ITKPIXV1 – PIXEL CHIP
BDAQ53 – A COLLABORATIVE TESTING SYSTEM

WHAT IS THE AIM OF BDAQ?

→ Collaborative, Modular, Understood
• Code Style according to PEP8 → Recommendation (set this up in your IDE E.g. Visual Studio Code)

• Stick to your minimal set of changes → many small merge requests are better than one big one

• Ensure most used functions to still be working

• Four eye principle → Get feedback from other contributors
• Setup PEP8 in your VISUAL STUDIO CODE ENVIRONMENT
→ Will manage spacing, and code style for you when pressing Ctrl+Shift + P

Test your code style with the following command and get a detailed report:

```
pip install flake8-diff
flake8-diff origin/development
```

```
449 $ flake8-diff origin/development
450 Found violations: localdb/manage_localdb.py
451 E713 @ 81:12 - test for membership should be 'not in'
452 W293 @ 125:1 - blank line contains whitespace
453 E303 @ 126:9 - too many blank lines (2)
454 E303 @ 187:5 - too many blank lines (2)
455 E128 @ 236:29 - continuation line under-indented for visual indent
```
Often empty spaces, empty lines, etc. are committed to a repo complicating your changes readibility

→ Easy check to be conducted here
ENSURE OLD CODE TO STILL BE WORKING

→ CI ensures long term stability, functionality and compatibility of BDAQ53
BDAQ53 CONTINOUS INTEGRATION (CI)

→ Two full fletched daq and dut simulations upon every commit deployed on servers in Bonn

→ Fully virtualized with docker

→ Full firmware compilation, automated flashing and scan running on real bdaq hardware
BDAQ53 CONTINOUS INTEGRATION (CI)

Test analysis:
• Ensure that new analysis output still corresponds to the old standard

Test EUDAQ:
• Ensure API compatibility with EUDAQ

Test Scans:
• Run scans consecutively with a rd53a/b mock to check for runtime errors.

Test Software:
• Ensure general BDAQ API compatibility and check for memory leaks

→ Fully virtualized with docker
Simplest case: Add your scan to the scan folder in the bdaq repo

```bash
pytest --cov=bdaq53 bdaq53/tests/test_software/scans/test_scans.py
```
ADD FUNCTIONS TO THE CHIP API

```python
def get_temperature_from_ADC(self, dADC, sensor):
    # ToDo: Calibrate diode temperature sensors
    return round((self._dV_to_T(self.get_V_from_ADC(dADC, use_offset=False), sensor=sensor)), 3)

def get_temperature_from_ADC_resistive_sensors(self, dADC, sensor):
    # ToDo: Calibrate resistitive temperature sensors
    return dADC
```

→ Have a new function for the chip?
→ Add it to the chip class
→ Checks functionality in corresponding scans and ensures long term support
Does it remove versatility, does it add to modularity, are all observed effects understood?

The more detail, the easier to understand, the easier to merge
THE BDAQ WORKFLOW

You find a bug or envision a new feature

Document as Gitlab Issue → reference code in repo

Get feedback

Contribute

Especially during corona contributor location is irrelevant!
BDAQ is:

- a team effort
- setup for easy contribution
- Open for maintaining code according to the modular and versatile idea of the readout.

Especially during corona contributor location is more irrelevant then ever