values. The reason for this was apparently because time bin flags were not being flipped during the decoding process in SRSAPVEvent, resulting in the ADC data not being assigned to the data structure. In order to find the root cause, we output the raw, hexadecimal data within SRSFECEventDecoder and compared the good data with the bad. What we found was that the time bin headers – which usually fall under 1300 ADC (set within the mapping config file) were missing. In their place were absurdly high ADC values that were not low enough to signal a new time bin within AMORE. This is a similar problem that sometimes shows up in the raw data plots on the MTS after an HDMI cable has been bumped. It is usually indicative of a bad connection to the APV. The raw data readout for this problem is shown in Figures 3 and 4.

Taking a step back, data is initially taken and stored by DATE in binary .raw files. This data is converted to ASCII format by the low level processes within AMORE that occur before our implementation of AMORE receives its input. Once this data is converted it is sent, one event at a time, to the MonitorEvent process within SRSPublisher. Here the hexadecimal raw data is initially read out through SRSFECEventDecoder and stored within SRSEventBuilder.

A single event is composed of the data from, at most, 16 FEC channels from every available FEC. Each channel corresponds to a particular APV; the data dump from this APV includes data from however many time bins are assigned per event (for the data set shown here there were 9 time bins). Typically, we read out 1,500 16 bit-ADC words per FEC channel per event (although this is adjustable). With 128 channels per APV and 9 time bins in this particular data set, that amounts to 1,152 ADC data words per readout. The remaining 348 words are divided between headers, footers and meaningless fluff.

An example of the beginning of the readout for a particular FEC is shown in Figure 2. At the beginning of each FEC readout, there is a 9 word event header. Currently, the only meaningful

data available in this header is the event number, although this could be changed in the future. The next block of data represents the beginning of a 1,500 word readout for a particular FEC channel. The first 32-bit word is the channel header, which includes the packet size. In order to read a typical ADC word, the 32-bit word is broken up into two 16-bit chunks. Then each of these chunks is broken up into 8-bit values, with the second value read first and the first value read second. For example, in Figure 2, the ADC word 0c2c comes from reading the second 8-bit chunk first and the first 8-bit chunk second.

The beginning of a time bin within a particular APV readout is marked by a series of six 32 bit words, of which at least three consecutive words must be below the APV header setting. After this header, the next 64 32-bit words contain the APV channel data for that particular time bin.

In the next update to AMORE, an option will be available within the shell script to enable raw data readout to a text file. This will provide a sanity check during analysis and allow easy debugging of low-level readout problems.

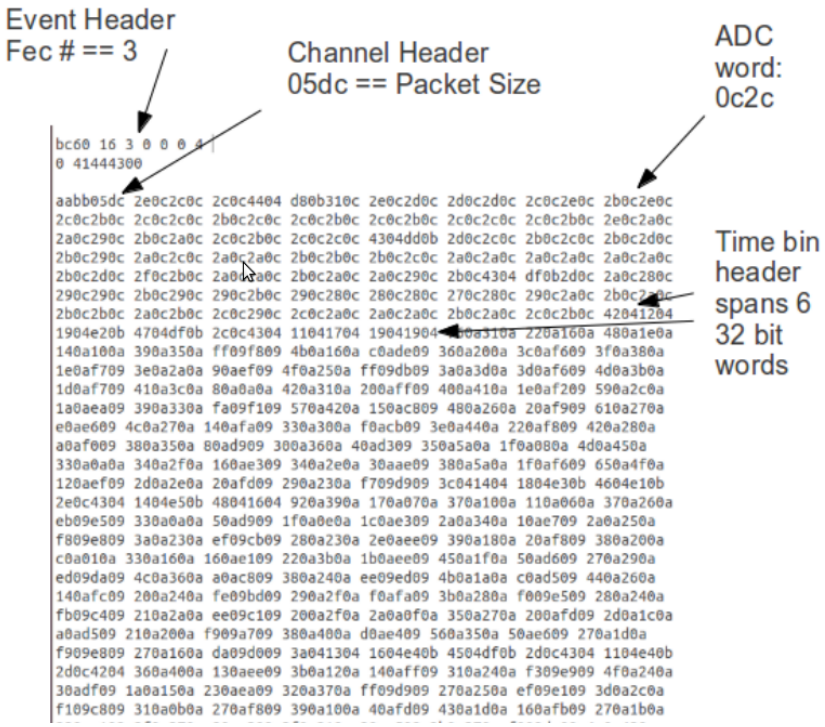

Figure 6: Interpreting raw data readout

```
4e2a apvKey 7 fecNo 4
aabb05dc
b0c0b0c b0c0c0c b0c0a0c b0c0a0c b0c0b0c c0c090c b0c0b0c b0c0b0c
90c0c0c a0c0b0c a0c0a0c b0c0a0c a0c0a0c c0c090c a0c0a0c c004950b
60c090c 80c080c 70c090c 90c090c 90c080c 80c080c 80c090c 90c090c
80c0a0c 80c080c 80c080c 80c0a0c 80c070c 80c080c 60c080c 60c060c
60cbf04 990b040c 60c060c 60c060c 50c070c 80c060c 70c080c 80c070c
50c070c 70c070c 60c080c 80c070c 70c070c 60c050c 50c060c 80c060c
80c060c 70c060c c0044704 4104940b bc044304 920bfe0b cc004 980bbd04
1b0a4f0a 7b0a470a 250a3b0a 4a0a490a 3b0a360a 5e0a2d0a 6a0a540a 3c0a300a
520a770a 960a590a 700a9f0a 920a730a 970a8d0a 9f0a4c0a 820a930a 830a5f0a
500a510a 600a400a 3f0a540a 4d0a280a 410a5f0a 580a270a 3d0a400a 4c0a330a
5c0a790a 920a5c0a 7d0a930a a40a630a 750aa90a 7d0a370a 670a9e0a 7a0a600a
4f0a550a 5f0a390a 330a580a 460a3e0a 450a540a 510a3b0a 420a610a 7d0a3b0a
570a980a 8a0a720a 790a780a b20a600a 860a950a a40a3e0a 6a0a780a 970a310a
630a7c0a 600a3e0a 440a4f0a 5b0a230a 3f0a4a0a 630a030a 3e0a6f0a 5c0a1c0a
590a900a 890a620a 730a880a 8b0a700a 640a860a 850a510a 6b0a990a 8a0a410a
aa044404 4104930b bc044504 930bfe0b be044604 930bbb04 260a700a 640a3d0a
5e0a610a 470a2c0a 7e0a570a 3e0a210a 970a5d0a 220a240a 4c0a6f0a 890a4e0a
4b0a840a 890a720a 790a840a 840a330a 680a910a 860a4f0a 6e0a6b0a 490a1e0a
690a6a0a 340a090a 720a5a0a 380a140a 780a3a0a 2a0a250a 490a710a 7b0a5b0a
500a850a 9f0a500a 5b0aa10a 730a390a 500a880a 800a520a 730a730a 510a110a
5d0a720a 320a250a 740a610a 3c0a170a 700a5e0a 5b0a190a 320a810a 790a690a
5d0a820a a80a630a 780a9a0a a20a470a 510a710a 780a3a0a 900a7a0a 500a280a
720a5f0a 380a0f0a 710a630a 4b0adc09 6a0a660a 3a0a070a 400a810a 840a580a
4f0a800a 8e0a730a 670a830a 750a430a 6e0a9e0a 890a240a a9044604 4004940b
bb044404 930bfe0b be044504 3f044204 230aae0a 960a8c0a 970aae0a 9d0a850a
```

Figure 7: Proper event readout, with time bin headers highlighted

```
4e2b apvKey 7 fecNo 4
aabb05dc
310c320c 320c310c 310c310c 300c310c 310c320c 320c320c 310c320c 2e0c300c
310c310c 320c2f0c 310c300c 310c310c 310c310c 2f0c300c 300c310c 300c2f0c
2f0c320c 300c300c 2f0c300c 300c2f0c 300c2f0c 2f0c2f0c 310c2f0c 300c2f0c
310c2e0c 2f0c300c 2e0c2f0c 2e0c2e0c 2e0c2e0c 2e0c2e0c 2e0c2f0c 2d0c2e0c
2f0c2f0c 2e0c2e0c 2f0c2f0c 2e0c300c 2f0c2d0c 2f0c2f0c 2e0c2f0c 2f0c2e0c
2d0c2f0c 2f0c2e0c 2f0c2f0c 300c2d0c 2f0c300c 2e0c2e0c 2f0c300c 2f0c2e0c
2f0c2e0c 2d0c2d0c 2d0c2f0c 2f0c2f0c 2f0c300c 2f0c2e0c 2f0c2d0c 2d0c2e0c
2d0c2c0c 2d0c2d0c 2d0c2c0c 2c0c2c0c 2c0c2e0c 2c0c2c0c 2e0c2e0c 2d0c2c0c
2e0c2f0c 2d0c2d0c 2d0c2f0c 2d0c2d0c 2e0c2e0c 2d0c2d0c 2d0c2f0c 2c0c2c0c
2e0c2e0c 2e0c2d0c 2e0c300c 2e0c2d0c 300c300c 2e0c2e0c 2f0c300c 2f0c2f0c
300c2f0c 2f0c2e0c 2e0c2f0c 2e0c2f0c 2d0c2f0c 2d0c2e0c 2f0c2e0c 2f0c2e0c
2f0c2f0c 300c2f0c 2e0c300c 2d0c2e0c 2e0c2e0c 2d0c2e0c 2e0c2d0c 2d0c2e0c
2d0c2e0c 2e0c2d0c 2e0c2e0c 2d0c2f0c 2e0c2e0c 2d0c2e0c 2f0c2e0c 2d0c2d0c
2e0c300c 2d0c2e0c 2f0c2f0c 2f0c2f0c 2f0c2f0c 300c2f0c 2f0c310c 2f0c300c
300c2e0c 2e0c2f0c 300c2f0c 2e0c2f0c 2f0c2f0c 2e0c2f0c 300c2e0c 2e0c2f0c
2e0c2e0c 2e0c300c 2e0c2f0c 2f0c2e0c 2f0c2f0c 2e0c2f0c 2e0c310c 300c2f0c
2f0c230c 230c1f0c 210c250c 250c1d0c 1d0c210c 220c200c 200c240c 240c1c0c
1d0c220c 220c1f0c 1f0c230c 230c1d0c 1d0c220c 220c1f0c 1e0c240c 240c1d0c
1c0c210c 200c1f0c 1f0c250c 240c1c0c 1c0c210c 210c1e0c 1e0c250c 240c1b0c
1c0c210c 220c1e0c 1e0c240c 230c1b0c 1d0c210c 220c1f0c 1f0c240c 260c1d0c
1b0c220c 230c200c 1f0c260c 250c1e0c 1d0c230c 220c1f0c 210c250c 260c1e0c
1c0c210c 210c200c 1f0c250c 250c1d0c 1c0c220c 210c1f0c 1e0c240c 240c1b0c
1a0c210c 210c1f0c 1f0c230c 250c1c0c 1c0c210c 210c200c 1f0c250c 250c1d0c
1c0c230c 200c1f0c 1c0c250c 220c1d0c 1c0c220c 210c200c 200c240c 240c1e0c
```

Figure 8: Bad event readout. The time bin headers are missing from this readout.

XIV. Accommodating Multiple Mapping Types within a Single APV

This will be updated soon.

XV. Alignment

This will be updated soon.

XVI. Tracking

This will be updated soonk.

XVII. Delta Y Analysis

In order to determine the average spatial resolution of our one dimensional zig-zag detectors, we created plots of the difference in y-hit position between adjacent detectors (ie. delta-y distribution). This could also be done on the 2 dimensional reference detectors by comparing the hit positions on a particular plane across detectors. We then fit a Gaussian curve to the data and extract the sigma value of the peak. In Figure 5, the sigma value is $\Delta y = 0.122$ strips and the strip pitch is approximately 2 mm. Since the two detectors are constructed identically, the resolution of the detectors can be found by taking $\frac{\Delta y}{\sqrt{2}} \approx 173 \mu m$.

These plots are available within SRSHistomanager.

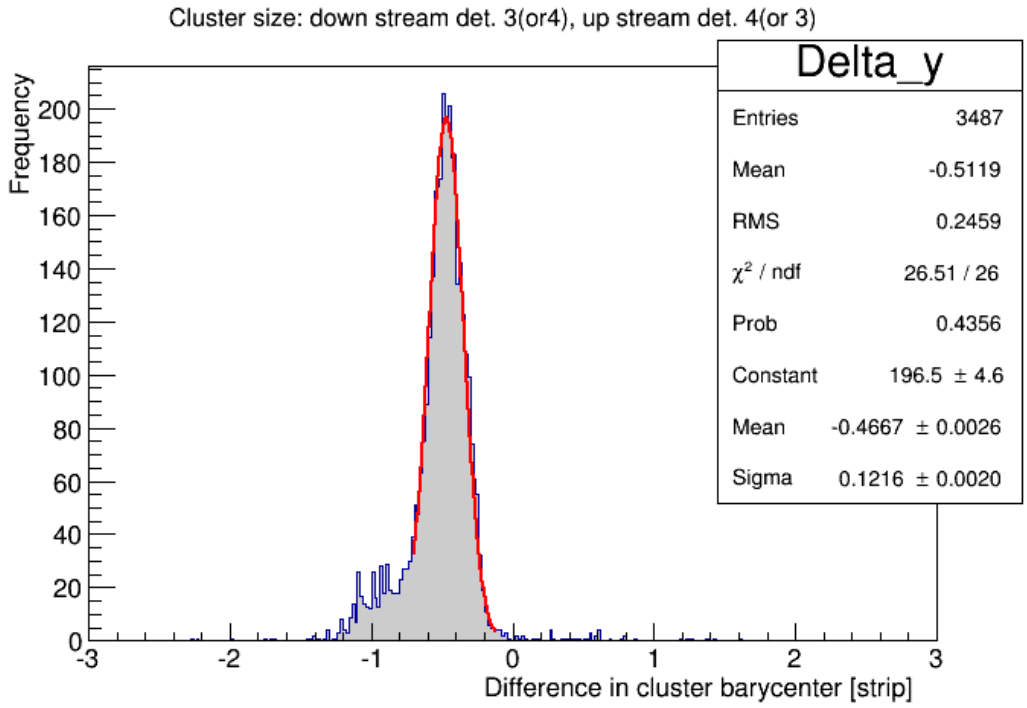

Figure 9: Initial Δ y distribution for a portion of the data for the 48 strip zig-zag detector.

XVIII. Data Backup and Code Repository

The newest version of AMORE is available on the cluster at */home/g4hep/AmoreVersions/amoreSRS/*. Important previous versions of the code are also available there, but the most recent version will always be titled amoreSRS. At this point, this code is also backed up on at least six other machines (SRSPC, AmorePC, and the personal computers for Aiwu, Vallary, Jessie and myself).

Data and results from the FNAL beam test are stored on */mnt/nas1/FNALBeamTest/*. This data is fully backed up on the SRSPC, as well as Vallary's and Aiwu's hard drives. We will also back the data up on the AmorePC.

XIX. Future Updates to AMORE

Some of these are mentioned in previous sections but included again for completeness.

- Raw data readout. This will be updated soon.
- Beam alignment. This will be updated soon.
- Beam tracking. This will be updated soon.
- Multiple mapping types within a single APV. This will be implemented over Christmas break.
- Zero suppression within FPGA, rather than AMORE. This project is slated to begin in the spring and will significantly reduce the processing time and the size of our raw data files. The aim will be to discard all channels not showing a signal above baseline while the data is still within the FEC, rather than writing out every single channel, only to throw away most of this data during analysis.
- Visual beam tracking event display for the monitoring package. A simple ROOT-based event display for the MTS already exists – although it is currently deprecated, since it was never integrated into the monitoring package. An updated version of this could be created for beam tests. The challenges would be setting the global positions for each detector and performing tracking analysis during monitoring. Currently, tracking modules are only used for offline analysis, in order to allow real-time monitoring. However, improvements could be made to AMORE to allow for online tracking during monitoring runs.

XX. Appendix A: Mapping Configuration File

```
1 ##################################################################################################################
2 #     DetName         PlaneName    DetNo    Plane   planeSize   nbConnectors  chNoMapping    LocalInvert, GlobalInvert, Pitch, NbOfStrips
3 ##################################################################################################################
4 #10x10: Cartesian readout
5 DET,   REF2,           REF2X,        0,       0,      104,          2,       CARTESIAN,          NO,           NO,     0.4,       256
6 DET,   REF2,           REF2Y,        0,       1,      104,          2,       CARTESIAN,          NO,           NO,     0.4,       256
7
8 #10x10: Cartesian readout
9 DET,   REF3,           REF3X,        1,       0,      104,          2,       CARTESIAN,          NO,          NO,     0.4,       256
10 DET,   REF3,           REF3Y,        1,       1,      104,          2,       CARTESIAN,          NO,           NO,     0.4,       256
11
12 #UVA 50cm GEM A
13 DET,   UVA1,           UVA1X,        2,       0,      450,          10,       CARTESIAN,       NO,          NO,     0.4,      1280
14 DET,   UVA1,           UVA1Y,        2,       1,      550,          12,       CARTESIAN,       NO,          NO,     0.4,      1536
15
16 #UVA EIC GEM
17 DET,   UVA2,           UVA2X,       3,       0,       270,          4,       EIC1,             NO,          NO,     0.4,       512
18 DET,   UVA2,           UVA2Y,       3,       1,       285,          8,       EIC1,             NO,          NO,     0.4,       1024
19
20 #CMS: Zigzag readout
21 DET,   CMSZZ,          ETA01,       4,       0,       255.936,          1,       CMSZZ,          NO,          NO,     2.015244,       128
22 DET,   CMSZZ,          ETA02,       4,       1,       274.0255,          1,       CMSZZ,          NO,          NO,    2.157681,       128
23 DET,   CMSZZ,          ETA03,       4,       2,       292.314,          1,       CMSZZ,          NO,          NO,     2.301685,       128
24 DET,   CMSZZ,          ETA04,       4,       3,       310.843,          1,       CMSZZ,          NO,          NO,     2.447583,       128
25 DET,   CMSZZ,          ETA05,       4,       4,       331.161,          1,       CMSZZ,          NO,          NO,     2.607567,       128
26 DET,   CMSZZ,          ETA06,       4,       5,       353.0325,          1,       CMSZZ,          NO,          NO,     2.779783,       128
27 DET,   CMSZZ,          ETA07,       4,       6,       377.608,          1,       CMSZZ,          NO,          NO,     2.973291,       128
28 DET,   CMSZZ,          ETA08,       4,       7,       404.427,          1,       CMSZZ,          NO,          NO,     3.184465,       128
29
30 #S4: ZigZag readout
31 DET,   S4ZZ,        HORIZ,        5,       0,      306.451,          1,       S4ZZ,             NO,          YES,     2.413,       128
32 DET,   S4ZZ,        VERTI,        5,       1,      100,              1,       S4ZZ,             NO,          YES,     2.413,       128
33
34 #10x10: Zigzag readout: 30 strips
35 DET,   ZZ30A,       ZZ01M,       6,       0,      104,              1,       ZZ30,                  NO,         YES,     3.5,     30
36
37 #10x10: Zigzag readout: 30 strips
38 DET,   ZZ30B,       ZZ02S,        7,       0,      104,              1,       ZZ30,                 NO,          YES,     3.5,       30
39
40 #UVA 50cm GEM B
41 DET,   UVA3,           UVA3X,        8,       0,      104,          2,       CARTESIAN,          NO,          NO,     0.4,      256
42 DET,   UVA3,           UVA3Y,        8,       1,      104,          2,       CARTESIAN,          NO,          NO,     0.4,      256
43
44 #10x10: Cartesian readout
45 DET,   REF1,           REF1X,        9,       0,      104,          2,       CARTESIAN,          NO,            NO,     0.4,       256
46 DET,   REF1,           REF1Y,        9,       1,      104,          2,       CARTESIAN,          NO,            NO,     0.4,       256
47
```

Figure 10: First part of the mapping configuration file. This section defines the detector planes of each detector.

```
48 ##################################################################################
49 #          APV Name        FEC Id   FEC chNo    APV Hdr  Active
50 ##################################################################################
51 #10x10: Cartesian readout
52 APV,       REF1XAPV1,         5,       8,       1500,     YES
53 APV,       REF1XAPV2,         5,       9,       1500,     YES
54 APV,       REF1YAPV1,         5,       10,       1500,    YES
55 APV,       REF1YAPV2,         5,       11,       1500,    YES
56
57 #10x10: Cartesian readout
58 APV,       REF2XAPV1,         5,       0,       1300,     YES
59 APV,       REF2XAPV2,         5,       1,       1300,     YES
60 APV,       REF2YAPV1,         5,       2,       1300,     YES
61 APV,       REF2YAPV2,         5,       3,       1300,     YES
62
63 #10x10: Cartesian readout
64 APV,       REF3XAPV1,         5,       6,       1300,     YES
65 APV,       REF3XAPV2,         5,        7,       1300,    YES
66 APV,       REF3YAPV1,         5,        4,       1300,    YES
67 APV,       REF3YAPV2,         5,        5,       1300,    YES
68
69 #UVA 50 cm UVA GEM 1
70 APV,       UVA1XAPV1,       2,        0,       1300,      YES
71 APV,       UVA1XAPV2,       2,        1,       1300,      YES
72 APV,       UVA1XAPV3,       2,        2,       1300,      YES
73 APV,       UVA1XAPV4,       2,        3,       1300,      YES
74 APV,       UVA1XAPV5,       2,        4,       1300,      YES
75 APV,       UVA1XAPV6,       2,        5,       1300,      YES
76 APV,       UVA1XAPV7,       2,        6,       1300,      YES
77 APV,       UVA1XAPV8,       2,        7,       1300,      YES
78 APV,       UVA1XAPV9,       2,        8,       1300,      YES
79 APV,       UVA1XAPV10,      2,        9,       1300,      YES
80 APV,       UVA1YAPV1,       2,        10,       1300,     YES
81 APV,       UVA1YAPV2,       2,        11,       1300,     YES
82 APV,       UVA1YAPV3,       2,        12,      1300,      YES
83 APV,       UVA1YAPV4,       2,        13,      1300,      YES
84 APV,       UVA1YAPV5,       2,        14,       1300,     YES
85 APV,       UVA1YAPV6,       2,        15,       1300,     YES
86 APV,       UVA1YAPV7,       3,        10,      1300,      YES
87 APV,       UVA1YAPV8,       3,        11,      1300,      YES
88 APV,       UVA1YAPV9,       3,        12,       1300,     YES
89 APV,       UVA1YAPV10,      3,        13,       1300,     YES
90 APV,       UVA1YAPV11,      3,        14,       1300,     YES
91 APV,       UVA1YAPV12,      3,        15,       1300,     YES
```

Figure 11: Second part of the mapping configuration file. This section assigns APVs to particular FEC channels.

```
#UVA EIC1M UVA GEM 2
APV,        UVA2XAPV1,       4,        4,       1300,      YES
APV,        UVA2XAPV2,       4,        5,       1300,      YES
APV,        UVA2XAPV3,       4,        6,       1300,      YES
APV,        UVA2XAPV4,       4,        7,       1300,      YES
APV,        UVA2YAPV1,       4,        8,       1300,      YES
APV,        UVA2YAPV2,       4,        9,       1300,      YES
APV,        UVA2YAPV3,       4,        10,      1300,      YES
APV,        UVA2YAPV4,       4,        11,      1300,      YES
APV,        UVA2YAPV5,       4,        12,      1300,      YES
APV,        UVA2YAPV6,       4,        13,      1300,      YES
APV,        UVA2YAPV7,       4,        14,      1300,      YES
APV,        UVA2YAPV8,       4,        15,      1300,      YES

#CMS: Zigzag readout
APV,        ETA01APV1,       3,        0,      1300,       YES
APV,        ETA02APV2,       3,        1,      1300,       YES
APV,        ETA03APV3,       3,        2,      1300,       YES
APV,        ETA04APV4,       3,        3,      1300,       YES
APV,        ETA05APV5,       3,        4,       1300,      YES
APV,        ETA06APV6,       3,        5,       1300,      YES
APV,        ETA07APV7,       3,        6,       1300,      YES
APV,        ETA08APV8,       3,        7,       1300,      YES

#S4: ZigZag readout
APV,        HORIZAPV1,       4,        0,       1300,      YES
APV,        VERTIAPV2,       4,        2,       1300,      YES

APV,        INACTAPV,        4,        1,       1300,       NO
APV,        INACTAPV,        4,        3,       1300,       NO

#10x10: Zigzag readout: 30 strip
APV,        ZZ01MAPV1,       3,        8,       1300,      YES

#10x10: Zigzag readout: 48 strip
APV,        ZZ02SAPV1,       3,        9,       1300,      YES

#UVA 50x50 UVA GEM 3
APV,        UVA3XAPV1,       5,        12,       1500,     YES
APV,        UVA3XAPV2,       5,        13,       1500,     YES
APV,        UVA3YAPV1,       5,        14,      1500,      YES
APV,        UVA3YAPV2,       5,        15,      1500,      YES
```

Figure 12: Third part of the mapping configuration file. This section assigns APVs to particular FEC channels.

```
136 ##################################################################
137 # ChannelNumber refers to the channel within the APV, while StripNumber refers to the physical strip location
138 # ONLY USED FOR THE S4 ZIGZAG
139 # VERTI 0-40; LRADI 41-81; SRADI 82-122; GROUND 123-127
140 #      ChannelNumber, StripNumber      PlaneName,       APV
141 ##################################################################
142 STRIP,     0,          85,             VERTI,       2
143 STRIP,     1,          86,             VERTI,       2
144 STRIP,     2,          84,             VERTI,       2
145 STRIP,     3,          87,             VERTI,       2
146 STRIP,     4,          83,             VERTI,       2
147 STRIP,     5,          88,             VERTI,       2
148 STRIP,     6,          82,             VERTI,       2
149 STRIP,     7,          89,             VERTI,       2
150 STRIP,     8,          81,             VERTI,       2
151 STRIP,     9,          90,             VERTI,       2
152 STRIP,     10,         80,             VERTI,       2
153 STRIP,     11,         91,             VERTI,       2
154 STRIP,     12,         79,            VERTI,       2
155 STRIP,     13,         92,            VERTI,       2
156 STRIP,     14,         78,            VERTI,       2
157 STRIP,     15,         93,            VERTI,       2
158 STRIP,     16,         77,            VERTI,       2
159 STRIP,     17,         94,            VERTI,       2
160 STRIP,     18,         76,            VERTI,       2
161 STRIP,     19,         95,            VERTI,       2
162 STRIP,     20,         75,            VERTI,       2
163 STRIP,     21,         96,            VERTI,       2
164 STRIP,     22,          74,            VERTI,       2
165 STRIP,     23,         97,            VERTI,       2
166 STRIP,     24,         73,            VERTI,       2
167 STRIP,     25,         98,            VERTI,       2
168 STRIP,     26,         72,            VERTI,       2
169 STRIP,     27,         99,            VERTI,       2
170 STRIP,     28,         71,            VERTI,       2
171 STRIP,     29,         100,            VERTI,       2
172 STRIP,     30,         70,            VERTI,       2
173 STRIP,     31,         101,            VERTI,       2
174 STRIP,     32,         69,            VERTI,       2
175 STRIP,     33,         102,            VERTI,       2
176 STRIP,     34,         68,            VERTI,       2
177 STRIP,     35,         103,            VERTI,       2
178 STRIP,     36,         67,            VERTI,       2
179 STRIP,     37,         104,            VERTI,       2
180 STRIP,     38,         66,            VERTI,       2
181 STRIP,     39,         105,            VERTI,       2
182 STRIP,     40,         65,            VERTI,       2
183 STRIP,     41,         106,            VERTI,       2
184 STRIP,     42,         64,            VERTI,       2
185 STRIP,     43,         107,            VERTI,       2
```

Figure 13: Fourth part of the mapping configuration file. This part is the hand mapped channel numbers for the 30x30. This was done because there were four different readout types on the same board. This is a hardcoded version of the automated work AMORE does all other readout boards.

```
186 STRIP,     44,         63,             VERTI,       2
187 STRIP,     45,         108,            VERTI,       2
188 STRIP,     46,         62,             VERTI,       2
189 STRIP,     47,         109,            VERTI,       2
190 STRIP,     48,         61,             VERTI,       2
191 STRIP,     49,         110,            VERTI,       2
192 STRIP,     50,         60,             VERTI,       2
193 STRIP,     51,         111,            VERTI,       2
194 STRIP,     52,         59,             VERTI,       2
195 STRIP,     53,         112,            VERTI,       2
196 STRIP,     54,         58,             VERTI,       2
197 STRIP,     55,         113,            VERTI,       2
198 STRIP,     56,         57,             VERTI,       2
199 STRIP,     57,         114,            VERTI,       2
200 STRIP,     58,         56,             VERTI,       2
201 STRIP,     59,         115,            VERTI,       2
202 STRIP,     60,         55,             VERTI,       2
203 STRIP,     61,         116,            VERTI,       2
204 STRIP,     62,          54,            VERTI,       2
205 STRIP,     63,         117,            VERTI,       2
206 STRIP,     64,         53,             VERTI,       2
207 STRIP,     65,         118,            VERTI,       2
208 STRIP,     66,         52,             VERTI,       2
209 STRIP,     67,         119,            VERTI,       2
210 STRIP,     68,         51,             VERTI,       2
211 STRIP,     69,         120,            VERTI,       2
212 STRIP,     70,         50,             VERTI,       2
213 STRIP,     71,         121,            VERTI,       2
214 STRIP,     72,         49,             VERTI,       2
215 STRIP,     73,         122,            VERTI,       2
216 STRIP,     74,         48,             VERTI,       2
217 STRIP,     75,         123,            VERTI,       2
218 STRIP,     76,         47,             VERTI,       2
219 STRIP,     77,         124,            VERTI,       2
220 STRIP,     78,         46,             VERTI,       2
221 STRIP,     79,         125,            VERTI,       2
222 STRIP,     80,         45,             VERTI,       2
223 STRIP,     81,         126,            VERTI,       2
224 STRIP,     82,          44,            VERTI,       2
225 STRIP,     83,         0,              VERTI,       2
226 STRIP,     84,         43,             VERTI,       2
227 STRIP,     85,         1,              VERTI,       2
228 STRIP,     86,         42,             VERTI,       2
229 STRIP,     87,         2,              VERTI,       2
230 STRIP,     88,         41,             VERTI,       2
231 STRIP,     89,         3,              VERTI,       2
232 STRIP,     90,         127,            VERTI,       2
233 STRIP,     91,         4,              VERTI,       2
234 STRIP,     92,         40,             VERTI,       2
```

Figure 14: Fifth part of the mapping configuration file. This part is the hand mapped channel numbers for the 30x30. This was done because there were four different readout types on the same board. This is a hardcoded version of the automated work AMORE does all other readout boards.

```
223 STRIP,     81,         126,            VERTI,      2
224 STRIP,     82,          44,            VERTI,      2
225 STRIP,     83,         0,             VERTI,      2
226 STRIP,     84,         43,             VERTI,      2
227 STRIP,     85,         1,             VERTI,      2
228 STRIP,     86,         42,             VERTI,      2
229 STRIP,     87,         2,             VERTI,      2
230 STRIP,     88,         41,             VERTI,      2
231 STRIP,     89,         3,             VERTI,      2
232 STRIP,     90,         127,              VERTI,        2
233 STRIP,     91,         4,             VERTI,        2
234 STRIP,     92,         40,            VERTI,        2
235 STRIP,     93,         5,             VERTI,        2
236 STRIP,     94,         39,            VERTI,        2
237 STRIP,     95,         6,             VERTI,        2
238 STRIP,     96,         38,            VERTI,        2
239 STRIP,     97,         7,             VERTI,        2
240 STRIP,     98,         37,            VERTI,        2
241 STRIP,     99,         8,             VERTI,        2
242 STRIP,     100,        36,            VERTI,        2
243 STRIP,     101,        9,             VERTI,        2
244 STRIP,     102,        35,            VERTI,        2
245 STRIP,     103,        10,            VERTI,        2
246 STRIP,     104,        34,            VERTI,        2
247 STRIP,     105,        11,            VERTI,        2
248 STRIP,     106,        33,            VERTI,        2
249 STRIP,     107,        12,            VERTI,        2
250 STRIP,     108,        32,            VERTI,        2
251 STRIP,     109,        13,            VERTI,        2
252 STRIP,     110,        31,            VERTI,        2
253 STRIP,     111,        14,            VERTI,        2
254 STRIP,     112,        30,            VERTI,        2
255 STRIP,     113,        15,            VERTI,        2
256 STRIP,     114,        29,            VERTI,        2
257 STRIP,     115,        16,            VERTI,        2
258 STRIP,     116,        28,            VERTI,        2
259 STRIP,     117,        17,            VERTI,        2
260 STRIP,     118,        27,            VERTI,        2
261 STRIP,     119,        18,            VERTI,        2
262 STRIP,     120,        26,            VERTI,        2
263 STRIP,     121,        19,            VERTI,        2
264 STRIP,     122,        25,            VERTI,        2
265 STRIP,     123,        20,            VERTI,        2
266 STRIP,     124,        24,            VERTI,        2
267 STRIP,     125,        21,            VERTI,        2
268 STRIP,     126,        23,            VERTI,        2
269 STRIP,     127,        22,            VERTI,        2
```

Figure 15: Sixth part of the mapping configuration file. This part is the hand mapped channel numbers for the 30x30. This was done because there were four different readout types on the same board. This is a hardcoded version of the automated work AMORE does all other readout boards.

XXI. Appendix B: Local Mapping

```
57              inline void SetLocalMapping(map<Int_t,Int_t> localStripMap) {fLocalStripCorrectionMap = localStripMap;}
58
59              Int_t StripNoCorrection(Int_t chNo){
60                      Int_t stripNo;
61
62                      if      (fMappingType == "CARTESIAN") stripNo = CartesianStripNoCorrection(chNo) ;
63                      else if (fMappingType == "ZZ30" || fMappingType == "ZZ48")    stripNo = ZigZagStripNoCorrection(chNo);
64                      else if (fMappingType == "MMRD51" && fAPVIndexOnPlane == 0)    stripNo = MMStripCorrectionAPV1(chNo);
65                      else if (fMappingType == "MMRD51" && fAPVIndexOnPlane == 1)    stripNo = MMStripCorrectionAPV2(chNo);
66                      else if (fMappingType == "MMRD51" && fAPVIndexOnPlane == 2)    stripNo = MMStripCorrectionAPV3(chNo);
67                      else if (fMappingType == "CMS" ) stripNo = CmsStripNoCorrection(chNo);
68                      else if (fMappingType == "CMSZZ") stripNo = CmsZigzagStripNoCorrection(chNo);
69                      else if (fMappingType == "NS2_30X30") stripNo = NS2StripNoCorrection(chNo);
70                      else if (fMappingType == "S4") stripNo = S4StripNoCorrection(chNo);
71                      else if (fMappingType == "S4ZZ") stripNo = S4ZigzagStripNoCorrection(chNo);
72                      else if (fMappingType == "EIC1") stripNo = EIC1StripNoCorrection(chNo);
73                      else    printf("SRSAPVEvent==Unknown mapping Type %s Index On Plane %i \n", fMappingType.Data(), fAPVIndexOnPlane);
74                      return stripNo;
75
76              }
```

Figure 16: This code can be found in SRSAPVEvent.h. Each of the mapping types is syphoned off to a separate function that handles the readout board specific mapping.

```
72 Int_t SRSAPVEvent::CartesianStripNoCorrection(Int_t chNo) {
73         //printf("Entering Cartesian Strip Correction\n");
74         Int_t stripCorrection = (32 * (chNo%4)) + (8 * (Int_t)(chNo/4)) - (31 * (Int_t)(chNo/16))  ;
75         if (fApvOrientation == 1) stripCorrection = 127 - stripCorrection ; // Remember: default is with local inversion (done because so that 0 is on
   left side of APV
76         return stripCorrection ;
77 }
```

Figure 17: This code can be found in SRSAPVEvent.cxx. This is the local mapping for the basic, straight strip readout boards.

XXII. Appendix C: Global Mapping

```
100  inline void SetGlobalInversion() { // this has to be done after pedestal subtraction (can't be done in SetStripNo as is); called now from
     SRSHit::GetPosition; SRSCluster::CalculatePosition;
101    if(fMappingType == "CARTESIAN" ||  fMappingType == "CMS" || fMappingType == "S4" || fMappingType == "CMSZZ" || fMappingType == "EIC1"){
102      if(fPlaneOrientation == -1) {fStripNo = (fNbApvConnectorsOnPlane * NCH) - 1 - fStripNo ; if (fStripNo < 0) cout << " strip No < 0 " << endl;}
103    }
104    else if (fMappingType == "ZZ30" && fStripNo < 30) {
105      if(fPlaneOrientation == -1) fStripNo = 29 - fStripNo ;
106    }
107    else if (fMappingType == "ZZ48" && fStripNo < 48) {
108      if(fPlaneOrientation == -1) fStripNo = 47 - fStripNo ;
109    }
110
111  }
112
113  void  SetStripNo(Int_t stripNo) {
114  //Describes how to map from the local strip number on the apv to the global strip number on the detector
115    if(fMappingType == "CARTESIAN" ||  fMappingType == "CMS" || fMappingType == "S4" || fMappingType == "CMSZZ" || fMappingType == "EIC1"){
116      fStripNo = stripNo + (NCH * (fapvIndexOnPlane % fNbApvConnectorsOnPlane)) ;
117      //if(fPlaneOrientation == -1) {fStripNo = (fNbApvConnectorsOnPlane * NCH) - 1 - fStripNo ; if (fStripNo < 0) cout << " strip No < 0 " << endl;}
118    }
119    else if (fMappingType == "ZZ30" || fMappingType == "ZZ48" || fMappingType == "S4ZZ") fStripNo = stripNo;
120    //else if (fMappingType == "S4") fStripNo = stripNo + (fapvIndexOnPlane * 128);
121  }
```

Figure 18: This code can be found in SRSAPVEvent.h. It shows the global strip assignment and the global inversion function. Note: at the time the strip assignment function is called, global inversion has not been done. This is done after common mode suppression to ensure the pedestal file aligns properly to the analysis data.

```
40  void ComputePosition() {
41   if (fMappingType == "ZZ48"){
42      //printf("computing pos ZZ\n");
43      if (fStripNo < 49) fPosition = -52 + (fStripNo * fPitch);
44    }
45    else if (fMappingType == "ZZ30") {
46        if (fStripNo < 31) fPosition = -52 + (fStripNo * fPitch);
47        else cout << " filling strips outside plane " << fStripNo << endl;
48    }
49    else if (fMappingType == "CARTESIAN" || fMappingType == "CMS" || fMappingType == "CMSZZ" || fMappingType =="NS2_30X30" || fMappingType == "MMRD51"
     || fMappingType == "S4" || fMappingType == "EIC1"){
50      fPosition = -0.5 * fPlaneSize + (fPitch * fStripNo);
51    else if (fMappingType == "S4ZZ") {
52    }
53    else printf("SRSHit-->Unknown Mapping type %s \n", fMappingType.Data());
54  }
```

Figure 19: This code can be found in SRSAPVEvent.h. It shows the positional calculation. This is done after global inversion and is called within the SRSCluster module.