Monthly CIS update Guide

Jacky Li and Peter Camporeale

May 2023

Based on Katie’s CIS update Guide

[1. Helpful links](#_e6d1himnao9y)

[2. General procedure](#_sm6q81m7jgpv)

[3. Alternative for LS2](#_qkrtu18jolu5)

[4. Examining high deviation channels](#_26j83327rxi)

[5. COOL ROBOT tags](#_9uyts96plfp3)

[6. Converting sqlite file tags](#_1ye00cto16oh)

# 1. 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.

[failed link] [2018 CIS guide](https://drive.google.com/file/d/1vojCK_cZTw7vfyHZnvRJ4DlovwmpdBey/view?usp=sharing): Detailed overview of CIS tech work, including an explanation of updates and databases.

[failed link] [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

Sasha (Alexander) (Sanya) Solodkov (Sanya.Solodkov@cern.ch) and Henric Wilkens (Henric.Wilkens@cern.ch) and the current RC are primary contacts for CIS updates.

# 2. General procedure

1. **Ssh into lxplus account**

* ssh -XY [USER@lxplus.cern.ch](mailto:USER@lxplus.cern.ch)

Jacky’s note: download Bitvise SSH Client and use it.

I have also tried to configure VScode to do some editing, and it is not worth the effort due to log in issues.

1. **Setup environment**
   1. Athena is the environment we use and it has a bunch of different versions
   2. You can do:

* asetup Athena [VERSION] (our current one is 22.0.46)
* asetup (this sets up the last used version)
* source setup.sh (This is an option for new version of TUCS, will automatically load the most recent version)

1. **Go to TUCS**

* This is where all the macros & workers are

1. **Remove/rename any tileSqlite.db file in TUCS/results directory**

* Otherwise the new database file cannot be created
* Best just to keep the Tucs/results folder clean before you run any code

1. **Run CIS\_DB\_Update.py macro**

* macros/cis/CIS\_DB\_Update.py --date '-28 days' (siphone output using >> to find date)
* macros/cis/CIS\_DB\_Update.py --date 'start' 'end' |&tee results/MonthCIS.txt (|&tee [file] keeps a log about all the output to the [file], I add “results/” so the output will be in result folder along other files. Change the MonthCIS to the month of running the update, such as MarCIS)
* The number of days can change depending on the date of the last update

1. **Check File:**

You now have some files created:

* 1. MonthCIS.txt : the CIS\_DB\_Update.py output log file
  2. tileSqlite.db : this file contains each channel and its newly calculated CIS constant. Will be in ~/Tucs/results
  3. CIS\_DB\_Update.txt : lists the channels in the update, as well as some statistics from the update. Also in ~/Tucs/results
  4. NoCIS.txt (sometimes): list of channels with no CIS response. Usually never needed. In ~/Tucs/results
  5. Plots: located in ~/Tucs/plots/latest/cis/CIS\_Update

1. **Examine Plots:**

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 <path to cis\_charge.C>

Eg.: .L ~.../macros/cis/root\_func/cis\_charge.C

* + 1. 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
  1. 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.
  2. 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.
  3. 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).

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. (Go back to step 5:)

* 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

1. **Prepare for Recalibration:**

~~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~~

(done automatically now in Jacky.py)

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

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

1. **Recalibrate Channels:**

I have created a macro for doing this. CIS\_DB\_recal.py, which once you put the channels you need to recalibrate in a file, will finish the recalibration for you and output the db file.

**ConstantUD:**

**macros/cis/CIS\_DB\_Update.py --date 'February 01, 2023' 'February 28, 2023' --ldate 442620 442775 442791 442820 442935 443018 443364 443414 443831 443835 444001 444007 444633 --recalALL|&tee results/FebCIS.txt**

**create file toRecalibrate.txt in ./results/**

**macros/cis/CIS\_DB\_recal.py --cistxt results/FebCIS.txt --cisupdate|&tee results/Recalib\_output.txt**

**FlagUD:**

**Create a folder FlagUD, then**

**create file RemoveBadCIS.txt and AddBadCIS.txt**

**run the following code in the folder with the txts**

**WriteBchToCool.py --execfile=../../macros/cis/update\_flags.py --online --upd4|&tee Flag\_log.txt**

**<ALL commands default to run in TUCS folder, not any subfolders.>**

1. **Presentation**

[need peter to fill in for his new latex model]

* 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'

1. **Change the comment and tag in the DB file** (highlighted = change this to be the name of your sqlite file)

* Be sure to change the comment of the sqlite file first:
  + Navigate to directory containing WriteDBMultipleIOV.py worker (Tucs/workers/) and edit the comment at the bottom
* Change the tag from UPD1 to UPD4 before uploading

**$ AtlCoolCopy "sqlite://;schema=tileSqlite.db;dbname=CONDBR2" "sqlite://;schema=tileSqlite\_with\_UPD4.db;dbname=CONDBR2" -ot TileOfl02CalibCisLin-RUN2-UPD4-16 -create**

* You can also read the contents of your sqlite file with the following:

**$ ReadCalibFromCool.py --schema='sqlite://;schema=tileSqlite\_with\_UPD4.db;dbname=CONDBR2'**

1. **Push constants to database**
   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~~

This is no longer needed, as they upgraded the db, so tagging is no longger needed.

* 1. scp tileSqlite\_upd4\_CIS\_LIN.db file to your home computer
  2. 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)
  3. \*Upload the upd4 tagged db to CISLIN\_ALL
  4. That’s it!!! Post to elog and exit :)

\*:

* Click on robot icon, then wait
* If nobody is logged in, you will be able to start a new session
* Click on the green CISLIN-ALL button at the button right
* Either upload a SQLite file with the updated constants and UPD4 tag or enter changes manually in the fields provided
* Submit changes and make sure to **exit the session**, otherwise others will not be able to access the Robot

1. **COOL flag status update**

* You will keep track of all the COOL Flag changes (BadCis, NoCis etc)
* There is a python script that /afs/cern.ch/user/a/amattill/public/
* If you can’t access it go to Hazal’s: /afs/cern.ch/user/h/hgoksu/public/TileCalorimeter/TileCalib/Tucs/CIS\_COOL\_UpdatesWr
* Copy that script and edit it, it has instructions
* Test the script by running the below command. It should create an sqlite file if you did everything correct. You can delete that file later  
   **WriteBchToCool.py --execfile=whatever\_you\_name\_thefile.py**
* Once you are sure the script works, upload it to ROBOT.
  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. Make two files, RemoveBadCIS.txt and AddBadCIS.txt and fill accordingly
  4. Copy Update\_flags.py flags directory with the two txt files
  5. Generate your sqlites from these files:
     1. WriteBchToCool.py --execfile=update\_flags.py --online --upd4
  6. Copy the tileSqlite\_upd4.db to your home computer and upload to to ADC\_UPD4\_UPD1 on the robot
  7. All done! Post to elog and exit

############################################################################

[Main note finished]

[below are other old notes that need to be modified or it seems to be can be deleted]

# 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.

# 3. Alternative for LS2

1. **Read db into text file**

Go to results folder in Tucs

* 1. ReadCalibFromCool.py --schema="sqlite://;schema=tileSqlite.db;dbname=CONDBR2" --folder=/TILE/OFL02/CALIB/CIS/LIN --tag=UPD1 | grep -v miss > cis.txt

1. **Make changes to problematic channels**
   1. Can use --region flag in standard update to check changing constants (don’t overwrite database!) (or just use StudyFlag.py as outlined in the [TUCS plots doc](https://docs.google.com/document/d/14RxNsx61A0L8cnDrbXUuIAbqMpTlqutxSXcQyy_V3zc/edit))
   2. Make one file for channels already in update (corr1.txt), one for channels not in update (corr2.txt). Channels that are not in the update will have plots labeled as such. Channels that are in the update can also be seen in the CIS\_DB\_Update file.

i.e.: Drawer Chan Gain Value (don’t include header)

EBA01 1 0 1.27

1. **Use WriteCalibToCool.py to update local db**
   1. WriteCalibToCool.py --schema="sqlite://;schema=tileSqlite.db;dbname=CONDBR2" --txtfile=corr1.txt --folder=/TILE/OFL02/CALIB/CIS/LIN --tag=UPD1 --run=XXXXXX
      1. --run=XXXXXX from CIS\_DB\_update.txt
   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
   3. Check constants with ReadCalibFromCool.py as in 8a
2. **Run calib\_to\_oracle to generate UPD1-, UPD4-, and ONL-tagged databases**
   1. ~solodkov/scripts/calib\_to\_oracle CALIB/CIS/LIN tileSqlite.db
3. **Use ROBOT to upload db**
   1. <https://atlas-tile-calib.cern.ch/>
   2. Upload UPD4-tagged sqlite to CISLIN\_ALL
4. **Use WriteBchToCool.py to update COOL flags**
   1. Change update\_cis.py to reflect desired changes
   2. WriteBchToCool.py --execfile=update\_cis.py --online --upd4
   3. Upload tileSqlite\_upd4.db to ADC\_UPD4\_UPD1

Check the folder, tag, and IOV with AtlCoolConsole.py (?)

# 4. Examining high deviation channels

For channels with high deviation from the databases (>5%), it may be necessary to compare reconstructed laser amplitudes with the CIS constant updates. To look at laser amplitudes, we plot the ratio of signals in the channel of interest against a stable channel in the same module, where a “stable” channel is simply a channel with a signal and CIS constant that was not updated.

1. **Select channel for comparison** 
   1. Compare two odd or two even PMTs
      1. Laser signal sent to all odd or all event PMTs, so ratio should be consistent for same-parity PMTs on the same fibre
   2. For EBA/EBC, use ch. 16 and 17 (D5). If 16 and 17 aren’t stable, use ch. 37 and 38 (D6). If none are stable, it’s okay to use other channels for simple validation, but avoid ch. 0, 1, 12, and 13 (E-cells).
   3. For LBA/LBC, use ch. 13 and 14 (D1). If 13 and 14 aren’t stable, use ch. 24 and 25 (D2).
2. **Double-check your indices and channel numbers!** 
   1. LBA=0, LBC=1, EBA=2, EBC=3
   2. Module number is zero-indexed, i.e. EBA13 -> EBA12
   3. Be careful with channel-to-PMT indexing - check conversion [here](http://zenis.dnp.fmph.uniba.sk/tile.html). We want same-parity PMTs, not channels.
3. **Make plots in ROOT or with /afs/cern.ch/user/e/eoakes/public/show\_las.py**
   1. Show when CIS constants were updated with a line
   2. Use a time frame longer than a single CIS update
   3. Filter laser signals at zero, also use wheelpos==8 for high gain channels
4. **Compare with CIS constants**
   1. Look at the jumps in the laser signal around the CIS update times
   2. If the laser signal returns to stability, good update
   3. If the laser signal deviates more, or the CIS constant is being dragged away from the real value, bad update
      1. Try to adjust the CIS constant by hand as in step 9 above

Alternatively, talk to Henric Wilkens and the laser team about their plots, i.e. <http://hwilkens.web.cern.ch/hwilkens/Tucs/plots/latest/LBA.pdf>. These plots show both gains with response corrected, but are difficult to use because of their size and long date range.

# 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.