

# PXD Software Overview

Harrison Schreeck

*2<sup>nd</sup> Institute Of Physics, Georg-August-Universität Göttingen*

# Basics

- We are using git to work on our software. It is a good idea to read a bit about the functionality of git so that you know how to use it, there are many guides available.
- Our software is spread over multiple repositories, if you are working on a computer where the install script was executed, they should be at these places:
  - `pxd_sc_lab_framework` → `~/lab_framework`
  - `pxd_sc_dhh` → `~/dhh`
  - `pxd_sc_testsetup` → `~/epics`
- You can clone them via ssh or https:  
`git clone ssh://git@stash.desy.de:7999/b2g/pxd_sc_lab_framework.git`  
`git clone https://username@stash.desy.de/scm/b2g/pxd_sc_lab_framework.git`
- Most of our scripts are in `lab_framework` but some of the configuration files are in `~/epics` and some support software is in `~/dhh`

## Basics (2)

- Content of ~/epics
  - **utility\_ioc.py**: This is a IOC that was developed for the labs to provide useful functions that would otherwise require to execute scripts by hand. It's most important functionality today is the automatic temperature measurement
  - **start\_epics**: This script is executed in the labs to start all necessary IOC to operate a PXD9 or Hybrid5 module. At DESY and KEK the IOCs are usually started via dedicated scripts!
  - **mergeIntoDB.py**: Script to merge a xml file with a commit in the ConfigDB.

# Basics (3)

- Content of ~/dhh
  - **dhh\_support\_sw/**: In this folder you find the subfolder dhh\_daq and pyDepfetReader. The former is our local daq which can be recompiled by executing the *build.sh* script. The latter is a python program which allows us to read data which was recorded using the dhh\_daq.
  - On most systems you will work the daq and the reader will be installed already. If there is a update/bugfix you might have to recompile or reinstall them:
    - `bash ~/dhh/dhh_support_sw/build.sh`
    - `pip install --user pyDepfetReader/` (The '/' is important!)

# Basics (4)

- Content of ~/lab\_framework
  - Here nearly all of our scripts are located, all of them have a common structure with 3 scripts:
    - measure.py to record data
    - analysis.py to analyze previously recorded data
    - update.py to update either the system or the ConfigDB
  - calibrations/pedestal/
    - Here you find scripts to measure, analyze and upload pedestals
  - calibrations/delays/
    - Here you find scripts to optimize the DHP-DCD data links
  - calibrations/hs\_link\_scan/
    - Here you find scripts to optimize the DHP-DHE highspeed data links
  - calibrations/offsets\_calibrations/
    - Here you find scripts to optimize the 2-Bit DAC pedestal offsets
  - ....many more

# DHE vs DHC setup

- In the labs we usually operate the system with a “simple” readout chain:
  - Module → DHE → PC
- At KEK and DESY it is more complicated due to the DHC:
  - Module → DHE → DHC → PC
- The addition of the DHC makes data taking a little bit more complicated and the scripts that work with the “DHE only” setups are not compatible. With the beginning of phase 2 we started to restructure our scripts. We did this in a separate branch, called “BEAST”.
- The big challenge for phase 3 is to bring these branches back together. Work is ongoing here in the Branch “feature/beast\_integration\_into\_master”
  - Scripts in this branch should support both modes of operation DHE only and DHC!

## Software details

- Data storage format:
  - DHC: /BASEPATH/my\_scan/2018\_19\_02\_001/H51rawframe\_data.dat
  - DHE: /BASEPATH/my\_scan/W11\_OF2/2018\_19\_02\_001/rawframe\_data.dat
- Each computer needs to have a master.setup.HOSTNAME.ini located in  
~/lab\_framework/configurations/

```

mastersetup.ini

[default]
setup = BEAST-II

[BEAST-II]
user = PXD
dhe = None
ps = None
useDHC = True
dhc = H51,H52
triggerDHC = None
useExternalTrigger = True
basepath = /data/BEAST

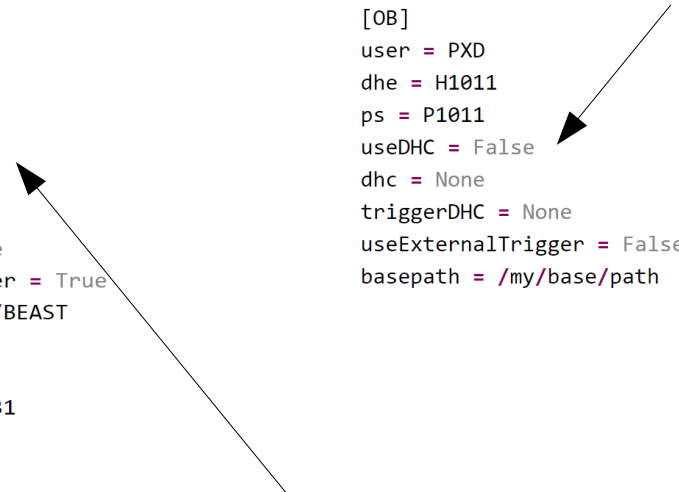
[H51]
dhes = H2131,H1131
ps = P2131,P1131
daqport = 6001

[H52]
dhes = H2132,H1132
ps = P2132,P1132
daqport = 6002

[OB]
user = PXD
dhe = H1011
ps = P1011
useDHC = False
dhc = None
triggerDHC = None
useExternalTrigger = False
basepath = /my/base/path
  
```

DHE mode

DHC mode



# Software details (2)

- Data ports:
  - Each DHE/DHC send the data to a specific port (can be adjusted via a PV). In the lab we usually only use port 6000 for the DHE
  - When handling with multiple DHCs one has to be careful and assign individual ports. For simplicity there is a function in `~/lab_framework/lib/dhc_utils.py` to get the port for a given dhc

```
def get_dhc_ports(dhc, nomenclature="phase3"):
    """
    Helper function which returns the right ports for a given DHC and nomenclature.
    The default nomenclature is phase3 and is also the most relevant. For backward compatibility the phase2
    nomenclature is also supported.

    :param dhc: DHC name like 'H51'
    :type dhc: string
    :param nomenclature: nomenclature which shall be used, either 'phase2' or 'phase3'
    :type nomenclature: string

    :return: inputport, dataport, controlport
    :rtype: int, int, int
    """
```



# Measurement Scripts

- Measurement scripts are all very similar, all of them
  - load some configuration from a local ini file (measure.ini) and the global master.setup.ini → ‘merged’ configuration
  - if necessary get information about the current system
  - Create a path (with timestamp!) for the data
  - Save a copy of the merged measure.ini in the data path
  - Create a PVDump and save it
  - change settings on the system
  - record data
  - restore original settings (this is very important!)
- To ensure that a measurement script is compatible with all variations of setups we have worked out a basic structure which all scripts should ideally follow
  - Full description here: <https://confluence.desy.de/display/BI/Script+Development>

# Analysis Scripts

- Analysis scripts are also all very similar, all of them
  - load some configuration from a local ini file (analysis.ini) and the measure.ini the was stored together with the data
  - Perform some sort of analysis on the data
    - The analysis script should be able to handle the analysis of multiple modules, if possible one process for each module should be spawned (but make sure that this does not overload the computer!)
  - Store the results as a dictionary in a file (analysis.npy)
    - If the analysis found optimal settings for some PVs, the dictionary should contain the PV name as the *key* and the optimal setting as the *value*

# Update Scripts

- Update scripts are also all very similar, all of them
  - Load the analysis results from the analysis.npy file
  - Update either the system or the ConfigDB
- Most update scripts are pretty simple as they usually have to set PVs (if updating the system) or create a new commit in the ConfigDB. The latter is rather easy as there is a library function which will just update all PVs specified in the analysis.npy file
- All update scripts have the following arguments:
  - --system → update the system
  - --db → update the ConfigDB
  - --file → location of the analysis.npy file
  - --commitid → ID of the commit in the ConfigDB you want to update

# ELOG

- All of our measurement and analysis scripts create elog entries that are usually submitted to the elog hosted @DESY
- There is an elog class defined in `~/lab_framework/lib/elog.py` which makes it (relatively) easy to create these entries
- The name of the logbook and the address of the server have to be defined in the `[elog]` section of the `master.setup.ini`
- To access the elog one can either create an ssh-tunnel (requires your DESY credentials)
  - `autossh -M 0 -o "ServerAliveInterval 30" -o "ServerAliveCountMax 3" -L 12345:b2-elog1:8080 username@bastion.desy.de`
- Or you can connect directly, but then you have to enter your credentials each time you submit an entry

```
[elog-old]
hostname = elog.belle2.org
logbook = PXD-Mass-Testing
port = 12345
https = False
useCurl = False
sshtunnel = True
```

```
[elog]
hostname = elog.belle2.org
logbook = elog/PXD-Mass-Testing
port = 443
https = True
useCurl = True
sshtunnel = False
```

# ELOG

- Details on how to integrate the elog class into your measurement/analysis script can be found here:
  - <https://confluence.desy.de/display/BI/Automated+ELOG+entries>
- For the currently existing scripts there is not much work to do, the elog class is adjusted for all of them.

## Example

```
if __name__ == "__main__":
    # Start of the script

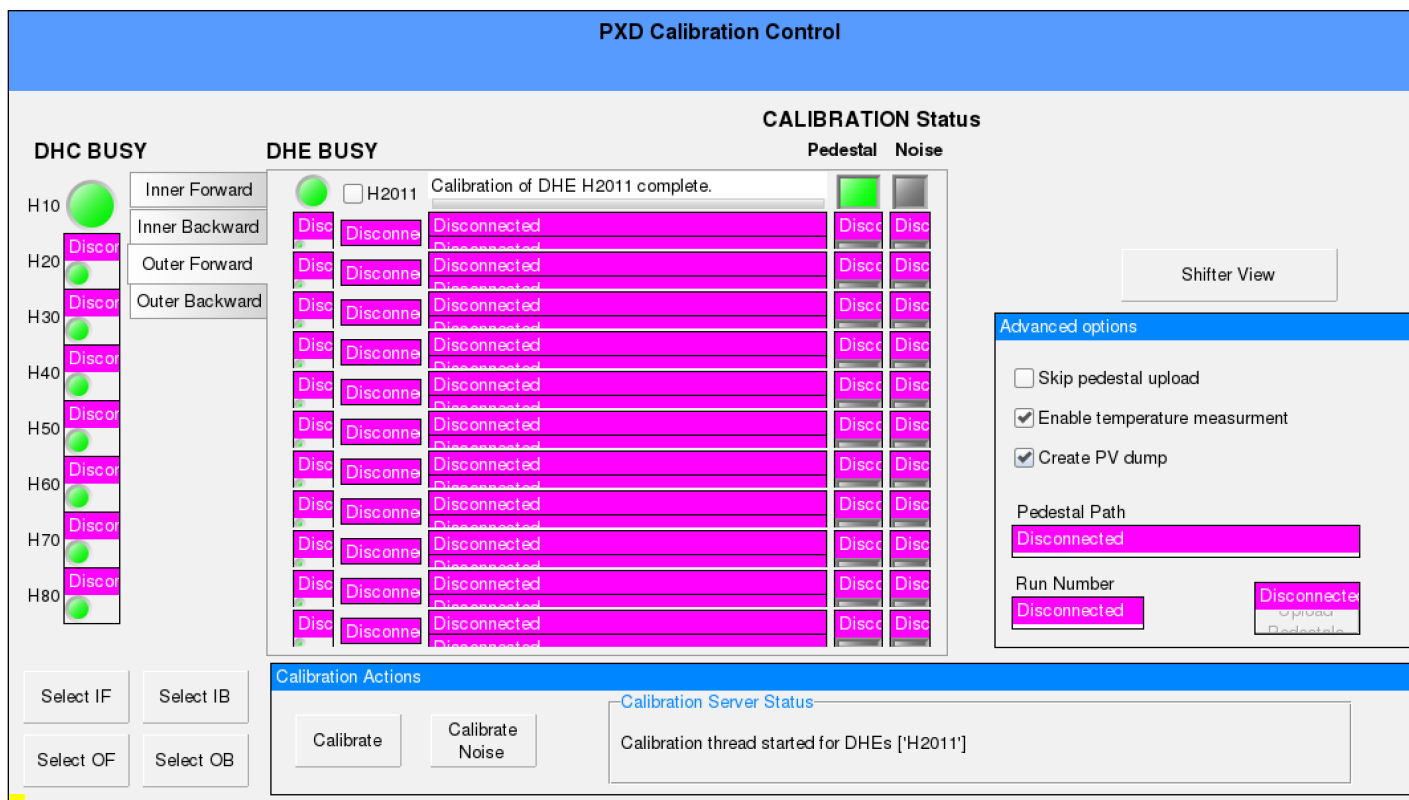
    # ELOG
    try:
        if useCredentials:
            credentials = elog.credentials()
        else:
            credentials = elog.credentials(noinput=True)
        myelog = elog.elog(configfile=settingsini, type="Delay Scan", username=credentials.getUsername(), password=credentials.getPassword())
    except Exception as e:
        print repr(e)
        print "ERROR WITH ELOG"

    # Do Measurement/Analysis

    # ELOG
    try:
        myelog.submitEntry(path="path/to/files")
    except Exception as e:
        print repr(e)
        print "ERROR WITH ELOG"
```

## Calibration IOC

- The calibration IOC (located in ~/lab\_framework/Beast/) was created to make it very easy for a non-expert to calibrate a large number of modules
- Currently the IOC allows the user to calibrate the pedestals of the modules
  - Via a GUI one can specify the modules/DHEs and start the calibration
  - The connected DHCs are then “blocked” and cannot be used for another calibration



**PXD Calibration Control**

**CALIBRATION Status**

DHC BUSY      DHE BUSY      Pedestal      Noise

H10    Inner Forward     H2011    Calibration of DHE H2011 complete.       

H20    Inner Backward    Disc    Disconne    Disconnected    Disc    Disc

H30    Outer Forward    Disc    Disconne    Disconnected    Disc    Disc

H40    Outer Backward    Disc    Disconne    Disconnected    Disc    Disc

H50                       Disc    Disconne    Disconnected    Disc    Disc

H60                       Disc    Disconne    Disconnected    Disc    Disc

H70                       Disc    Disconne    Disconnected    Disc    Disc

H80                       Disc    Disconne    Disconnected    Disc    Disc

Select IF    Select IB

Select OF    Select OB

Calibrate    Calibrate Noise

Calibration Server Status

Calibration thread started for DHEs ['H2011']

Advanced options

Skip pedestal upload

Enable temperature measurement

Create PV dump

Pedestal Path

Disconnected

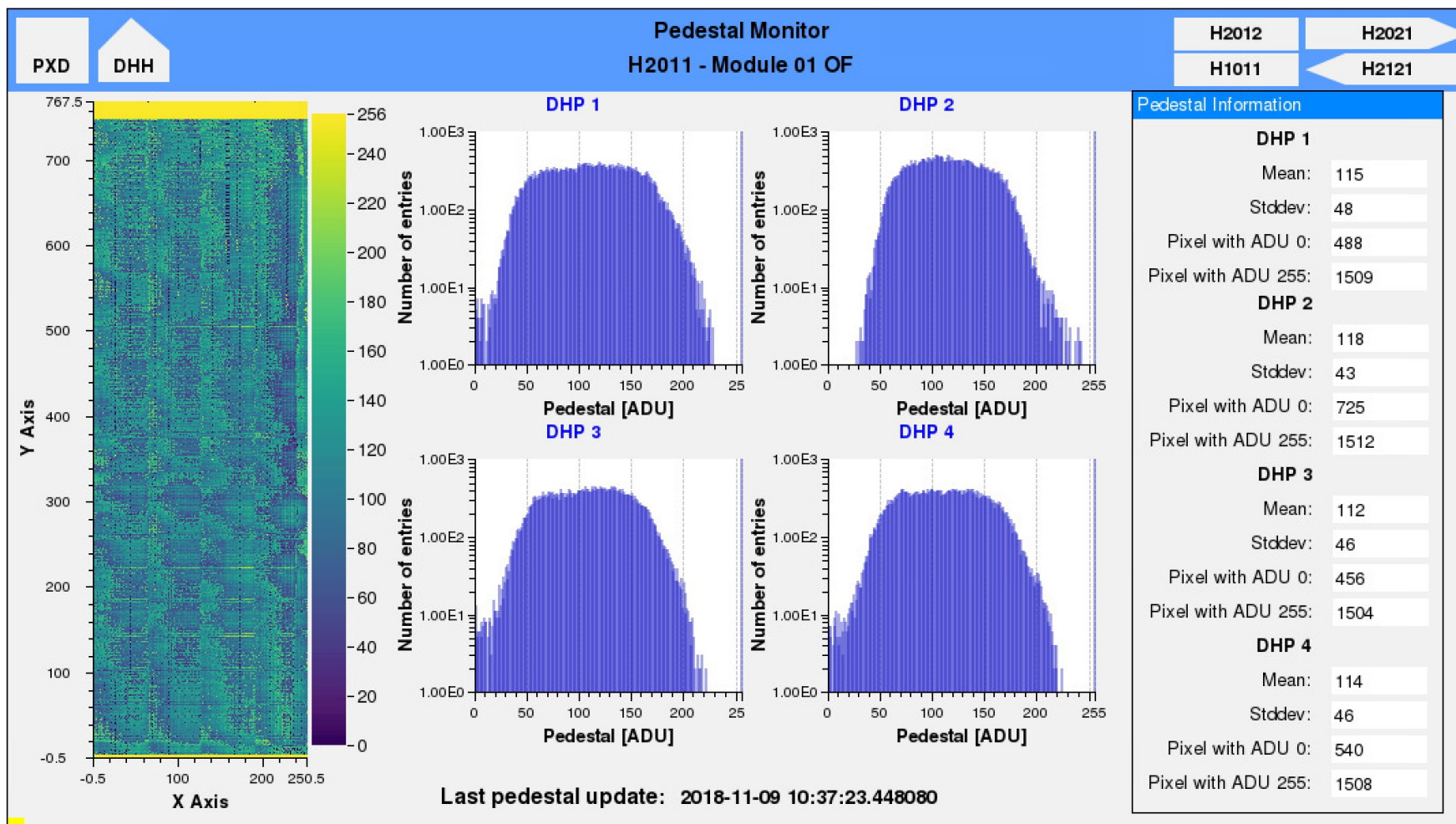
Run Number

Disconnected    Disconnect

Shifter View

# Calibration IOC

- By clicking on the pedestal LED one gets to the pedestal monitor opi. Here the latest uploaded pedestals are shown. **Important:** The pedestals shown here might be wrong if a manual upload via command line has been performed!



# Calibration IOC

- The IOC can be started via
  - `python ~/lab_framework/Beast/calibrationIOC.py --setup phase3`  
where “setup” has to be defined in the “master.setup.ini”. Based on this setup the DHC/DHE mapping is determined
- The IOC will print a graphical representation of the system during startup, here you can check if everything is right

```
[hschreeck@localhost Beast]$ ./calibrationIOC.py --setup DHC-Demo
ELOG user credentials (DESY Account):
Username:
Password:
2018-11-11 17:35:16,903 elog::test_credentials [WARNING]: These credentials are not correct for elog.belle2.org!
2018-11-11 17:35:16,998 calibrationIOC::main [INFO ]: Calibration IOC started for setup DHC-Demo.
H10
|---- H2011
|---- H1011
|---- H2012
|---- H1012

H20
|---- H2021
|---- H1021
|---- H2022
|---- H1022

Basepath set to /run/media/belle2/DATA2/measurements/
```



## elog\_server.py

- The IOC with the worst name, it was created to automatically create and submit elog entries for the runs in phase2 based on the global Belle2 Status (there is PVI!)
- In addition it automatically starts/stops our local DAQ

PXD DAQ Control

<b>DHC H10</b> status: CLOSED path: <input type="text"/> control port: 32767 UDP port: 6000 data port: 20248 file name: H10data.dat Exit Init Start Stop	<b>DHC H20</b> status: <span style="background-color: #ff00ff; color: white;">Disconnected</span> path: <span style="background-color: #ff00ff; color: white;">Disconnected</span> control port: <span style="background-color: #ff00ff; color: white;">Disconnected</span> UDP port: <span style="background-color: #ff00ff; color: white;">Disconnected</span> data port: <span style="background-color: #ff00ff; color: white;">Disconnected</span> file name: <span style="background-color: #ff00ff; color: white;">Disconnected</span> Exit Init Start Stop	<b>DHC H30</b> status: <span style="background-color: #ff00ff; color: white;">Disconnected</span> path: <span style="background-color: #ff00ff; color: white;">Disconnected</span> control port: <span style="background-color: #ff00ff; color: white;">Disconnected</span> UDP port: <span style="background-color: #ff00ff; color: white;">Disconnected</span> data port: <span style="background-color: #ff00ff; color: white;">Disconnected</span> file name: <span style="background-color: #ff00ff; color: white;">Disconnected</span> Exit Init Start Stop	<b>DHC H40</b> status: <span style="background-color: #ff00ff; color: white;">Disconnected</span> path: <span style="background-color: #ff00ff; color: white;">Disconnected</span> control port: <span style="background-color: #ff00ff; color: white;">Disconnected</span> UDP port: <span style="background-color: #ff00ff; color: white;">Disconnected</span> data port: <span style="background-color: #ff00ff; color: white;">Disconnected</span> file name: <span style="background-color: #ff00ff; color: white;">Disconnected</span> Exit Init Start Stop	<b>All DHCs (combined)</b> status: CLOSED path: <input type="text"/> control port: 32767 UDP port: -1 data port: 20248 file name: data.dat Exit Init Start Stop <input checked="" type="checkbox"/> Follow RunControl <input checked="" type="checkbox"/> Record Pedestals RC status: <span style="background-color: #cccccc; color: black;">RUNNING</span> RC exp. number: 1 RC run number: 666
<b>DHC H50</b> status: <span style="background-color: #ff00ff; color: white;">Disconnected</span> path: <span style="background-color: #ff00ff; color: white;">Disconnected</span> control port: <span style="background-color: #ff00ff; color: white;">Disconnected</span> UDP port: <span style="background-color: #ff00ff; color: white;">Disconnected</span> data port: <span style="background-color: #ff00ff; color: white;">Disconnected</span> file name: <span style="background-color: #ff00ff; color: white;">Disconnected</span> Exit Init Start Stop	<b>DHC H60</b> status: <span style="background-color: #ff00ff; color: white;">Disconnected</span> path: <span style="background-color: #ff00ff; color: white;">Disconnected</span> control port: <span style="background-color: #ff00ff; color: white;">Disconnected</span> UDP port: <span style="background-color: #ff00ff; color: white;">Disconnected</span> data port: <span style="background-color: #ff00ff; color: white;">Disconnected</span> file name: <span style="background-color: #ff00ff; color: white;">Disconnected</span> Exit Init Start Stop	<b>DHC H70</b> status: <span style="background-color: #ff00ff; color: white;">Disconnected</span> path: <span style="background-color: #ff00ff; color: white;">Disconnected</span> control port: <span style="background-color: #ff00ff; color: white;">Disconnected</span> UDP port: <span style="background-color: #ff00ff; color: white;">Disconnected</span> data port: <span style="background-color: #ff00ff; color: white;">Disconnected</span> file name: <span style="background-color: #ff00ff; color: white;">Disconnected</span> Exit Init Start Stop	<b>DHC H80</b> status: <span style="background-color: #ff00ff; color: white;">Disconnected</span> path: <span style="background-color: #ff00ff; color: white;">Disconnected</span> control port: <span style="background-color: #ff00ff; color: white;">Disconnected</span> UDP port: <span style="background-color: #ff00ff; color: white;">Disconnected</span> data port: <span style="background-color: #ff00ff; color: white;">Disconnected</span> file name: <span style="background-color: #ff00ff; color: white;">Disconnected</span> Exit Init Start Stop	

**Local Test Control**

RC status: READY

RC exp. number:

RC run number:

# elog\_server.py

- In the BEAST branch many settings are hard-coded for the phase2 setuo and there is a special configutaion file for the elog\_server
- In the feature/beast\_integration\_into\_master branch the elog\_server loads the configuration (like the CalibrationIOC) from the master.setup.ini
- The IOC is started with
  - `python ~lab_framework/runcontrol/Elogsever/elog_server.py`
- If the `--local` flag is used, it will not listen to the global Belle2 Runcontrol PVs, but to its own
  - `PXD:ELOG:State2:cur:S`
  - `PXD:ELOG:ExpNumber:I`
  - `PXD:ELOG:RunNumber:I`
- This can be useful to perform some local tests but using the full functionality of the `elog_server` (automatic logging, starting/stopping the DAQ,...)

# Script Status

- Unfortunately we currently have 3 different branches of the lab\_framework that are in use
  - **Master:** The branch that was/is used in the labs. All scripts in this branch have been tested and work with single module DHE setups. This branch wasn't updated in quite some time.
  - **BEAST:** This branch was created during the phase2 commissioning. Some of the scripts have been adapted to work with DHC setups. This branch also contains the calibrationIOC and the elog\_server
  - **feature/beast\_integration\_into\_master:** This branch was created to bring the master and the BEAST branch back together. This branch is currently under development. We should try to get all scripts in this branch ready for DHE and DHC usage! The calibration IOC in this branch has already been prepared for the phase3 system as well as the pedestal script.

# Script Status

•

	master	BEAST	feature/beast_integration_into_master
pedestal	DHE	DHC	DHE
DHE-DHP HS links	DHE	DHC	DHC
DHP-DCD delays	DHE	DHC	DHE
2-Bit DAC delays	DHE	DHE	DHE
2-Bit DAC offsets	DHE	DHC	DHE

# Script Testing

- At least the measurement and upload scripts require a system to test and debug them on
  - DHE testing: → possible in the labs
  - DHC testing: → only possible at DESY and KEK
- I would propose to test the most important scripts (pedestal, delays, ...) as soon as possible and then merge the feature branch into the master. Other scripts (for example ADC curves) can be adjusted later.
- We have too many open/forgotten feature branches, we should try to avoid this in the future

## Important Links

- General coding guidelines:
  - <https://confluence.desy.de/display/BI/Coding+Guidelines>
- Script development guide:
  - <https://confluence.desy.de/display/BI/Script+Development>