I. ANALYSIS PROCEDURE

A. Preparing the Unit Distortion Tables

It was determined that the optimal way in which to build the unit distortion tables was with real data instead of Monte Carlo. An empty target Monte Carlo sample would have to be carefully tuned in order to have the same angular distributions as real data in order to accurately reflect the fact that tracks at different angles change in different ways under each shift/rotation. This requires the reconstruction of the alignment runs with each of the separate shifts and rotations. The following ccdb tables have been created and can be reused in future alignments:

```
1) nominal_shift_r1_localx_0p2cm
2) nominal_shift_r2_localx_0p2cm
3) nominal_shift_r3_localx_0p2cm
4) nominal_shift_r1_localy_0p2cm
5) nominal_shift_r2_localy_0p2cm
6) nominal_shift_r3_localy_0p2cm
7) nominal_shift_r1_z_0p2cm
8) nominal_shift_r2_z_0p2cm
9) nominal_shift_r3_z_0p2cm
10) nominal_shift_r1_y_0p2deg
11) nominal_shift_r2_y_0p2deg
12) nominal_shift_r3_y_0p2deg
```

The ccdb alignment table is read in the global coordinate system so the naming convention here is used to specify that shifts in x and y are in the local coordinate system (i.e. a local 0.200 cm shift in x in sector 2 corresponds to a 0.100 cm shift in x and 0.173 cm shift in y in the global system). A local shift in z is equivalent to a global shift in z and so the z shift tables are not referred to as "local". Finally, the rotations tables have been specified with the angle they are rotated around and an amount specified in degrees. In hindsight, it would have been preferable to name these, e.g. "cy".

The ccdb tables must be specified in the yaml file used for reconstruction in two separate places with the variable "dcGeometryVariaton",

```
DCHB:
      variation: rga_fall2018
      dcGeometryVariation: nominal_shift_r1_localx_0p2cm
      dcT2DFunc: "Polynomial"
      dcWireDistortion: "true"
    DCTB:
      variation: rga_fall2018
      dcGeometryVariation: nominal_shift_r1_localx_0p2cm
```

The "variation" variable will pull all other relevant constants.

B. Track Selection

We applied two separate cuts in order to ensure that the tracks being considered for the alignment procedure were coming from real electrons. First, the CLAS12 cherenkov detector was required to detect two or more photoelectrons. This cut was performed by selecting events in which the "nphe" variable in the "REC::Cherenkov" bank was greater than or equal to two. Second, a minimum calorimeter energy of 0.06 GeV (the CLAS12 calorimeter has a sampling fraction of 1/4) was required. These tracks were selected by requiring the "energy" variable in the "REC::Calorimeter" bank was greater than 0.06 GeV.

C. Script 1: Determing the size of residual shifts

Three separate scripts were used in the initial alignment procedures and are available in the github repository xxxxxxxx.

The first is *shift_tables.groovy*, a groovy (Java) script written to extract the size of the shifts on the mean of the residual distributions. The script accepts 4 input arguments and then a variable number of arguments depending on your choices. The first two inputs are directories of hipo files, generally intended to be one with the default geometry and one with a shifted geometry. The second two (3 and 4) arguments are the number of bins the polar angle, θ, and the local azimuthal angle, ϕ (ranging from 0 to 60 in each sector), the residuals should be divided into. Finally, the remaining arguments are the upper bounds of each bin. For example, an input argument of

```
$ run-groovy shift_tables.groovy /directory_A/ /directory_B/ 2 2
      10 20 30 60
```

would calculate the difference in residuals between hipo files in $directory_A$ and $directory_B$ with θ bins of [0,10] and [10,20] and ϕ bins of [0,30] and [30,60]. An optional final argument places a limit on the number of files to iterate over. If nothing is specified then the value is set to the number of files in the directories. The output would look something like:

```
{{-245.7, -116.5, -147.2, -96.2, -46.6, -72.0, -167.1, -155.5,
-182.2, -179.9, -248.0, -173.5, 168.5, 54.4, 55.9, -222.5, 76.4,
-116.8, 171.9, 57.1, 203.9, 197.4, 202.1, 120.6, -263.0, -530.8,
-349.7, -340.6, -45.6, -443.1, -171.3, -323.3, -109.4, -424.5,
-143.7, -602.0, },
{-37.0, -34.0, -53.1, -33.7, -24.6, -122.2, 150.9, -128.5, -148.0,
-181.6, -118.4, -62.0, -21.2, -100.5, -291.9, 62.2, -15.5, -59.3,
-147.9, 163.8, 15.5, 204.1, 63.9, -26.6, -197.0, -527.0, -100.6,
-480.2, -44.7, -415.7, -266.3, -442.1, -173.8, -329.1, -80.6,
-604.0, },
{24.1, -331.6, 100.4, -192.1, -133.4, -133.4, 29.4, -433.1, 129.8,
-33.3, -346.7, 114.4, -151.8, 276.8, 146.7, 140.9, 274.0, 22.6,
175.6, 306.7, 142.2, 88.4, 321.7, -1.3, -55.6, -172.3, 53.7,
-378.1, 12.6, -268.7, 11.0, -481.5, 107.8, -168.8, -233.9,
-193.0, },
{-392.4, -99.8, -265.2, -148.3, -429.8, 6.0, -107.1, -18.4, -30.8,
-123.9, -66.6, -192.8, 76.0, 206.9, 141.7, 92.4, 299.7, 71.5, 17.9,
36.3, 108.3, 2.7, 128.8, -216.8, -1.1, -89.6, 32.1, -160.8, -157.6,
-311.0, 125.2, -531.8, -35.7, -69.8, 157.6, 19.2, }};
```

where the first list corresponds to the mean difference in residuals of each of the 36 layers in the $\theta \in [0, 10]$ and $\phi \in [0, 30]$ bin, the second corresponds to $\theta \in [0, 10]$ and $\phi \in [30, 60]$, etc. Note that because the shifts and rotations are performed in the local sector coordinate system the changes are the same for all six sectors. There is an extra space at the end of each list that will correspond to the 37th element of alignment: the downstream vertex position. This method necessitates running over a directory with each separate shift or rotation separately; in future this could be optimized with a master code to produce all the tables simultaneously.

D. Script 2: Determing the size of vertex shifts

The second script is *vertex_studies.groovy*, a groovy (Java) script written to study the z position of the vertex of selected tracks. The "alignment runs" were empty target and so a plot of the vertex of tracks shows distributions peaked at the target walls. For the alignments discussed here the downstream wall was fit and aligned to survey data but this choice is somewhat arbitrary. *vertex_studies.groovy* operates in the same way as *shift_tables.groovy*. The script accepts 3 input arguments and then a variable number of arguments. The first is a directory of hipo files and the second and third are the number of bins to divide θ and ϕ into. The remaining arguments are again the upper bounds of each bin. For example, an input argument of

```
$ run-groovy vertex_studies.groovy /directory/ 2 2 10 20 30 60
```

would plot the vertex distributions separately for all tracks in the files in *directory* with θ bins of [0,10] and [10,20] and ϕ bins of [0,30] and [30,60]. An additional final argument is available again to limit the number of files to iterate over.

The downstream peak is then fit and added to the corresponding position in the arrays of the previous script. In this alignment procedure the peaks were fit using the interactive groot GUI fitting procedure but in future this process could be optimized.

E. Script 3: MINUIT Minimization

The final script is *alignment_table.cpp* a root (C++) script that takes the information obtained in the first two scripts and uses MINUIT to find the alignment parameters. A number of arrays are specified at the top of the script corresponding to the starting mean positions of of the residuals in each layer, divided up into successive θ and ϕ bins. After this, the difference a 0.2 cm shift or 0.2 degree rotation makes on these means is listed for each possible shift and rotation. The script will take the starting positions of each of the 36 layers and vertex and the corresponding amount each chamber shifts these values and attempt to minimize the mean layer position and the difference between vertex survey position and reconstructed value. The output of the minimization script will look something like:

```
#& region sector component dx dy dz dtheta_x dtheta_y dtheta_z
1 1 0 0.263 0 0.192 0 -0.072 0
1 2 0 0.274 0.475 -0.050 0 -0.072 0
1 3 0 -0.0361 0.062 -0.108 0 -0.082 0
1 4 0 0.099 -3.706e-10 -0.031 0 0.026 0
1 5 0 -0.035 -0.061 -0.031 0 0.0265 0
1 6 0 0.117 -0.193 -0.0113 0 -0.064 0
2 1 0 0.166 0 -0.298 0 -0.0476 0
2 2 0 0.156 0.270 -0.356 0 -0.177 0
2 3 0 -0.049 0.085 -0.159 0 -0.047 0
2 4 0 -0.150 5.619e-10 -0.379 0 0.041 0
2 5 0 -0.005 -0.009 -0.235 0 0.0421 0
2 6 0 0.0212 -0.0368 -0.176 0 -0.089 0
3 1 0 -0.024 0 -0.237 0 -0.084 0
3 2 0 -0.161 -0.280 -0.178 0 -0.191 0
3 3 0 -0.0392 0.068 -0.033 0 -0.024 0
3 4 0 -0.319 1.173e-09 -0.200 0 0.0279 0
3 5 0 -0.141 -0.244 -0.204 0 0.0089 0
3 6 0 -0.022 0.0388 -0.167 0 -0.103 0
```

This table can be copied directly into a text file and uploaded to the ccdb in order to use in alignment of data.