Helpful links

[TileCisCalibration Twiki](https://twiki.cern.ch/twiki/bin/view/Atlas/TileCisCalibration): Overview of CIS calibration

[TileCisCalibrationProcedure Twiki](https://twiki.cern.ch/twiki/bin/viewauth/Atlas/TileCisCalibrationProcedure): Information about CIS updates and database tags

[PrimaryCisTools Twiki](https://twiki.cern.ch/twiki/bin/view/Atlas/PrimaryCisTools): Detailed description of useful macros and plotting tools.

[TUCS CIS plots guide](https://docs.google.com/document/d/14RxNsx61A0L8cnDrbXUuIAbqMpTlqutxSXcQyy_V3zc/): Working execution examples for plotting procedures, more narrow but inspired by PrimaryCisTools.

[Run List (Tile In One Plugin)](https://tio.cern.ch/run-list/): find recent CIS runs

Katie’s CIS helpers: <https://github.com/katie-hughes/CIS_helpers>

They are also located in ~khughes/public/CIS\_helpers within lxplus, but you should clone the repository somewhere in your lxplus account so you can edit things!

Readme.md contains lots of information about each script and its optional arguments.

Run: git clone <https://github.com/katie-hughes/CIS_helpers.git>

(SHould also do git pull to add newest updates)

Sasha (Alexander) (Sanya) Solodkov (Sanya.Solodkov@cern.ch) and Henric Wilkens (Henric.Wilkens@cern.ch) will be your primary contacts for CIS updates.

1. Log into your lxplus account
   1. ssh -XY USERNAME[@lxplus.cern.ch](mailto:username@lxplus.cern.ch)
2. Setup Athena environment
   1. asetup
      1. Version that is automatically loaded for me is 22.0.46
      2. To load environment for a specific version, run asetup Athena [VERSION]
   2. ALTERNATE for newer version of Tucs: source setup.sh (will automatically load the most recent version)
3. Ensure the ~/Tucs/results folder is clean – specifically that there is no old tileSqlite.db file already there (it is fine if they are renamed or in different sub-directories). If not, things can get overwritten in weird ways.
4. Run CIS update macro from Tucs directory over the last month
   1. macros/cis/CIS\_DB\_Update.py --date 'start' 'end' |&tee MonthCIS.txt
   2. Example: macros/cis/CIS\_DB\_Update.py --date 'March 1, 2022' 'April 1, 2022' |&tee results/MarCIS.txt
   3. I like to use |&tee, it keeps everything printed on the terminal so you can see the progress but also writes to a file, overwriting the contents each time (<https://askubuntu.com/questions/420981/how-do-i-save-terminal-output-to-a-file>)
5. You now have some files created:
   1. tileSqlite.db : this file contains each channel and its newly calculated CIS constant. Will be in ~/Tucs/results
   2. CIS\_DB\_Update.txt : lists the channels in the update, as well as some statistics from the update. Also in ~/Tucs/results
   3. NoCIS.txt (sometimes): list of channels with no CIS response. Usually never needed. In ~/Tucs/results
   4. Plots: located in ~/Tucs/plots/latest/cis/CIS\_Update
6. Verify that amplitude/charge ratios and timing plots for the runs used are good
   1. I have a script to automate this process, cis\_charge.C
      1. root
      2. .L ~.../CIS\_helpers/cis\_charge.C
      3. check\_all\_runs()
         1. You currently have to update the list of runs directly in the function check\_all\_runs, I don’t have an automated way of doing this.
         2. An easy way to extract the list of runs is running my script ReturnRuns.py which takes in the MonthCIS.txt file you get from running the update macro
            1. python3 ~/…/CIS\_helpers ReturnRuns.py -f MonthCIS.txt -c
   2. For each run, the script will produce 4 different plots:
      1. Timing (LG and HG): should be in the range of (-15, 10) ns. Sometimes this can be shifted by multiples of +/- 25 ns. If it is not in this range, the run shouldn’t be used. (I don't entirely understand why, but I think it has to do with how the points are sampled from the pulse shape.)
      2. Amplitude/Charge ratio (HG and LG): To be “good”, there shouldn’t be any extreme outliers from the trend. My understanding is that the CIS constant is calculated as the slope of ADC Amp vs injected charge. This should be a straight line, so amplitude/charge not being relatively constant means there is some outlier that is messing up the calculation of the CIS constant.
         1. IF Amplitude is exactly 0, it is not a problem as the cis update script will automatically filter it out. LBA has a recurring problem with this but it does not affect CIS constant calculations. Currently in cis\_charge.C, the cutoff is for amplitude of 6 (LG) or amplitude of 40 (HG) as recommended by Sasha.
   3. When timing or amp/Q is bad, you can often see this in the CIS plots as a consistent outlier point an entire partition/gain combo.
   4. Technically, bad timing or amp/q only invalidates a run for a particular partition and gain combination. For example, if timing is bad for LBA\_lowgain, it could be used in the calculation for LBA\_highgain, EBA, and EBC. I have found that it just simplifies the analysis to remove it for ALL partitions/gains if it is only bad for one, so long as there are still enough other runs that are good. (otherwise you have to do the update up to 8 different times and mesh together all these different sqlites, it becomes very messy for a near negligible difference).
7. IF runs need to be removed because of bad timing or amp/q, rerun the CIS update using the --ldate parameter to list the runs to be used.
   1. Delete/move/rename previous plots and sqlite so they do not get overwritten.
   2. macros/cis/CIS\_DB\_Update.py --date '-31 days' --ldate [runs separated by spaces] |&tee MonthCIS.txt
   3. The --date parameter no longer matters if you use --ldate but you still have to include it otherwise the macro doesn’t run (this should be fixed at some point)
   4. Example: macros/cis/CIS\_DB\_Update.py --date '-31 days' --ldate 413579 413594 414257 414464 414513 414763 415169 415434 415531 415564 415914 416099 |&tee MarCIS.txt
8. Read Sqlite into txt file
   1. ReadCalibFromCool.py --schema="sqlite://;schema=tileSqlite.db;dbname=CONDBR2" --folder=/TILE/OFL02/CALIB/CIS/LIN --tag=UPD1 | grep -v miss > cis.txt
   2. Format of cis.txt: PartitionModule Channel Gain Value
9. Examine plots by hand
   1. I find it laggy to view over ssh, so I copy them to my own computer
      1. scp -r USERNAME[@lxplus.cern.ch](mailto:khughes@lxplus.cern.ch):~/Tucs/plots/latest/cis/CIS\_Update ~/Desktop
   2. It is possible to view them over ssh if you have a linux machine by using eog:
      1. Example: eog ~/Tucs/plots/latest/cis/CIS\_Update hist.png
10. Record any problematic channels or changes that will need to be made manually. You can list these in the report. It is most important to report if the channels are not already flagged as masked
    1. Channels to Recalibrate
       1. Sometimes there is one or two outlier points, or a constant shift in the CIS constant
       2. Solution: recalibrate after problematic location (see step 12)
       3. Keep track of which channels need to be recalibrated, and at which date
    2. Incorrect BadCIS flags
       1. Sometimes TUCS incorrectly assigns/doesn’t assign the BadCIS flag
       2. Keep track of which channels need to have this flag added/removed.
       3. BadCIS channels are either far from the detector average (~ >5%, sort of subjective) or unstable, jump around a lot,etc.
    3. Half gain channels
       1. An electronics issue
       2. Half gain channels should NOT be jumping between half gain and normal levels. This is a sign that the CIS constant fit is bad.
       3. LBA14 (“the demonstrator”) will always be at half gain because it uses new electronics. Do not mark it as bad CIS because of this!
    4. Channels unusually far from detector average
    5. High scatter channels
    6. Channels that are drifting or have some unexpected change in behavior
11. Recalibrate Channels
    1. I created a python script to assist with recalibrating macros, StudyFlagCommand.py (in CIS\_helpers). To use:
       1. Fill a txt file toRecalibrate.txt with the channels to be recalibrated
       2. Make sure sqlite.db and CIS\_DB\_update.txt are in a different folder
       3. Run: python3 ~/…/CIS\_helpers StudyFlagCommand.py --cistxt MonthCIS.txt --cisupdate
          1. --cisupdate flag is optional but strongly recommended as constants will be more accurate, it will give you the exact numbers instead of you having to read them off the plot. (Recently made so it might have bugs? lol)
       4. It can also break it down further if you have different runs for different partitions but you probably won’t have to do that
       5. From Tucs directory run ./results/Month/RecalCommands.sh
          1. This will create a file toRecalibrate.txt located in ~/results.
       6. Then from ~/Tucs/results do mv toRecalibrate.txt MonthCIS
          1. This will overwrite previous toRecalibrate that only contained the dates and not the values, which is what we want!!!
    2. New plots will be created in Tucs/plots/latest/cis/StudyFlag/TimeStability/all (If you use Study Flag Macro) or just in regular CIS\_update place (if you use regular CIS update macro)
       1. If you use --cisupdate flag, you should double check that the values on the plots match what is in the text file.
       2. If you do not use --cisupdate flag you will have to manually read values off the plots and add them to a text file.
    3. Next you want to update the Sqlite file to include these new values. You have to do this in two steps – one of channels in the update, and one of channels not. Running my script corr.py will produce the two txt files you need, it uses the fact that the channels in the update are all listed in CIS\_DB\_Update.txt. corr1.txt is for channels in the update and corr2.txt is for ones not.
       1. python3 ~/…/CIS\_helpers corr.py (run in same directory as CIS\_DB\_Update.txt)
          1. Will automatically assume filename is toRecalibrate.txt. Ensure that the one being used is the one that actually contains the constants.
    4. Next run these two commands to update your local sqlite file. Get the run number XXXXXX from CIS\_DB\_Update.txt
       1. WriteCalibToCool.py --schema="sqlite://;schema=tileSqlite.db;dbname=CONDBR2" --txtfile=corr1.txt --folder=/TILE/OFL02/CALIB/CIS/LIN --tag=UPD1 --run=XXXXXX
       2. WriteCalibToCool.py --inschema=COOLOFL\_TILE/CONDBR2 --outschema="sqlite://;schema=tileSqlite.db;dbname=CONDBR2" --update --txtfile=corr2.txt --folder=/TILE/OFL02/CALIB/CIS/LIN --tag=UPD1 --run=XXXXXX
    5. Read updated sqlite file into another txt file and verify that the channels have indeed been updated / it is different from the first cis.txt file.
       1. ReadCalibFromCool.py --schema="sqlite://;schema=tileSqlite.db;dbname=CONDBR2" --folder=/TILE/OFL02/CALIB/CIS/LIN --tag=UPD1 | grep -v miss > cisRecal.txt
12. Push constants!
    1. The online database requires files with certain tags. To generate these use a script from Sasha.
       1. ~solodkov/scripts/calib\_to\_oracle CALIB/CIS/LIN tileSqlite.db
    2. scp tileSqlite\_upd4\_CIS\_LIN.db file to your home computer
    3. Log into the robot: [https://atlas-tile-calib.cern.ch](https://atlas-tile-calib.cern.ch/v1/) (only one person can log into each version of the robot at a time, so sometimes certain pages are occupied and you have to use another listed below)
       1. [https://atlas-tile-calib.cern.ch/v1/](https://atlas-tile-calib-dev2.cern.ch/v1/)
       2. …
       3. [https://atlas-tile-calib.cern.ch/v9](https://atlas-tile-calib-dev2.cern.ch/v9)
    4. Upload the upd4 tagged db to CISLIN\_ALL
    5. That’s it!!! Post to elog and exit :)
13. Update Flags
    1. The cis update db will automatically push flag updates but sometimes you need to do some by hand.
    2. I find it easiest to make a new directory to deal with the flags so you don’t get the sqlites mixed up
       1. mkdir flags
    3. I have some scripts in CIS\_helpers to read in the flag values.
       1. Update\_flags.py will be used to actually read the values in as an execfile
       2. Test\_update\_flags.py is (surprise) a test so you can print the values out and make sure there are no typos, etc. for when you run the actual thing.
       3. Copy both of these to the flags directory you just made!
    4. Make two files, RemoveBadCIS.txt and AddBadCIS.txt and fill accordingly
    5. Run test\_update\_flags.py and ensure that comments all seem right, no typos, etc
    6. Generate your sqlites from these files:
       1. WriteBchToCool.py --execfile=update\_flags.py --online --upd4
    7. Copy the tileSqlite\_upd4.db to your home computer and upload to to ADC\_UPD4\_UPD1 on the robot
    8. All done! Post to elog and exit
14. Make summary plots for presentation
    1. macros/cis/Public\_Super\_Macro.py --gcals --date '05/01/22' '06/01/22' --datelabel 'May 1 - June 1, 2022' --mean --lowmem --rmsplots --flagplots
       1. Make sure to update to python3 syntax (see getting started with Tucs document)
    2. macros/cis/Public\_Super\_Macro.py --history -0.5 0.5 --date 'April 1, 2022' 'May 1, 2022' --ndate 'April 1, 2021' 'May 1, 2021' --datelabel 'Apr. 2021 vs 2022'
    3. Copy (example): scp -r khughes@lxplus.cern.ch:~/Tucs/plots/latest/cis/Public\_Plots /home/katie/CIS/May2022/

Summary of commands to run:

* ssh -XY USERNAME@lxplus.cern.ch
* asetup
* cd Tucs
* [clean Tucs/results]
* macros/cis/CIS\_DB\_Update.py --date 'start' 'end' |&tee results/MonthCIS.txt
* cd results
* mkdir MonthCISUpdate
* mv tileSqlite.db CIS\_DB\_Update.txt MonthCIS.txt MonthCISUpdate
* cd MonthCISUpdate
* python3 ~/.../CIS\_helpers/ReturnRuns.py -f MonthCIS.txt -c
* [copy the list of runs from runList.txt into ~/.../CIS\_helpers/cis\_charge.C function check\_all\_runs()]
* root
* .L ~/.../CIS\_helpers/cis\_charge.C
* check\_all\_runs()
* .exit()
* [examine plots and record which runs are bad]

IF there are bad runs:

* + cd ~/Tucs/plots/latest/cis/
  + rm -r CIS\_Update OR mv CIS\_Update OLD\_CIS\_Update [ensure you don’t get old plots mixed in with new ones, you can also rename it if you want to save for some reason]
  + cd ~/Tucs
  + macros/cis/CIS\_DB\_Update.py --date 'start' 'end' --ldate [new list of runs] |&tee results/MonthCIS.txt
  + cd results/MonthCISUpdate
  + rm \* [delete old files]
  + cd ..
  + mv tileSqlite.db CIS\_DB\_Update.txt MonthCIS.txt MonthCISUpdate
* cd ~/Tucs/results/MonthCISUpdate
* ReadCalibFromCool.py --schema="sqlite://;schema=tileSqlite.db;dbname=CONDBR2" --folder=/TILE/OFL02/CALIB/CIS/LIN --tag=UPD1 | grep -v miss > cis.txt

IF you want to view the plots on your own computer:

* + open new terminal tab
  + scp -r USERNAME@lxplus.cern.ch:~/Tucs/plots/latest/cis/CIS\_Update ~/Desktop [or whatever location u want]

ELSE:

* + eog ~/Tucs/plots/latest/cis/CIS\_Update/hist.png
* [track problematic channels, channels to be recalibrated, and channels with flag updates]
* touch toRecalibrate.txt [and fill with channels to recalibrate]
* python3 ~/.../CIS\_helpers/StudyFlagCommand.py --recal\_file toRecalibrate.txt --cistxt MonthCIS.txt --cisupdate
* cd ~/Tucs
* ./results/MonthCISUpdate/RecalCommands.sh
* cd results
* mv toRecalibrate.txt MonthCISUpdate
* cd MonthCISUpdate
* python3 ~/.../CIS\_helpers/corr.py
* [get latest run number from CIS\_DB\_Update.txt for next commands]
* WriteCalibToCool.py --schema="sqlite://;schema=tileSqlite.db;dbname=CONDBR2" --txtfile=corr1.txt --folder=/TILE/OFL02/CALIB/CIS/LIN --tag=UPD1 --run=XXXXXX
* WriteCalibToCool.py --inschema=COOLOFL\_TILE/CONDBR2 --outschema="sqlite://;schema=tileSqlite.db;dbname=CONDBR2" --update --txtfile=corr2.txt --folder=/TILE/OFL02/CALIB/CIS/LIN --tag=UPD1 --run=XXXXXX
* ReadCalibFromCool.py --schema="sqlite://;schema=tileSqlite.db;dbname=CONDBR2" --folder=/TILE/OFL02/CALIB/CIS/LIN --tag=UPD1 | grep -v miss > cisRecal.txt
* diff cis.txt cisRecal.txt [verify they are different, and that recal has the correct new values]
* ~solodkov/scripts/calib\_to\_oracle CALIB/CIS/LIN tileSqlite.db
* scp tileSqlite\_upd4\_CIS\_LIN.db file to your home computer and upload to to CISLIN\_ALL
* mkdir flags
* cp ~/.../CIS\_helpers/\*update\_flags\* flags/
* touch RemoveBadCIS.txt AddBadCIS.txt [and fill accordingly]
* python3 test\_update\_flags.py [ensure that comments all seem right, no typos, etc]
* WriteBchToCool.py --execfile=update\_flags.py --online --upd4
* scp tileSqlite\_upd4.db to your home computer and upload to to ADC\_UPD4\_UPD1

# 

# 

# C++ Files to be run via Root

my\_plot\_cis.C

Lets you look at individual cis constant fits of adc count vs injected charge

plot(run, partition, module, channel, gain)

plot\_all\_runs(partition, module, gain)

Partition:

1=LBA

2=LBC

3=EBA

4=EBC

cis\_charge.C

Plots timing and amplitude/charge.

Most relevant command: check\_all\_runs()

Would be nice if I can put the list/vector directly as a function argument but idk how to do that. Instead I just edit the line in plot\_all\_runs.

Other relevant commands:

cis\_ampq(run, module=2, chan=2, xval=0)

cis\_timing(run, module=2, chan=2, xval=0)

Currently check\_all\_runs() defaults to checking just module 2, channel 2 for all partitions. Usually this is representative (ie if there is a problem in the partition, it will be visible in all modules and channels). But sometimes just to be safe I will try another module if there seems to be a large problem just to make sure it is replicated there.

# 

# 

# Todo:

Fix corr.py so it doesn’t care if the dates are there (not really necessary with improved StudyFlagCommand now)

Figure out how to exclude the demonstrator from the CIS constant distribution plots. It is always an outlier as it is never to be flagged as BadCIS, but it is always at half of the value of the other channels (only for HG).

* I verified that the lower outlier bump does come from the demonstrator. All 48 channels are around 40-41 in LBA14. To do this I used ReadCalibFromCool.py > CISValues.txt
* Then there are also these 6 following channels, but they are already flagged as BadCIS and there are also not many of them.
  + LBA37 19 40.93
  + EBA15 8 40.689999
  + EBA16 0 41.361607
  + EBA36 15 40.970001
  + EBA64 3 40.41
  + EBC09 40 41.406292
* I tried setting --region flag to everywhere except LBA14 but this doesn’t seem to make it go away.
  + --region EBA EBC LBC LBA\_m00 LBA\_m01 LBA\_m02 LBA\_m03 LBA\_m04 LBA\_m05 LBA\_m06 LBA\_m07 LBA\_m08 LBA\_m09 LBA\_m10 LBA\_m11 LBA\_m12 LBA\_m13 LBA\_m15 LBA\_m16 LBA\_m17 LBA\_m18 LBA\_m19 LBA\_m20 LBA\_m21 LBA\_m22 LBA\_m23 LBA\_m24 LBA\_m25 LBA\_m26 LBA\_m27 LBA\_m28 LBA\_m29 LBA\_m30 LBA\_m31 LBA\_m32 LBA\_m33 LBA\_m34 LBA\_m35 LBA\_m36 LBA\_m37 LBA\_m38 LBA\_m39 LBA\_m40 LBA\_m41 LBA\_m42 LBA\_m43 LBA\_m44 LBA\_m45 LBA\_m46 LBA\_m47 LBA\_m48 LBA\_m49 LBA\_m50 LBA\_m51 LBA\_m52 LBA\_m53 LBA\_m54 LBA\_m55 LBA\_m56 LBA\_m57 LBA\_m58 LBA\_m59 LBA\_m60 LBA\_m61 LBA\_m62 LBA\_m63

# 5. COOL ROBOT tags

<https://atlas-tile-calib.cern.ch/>

* ADCUPD4: update UPD4 only
* ADC\_UPD4\_UPD1: update UPD4+UPD1 simultaneously
* ADC\_UPD1\_ONL: update UPD1 and ONL simultaneously (cannot be used with single sqlite file yet, since very different statuses for UPD1 and ONL)
* ADC\_UPD4\_UPD1\_ONL: update all 3 but avoid atm for similar reasons
* ADCtrigger: update ONL only (needed for trigger during data taking)

# 6. Converting sqlite file tags

Using ~solodkov/scripts/calib\_to\_oracle (untested):

1. upd1='CheckTagAssociation.py --folder=/TILE/OFL02/CALIB/CIS/LIN --globaltag=CURRENTES | tail -1 | awk '{print $NF}''
2. upd4='CheckTagAssociation.py --folder=/TILE/OFL02/CALIB/CIS/LIN --globaltag=CURRENT | tail -1 | awk '{print $NF}''
3. AtlCoolCopy "sqlite://;schema=tileSqlite.db;dbname=CONDBR2" "sqlite://;schema=tileSqlite\_upd4.db;dbname=CONDBR2" -create -folder /TILE/OFL02/CALIB/CIS/LIN -tag $upd1 -outtag $upd4

In the case of CIS/LIN the full UPD1 and UPD4 tags are TileOfl02CalibCisLin-RUN2-HLT-UPD1-00 and TileOfl02CalibLasLin-RUN2-UPD4-16, but next year they'll probably be -RUN3- etc.