Links:

* SBE data processing manual:

<https://www.seabird.com/asset-get.download.jsa?code=251446>

* Sea Bird data processing download (Seasoft V2):

[Sea-Bird Scientific Software - Sea-Bird Scientific (seabird.com)](https://software.seabird.com/)

* Sea Bird community toolkit for python:

*Seabird are currently updating the UI on their software, alongside this they are actively releasing and documenting a bunch of stuff regarding calling their data proc via python. We should look at whether this should be used instead of the current calling method which was written by an external scientific institute (hakai) prior. \*Noting here that the documentation I have seen here mentions lack of support for the instrument models we use.*

[Sea-BirdScientific/seabirdscientific: The Sea-Bird Scientific Community Toolkit (github.com)](https://github.com/Sea-BirdScientific/seabirdscientific)

* Files in:

WQS001.hex – Data file which comes off the instrument

4409\_20020701.xmlcon – instrument configuration file, outlines which sensor packages and calibration coefficients are tied to this CTD at the time of the cast, used to convert .hex data to a user readable .cnv file.

8 x \*.psa files – project setup files for each data processing step, these house information relevant to the processing parameters. These are used to “process” the .cnv file along a string of .cnv steps in this order: Convert – Filter – Align – CellTM – Loop edit – Wild edit – Derive – Bin Av

* Basic current workflow:

Instrument config files and PSA files are located by serial number and date in the config folder – these are updated once per year when CTD is serviced by manufacturer. Raw hex file is placed in the raw folder by user. When process button is clicked, an external database with deployment details is searched for the filename, to gather latitude metadata, if no database is connected or file not found the user is prompted for manual latitude entry. The hex file header info is scraped for a deployment date and a CTD Serial number. These two values are then used to search the config folder for the correct xmlcon file, and respective PSA files. Using these, the file then runs through all 8 processing steps, each time appending the name with a step value, and subsequently using that file for the next step. Once complete, the files used are gathered in a standardised folder structure, and move to a processed area.

* Notes

Database link is not included in github, lets put this to the side for now and manually enter -19 lat.

Current code limitations/todo:

* Button functions:
  + Path choice buttons do not function. Paths need to be defined manually in the config.py file
  + Stop and cancel buttons do not function correctly. – *cancel button hopefully fixed line 352 if derive\_latitude is not None:. This is working to skip the file but creating a layout issue tie in where the define button on the main page below cancel is also clicked at the same time.*
* Assorted
  + Change processed folder to “processing” - complete
  + Change completed to “processed” – complete
  + Code hangs if folders are already in processed / processing folders. This should likely throw an error instead.
  + Add more error prompts –
    - PSA file unreadable
* File streams
  + Currently .psa files are found in config folder, used, then copied into cast folder in “*processing*”, then moved to completed folder at the end. I would like these to be copied into the cast folder in “*processing*”, used from here, then moved to “*processed*”. This is so that if we adjust the .psa file parameters during the proposed QAQC steps the change is only applied to this one cast, and then carries through with folder structure.
  + Folder to only move from processing to processed if full processing is done and user has confirmed happy with the data. Install prompt prior to moving files.
    - Change so that .hex file is moved from raw folder, not copied (once above is done)
  + Future addition for processed folder to have a further layer for sampling trip number (processed/Trip\_8010/wqs001…)
    - This would require a call to the oceanDB to find what trip number cast was on.
* Processing step selection
  + Build in system to select which dataproc steps to complete with tick boxes. Default to be all.
  + Build in system where if hex file in raw folder already has a folder in “processing”, dataproc steps selected will pick up from here and replace downstream files rather than starting from raw.
* Data report
  + After bin\_av step, code to pause and give user metrics on the bin\_av file.
    - Start with are there any negative values.
    - Scan range for each bin
    - More metrics to follow
  + If user unsatisfied with report, user to adjust .psa files
    - This could be done manually or in GUI. Decision to be made.
    - File potentially to be removed from processing folder if user clicks no to satisfactory test
      * Issue with this is that .psa file needs to be adjusted, and psa files are a general use file until they are copied into processing folder and used. This is an argument to leave files in processing folder, and allow the option for reprocessing once .psa files have been changed.
  + At this point it would be good to see a plot of the data as well
* Version control log
  + Script to continually update a version control log,
    - File in
    - Config file used
    - Dataproc steps run
    - Parameters used? Scrape from bin\_down cnv file with line search
    - More to come
* There is a train of thought to have a system where the code gathers a bunch of information prior to starting any processing like Roxanas did. It would look at all the files in the “waiting to be processed” folder, and cross reference with oracle database looking for metadata, as well as in the completed folder to see if they had already been done. Once metadata was entered in the database, it would then process only these casts. – more to come
* Heavily document each step throughout code
  + JM is fine navigating, altering and writing small amounts of python code, but not power-user enough to create this whole thing properly. A point to note is that over time things like config file headers, instrument models ect change. We want this script created in a way that does not require us to bug you for little changes that seabird make or simple adjustments like naming convention changes ect.