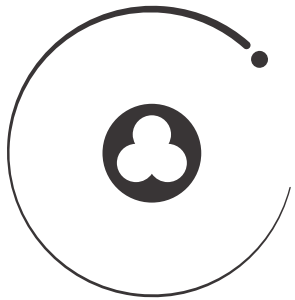
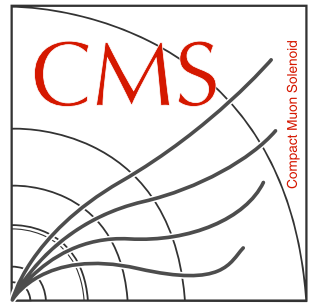


HELSINKI
INSTITUTE OF
PHYSICS



TOTEM GEM tracker readout & new SRS developments

RD51 mini-week

L. Forthomme F. Garcia

Helsinki Institute of Physics

February 15-19, 2021

TOTEM GEM tracker in a nutshell

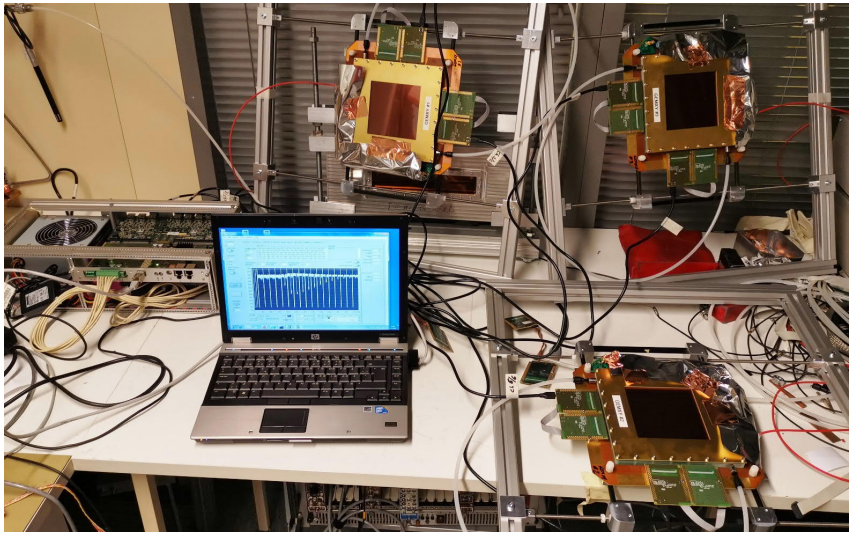
TOTEM-PPS Collaboration in need for a **telescope** for its **R&D programme**:

- characterisation of **detectors efficiency and uniformity** (+ timing precision) at CERN SPS-H8
- wide range of DUT technologies: scintillator tiles, scCVD sensors, LGADs, ...

Basic requirements

1. ~1 kHz **counting rate**
2. **Spatial resolution** from 10 μm – 100 μm
3. **Flexibility** for improvements/additions
   - e.g. event builder integration of MCP-PMT for timing applications
4. Portability
5. Use of already developed **DAQ**
6. Use of already existing **tracking reconstruction**
7. Reasonable cost

TOTEM-Helsinki triple-GEM detector



Pros

- 10×10 cm^2 area coverage
- **Spatial resolution** ~60 μm
- DAQ/tracking reconstruction exist
  ✗ DATE/AMORE software chain **not maintained anymore**
- Hardware available at CERN store, APV25 hybrids, plenty of support from RD51
- Low cost

Cons

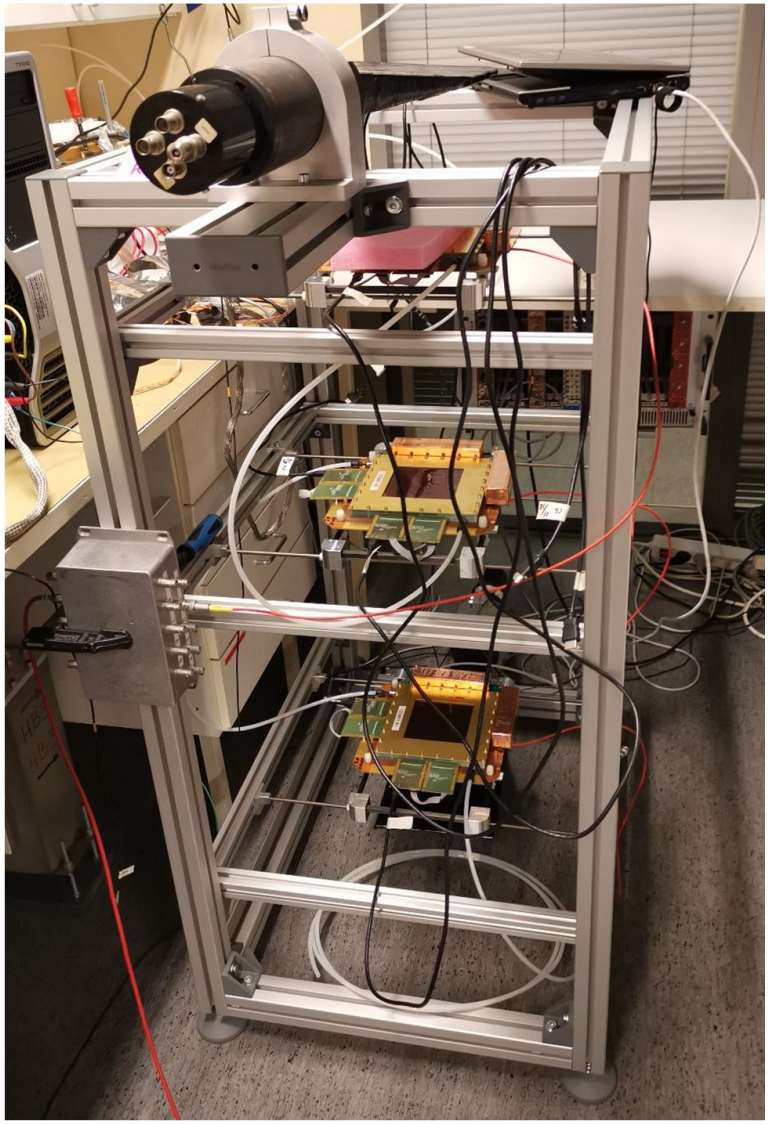
- **Requires gas** $ArCO_2$ (70/30)
- Not easy to **transport**
  ✓ for "permanent" installation in TOTEM-PPS experimental area at SPS-H8

TOTEM-Helsinki triple-GEM detector - current status

Built in Helsinki, ready to be shipped to CERN

- **mechanical integration** of XY planes, muon counters, and overall frame ready
- SRS FEC and APV25 hybrids **electronics tests** successfully passed

Timeline for first operations: mid-March



F. Garcia

SRS driver

Given the retirement of ALICE's DATE, developed a library handling all parts of UDP communication with SRS module, and object-oriented definition of all registers.

- "Standard stack", `C++14` & `CMake`, designed for **portability**
  - core code relies on UNIX socket API and requires **no external dependency**
  - single shared object + headers library, **easily imported** elsewhere, heavily **documented**
- Own implementation of the `slow_control` executable with more debugging tools, dry-run mode, ...
  - library has full parsing/writing control over `.txt` configuration files, can write its own (e.g. to store per-run configuration)
- (Boost) **Python binding** for quick scripting
  - slow control implemented, readout to follow

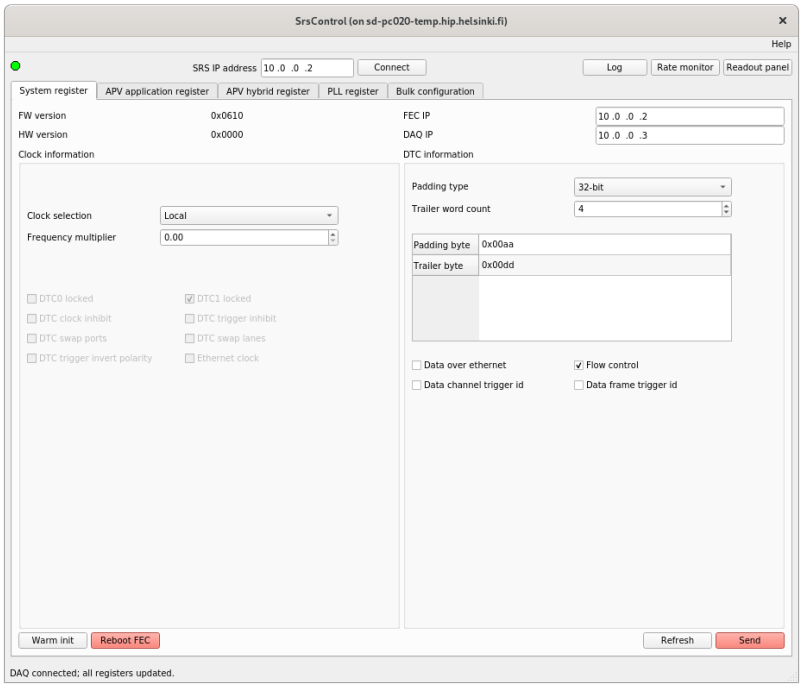
```
import pysrsdriver as srs

sc = srs.SlowControl('10.0.0.2')
apvapp = sc.readApvAppRegister()
apvapp.triggerDelay = 0x100 # BCKL_TRGDELAY
apvapp.triggerSeqPeriod = 40000 # BCKL_FREQ
# (...)
```

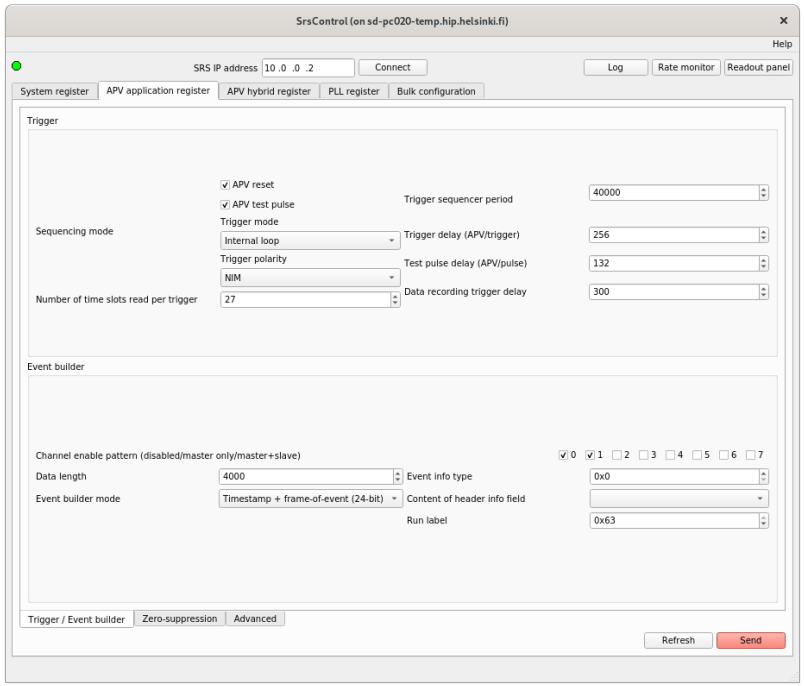
- Currently supported readout mode: APV25, zero-suppressed mode development ongoing
  - unpacking part factorised from slow control, framework to **define other data formats**
- Released under **GPLv3**, currently accessible with a CERN NICE account:
  - /lforthom/srsdriver

SRS control

Qt5 **graphical user interface** with the SRS driver

- developed for the **testing and validation** of this latter
- all communication/readout is operated through the driver, only Qt as external requirement
- potential candidate for the **slow control operations** in TOTEM test beam control room

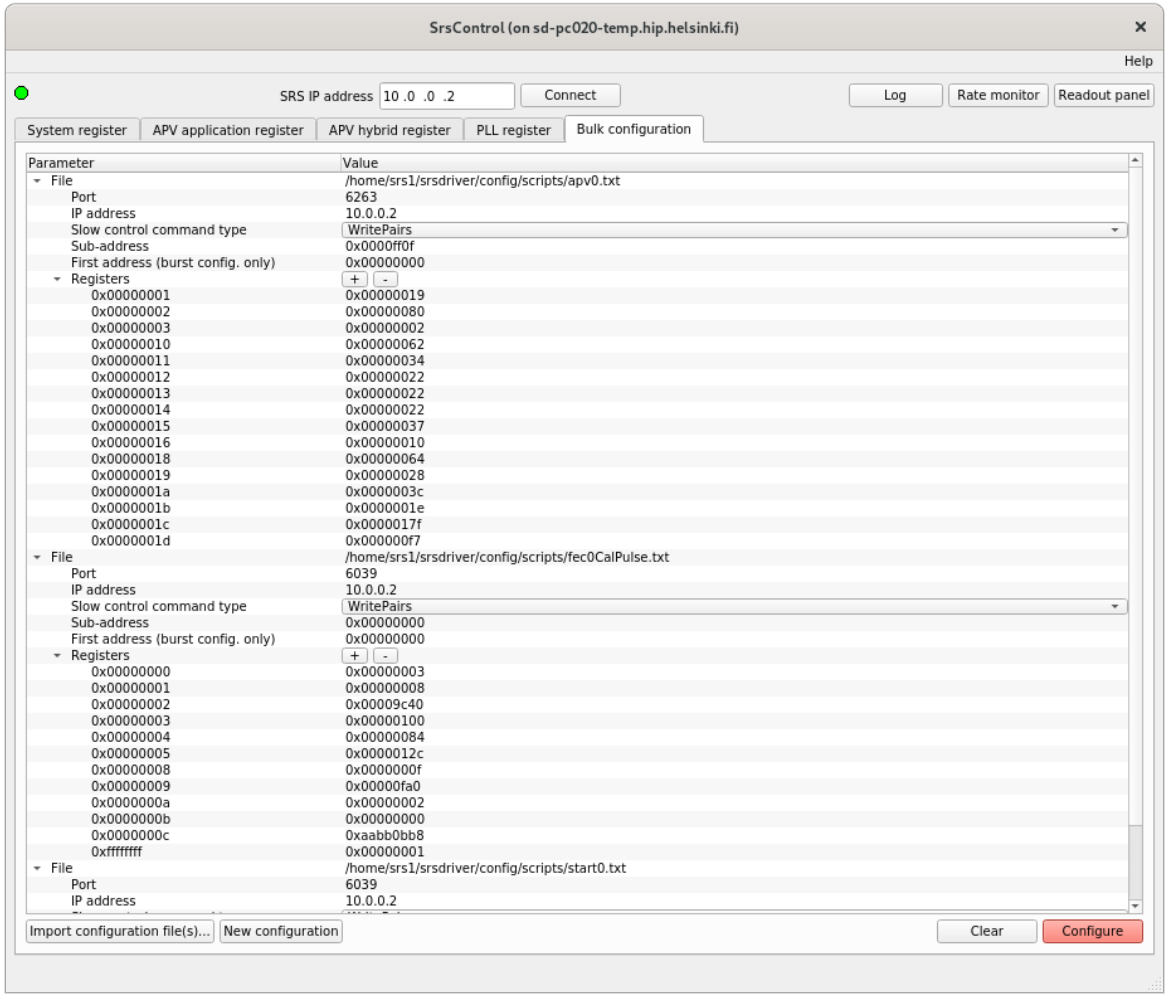




- /lforthom/srscontrol (users with CERN NICE account)

SRS control - a few features

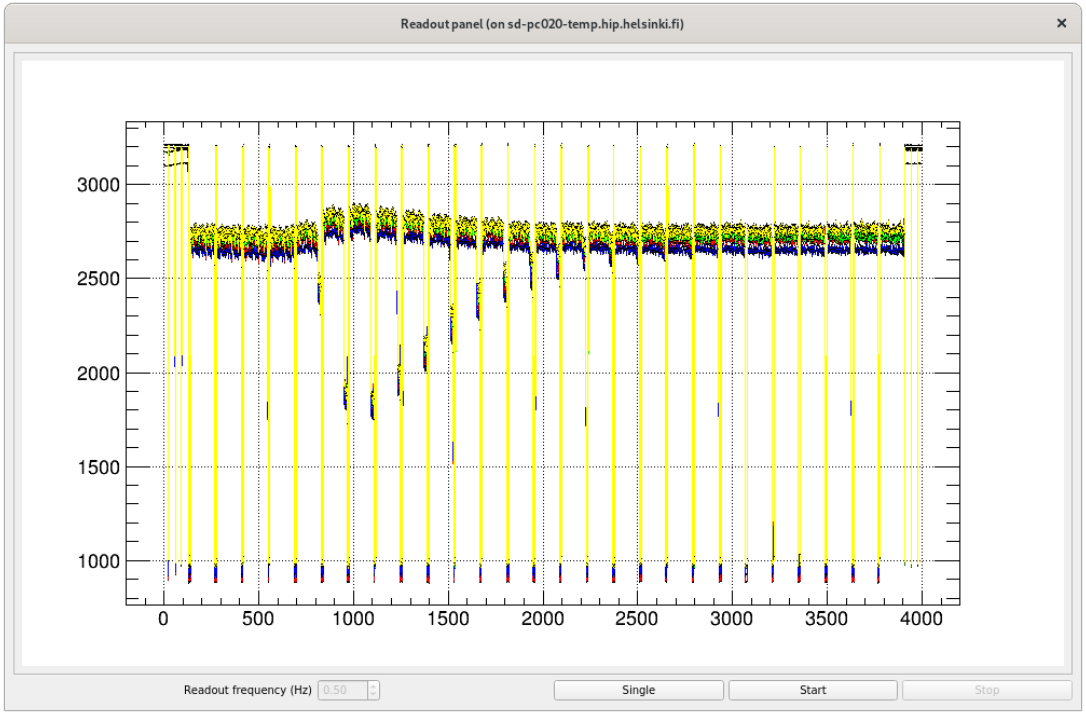
Trivial **editor** developed to load/edit/upload external `.txt` **configuration files**

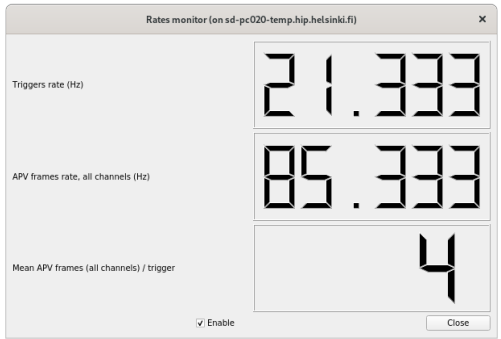


- **sequential storage** for later usage to be implemented (`.tar`?)

SRS control - a few features (cont.)

Easy debugging of configuration through **monitoring** of full frames and rates





- earlier is **ROOT-based**, can be easily extended with more advanced online features (fits, pedestal subtraction, . . .)
- many **more rate monitors** to be added, when more unpacking modes are handled

Slow control and data acquisition

SRS driver flexibility, one single library required for slow control and readout components

- already integrated readout part into a producer for EUDAQ2
  - optionally steer slow control through input of sequenced `.txt` configuration files, or rely on SRS control GUI for prior configuration
  - currently limited monitoring tool, access to full frame, coarse APV unpacking, still a work in progress
- possibly write a wrapper for CMS' XDAQ, or others
  - again, full documentation of API is available for users applications

Tracking and event reconstruction

- given its ongoing development effort, Corryvreckan is a good candidate:
  - EUDAQ2 unpacking already validated,
  - only requires to define APV25 data format and GEMXY + DUT geometries
  - alignment/clustering/tracking (incl. GBL) algorithms already implemented, steering configurations to be implemented

Conclusions

Parallel effort ongoing for the construction, characterisation, and operation of a new triple-GEM tracker for TOTEM test beam area at H8

- **sensors** already characterised, **full readout chain** to be tested with SRS
- new implementation of a **software toolbox** to ease communication with SRS motherboard (slow control and fast readout)
- new **graphical tool** to help setting up a **run configuration**
- implementation of the **readout part** into e.g. EUDAQ2 as a proof of concept for operations at H8
- **future developments** foreseen for the reconstruction package and commissionning tools (alignment/efficiency mapping/...)

Backup

Calibration pulse

Calibration Pulse:

Injection of four different charge amplitude to four different APV25.