2020 TESTBEAM SETUP & FLAME DATA

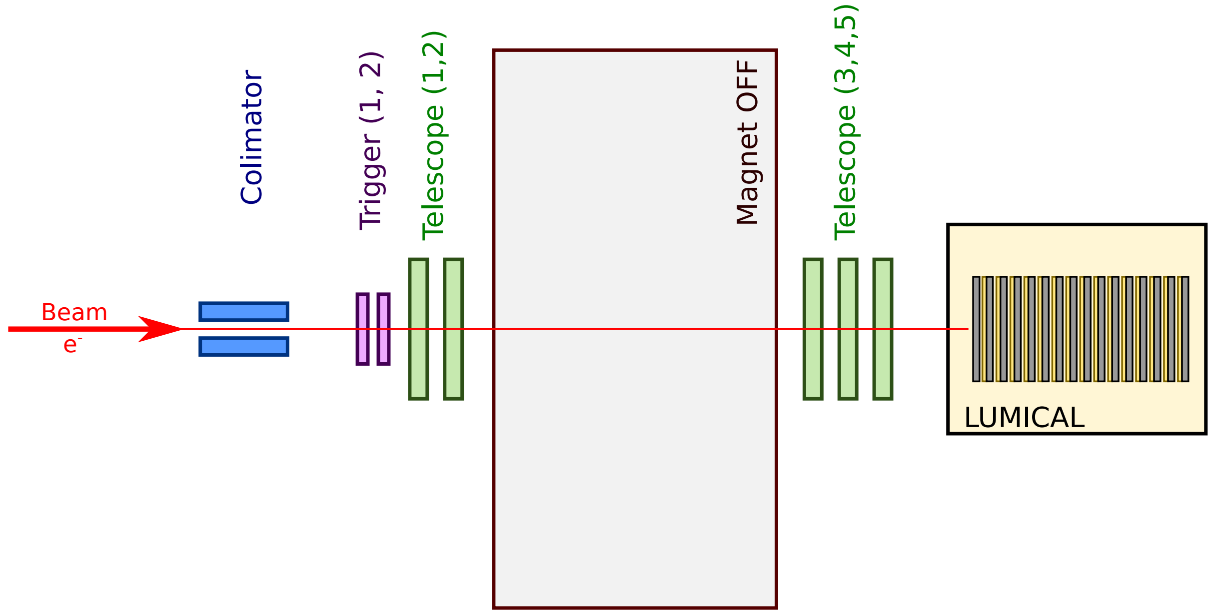
Szymon Bugiel

01.04.2020

OUTLINE

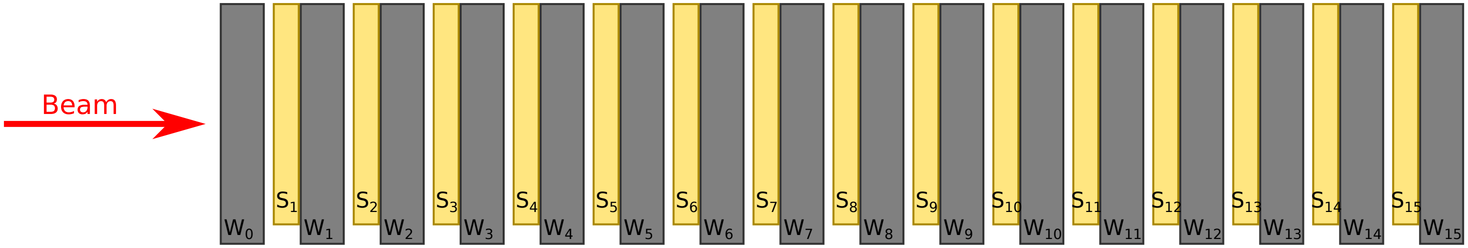
- **Testbeam setup:**
  - Regular setup descriprion
  - LumiCal configurations - runs overview
  - LUXE setup description
- **Flame data**
  - Data file description
  - Flame tree reader
- **FLAME ↔ APV ↔ TELESCOPE data correlation**

TESTBEAM SETUP – REGULAR CONFIGURATION



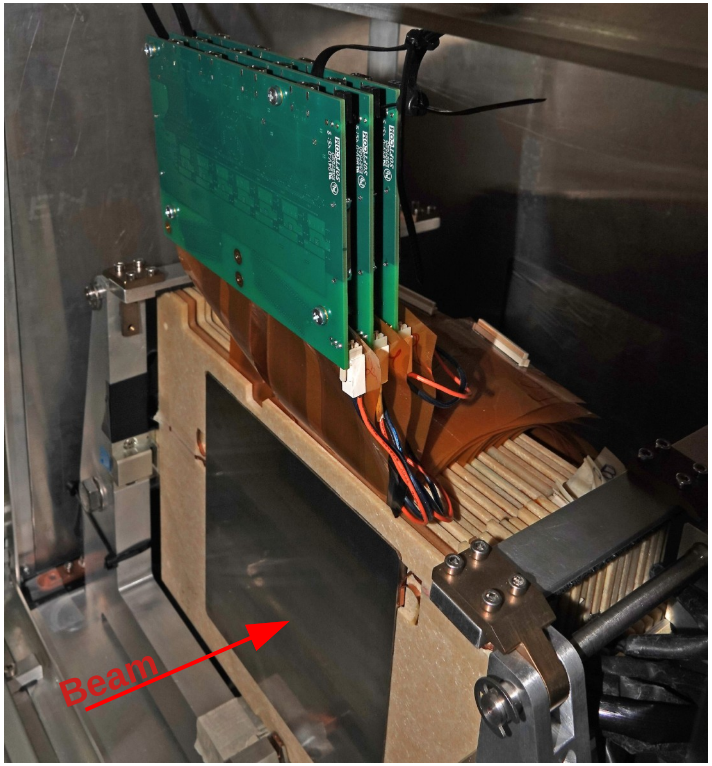
- Beam spot after the colimator ~5mm x 5mm
- Two scintilator triggers operating in coincidence mode
- 5 telescope planes – 2 before and 3 after the magnet
- Magnet switched OFF
- LumiCal placed on movable table

LUMICAL CONFIGURATION



Stack overview

- 15 sensor layers (S1 - S15) glued to tungsten absorbers (W1 – W15)
- Additional tungsten layer in front of the stack (W0)



RUNS OVERVIEW - RUN 661 – 881: FLAME STANDALONE

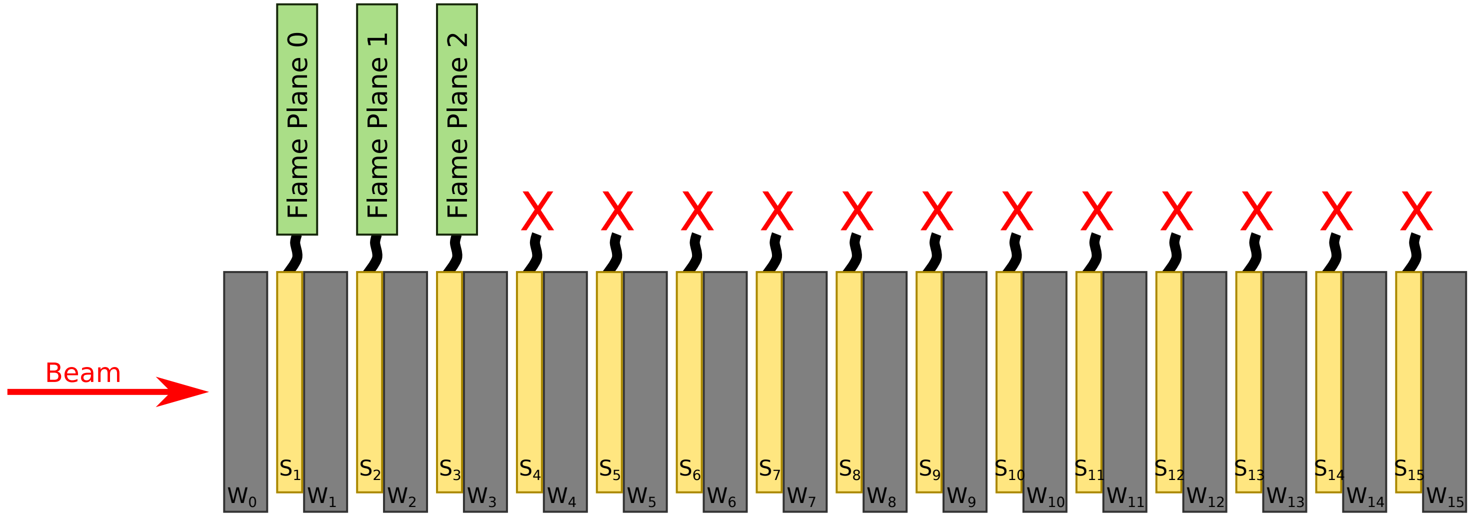
Main purpose

- Shower development accros the calorimeter.

- **Testbeam setup – regular**
- **LumiCal setup:**
  - Splitted into sub-configurations since only 3 Flame readout were available

---

- **LumiCal Configuration A (runs: 661 - 683)**
  - 661–667 → 3.6 GeV Beam
  - 668–683 → 5 GeV Beam



RUNS OVERVIEW - RUN 661 – 881: FLAME STANDALONE

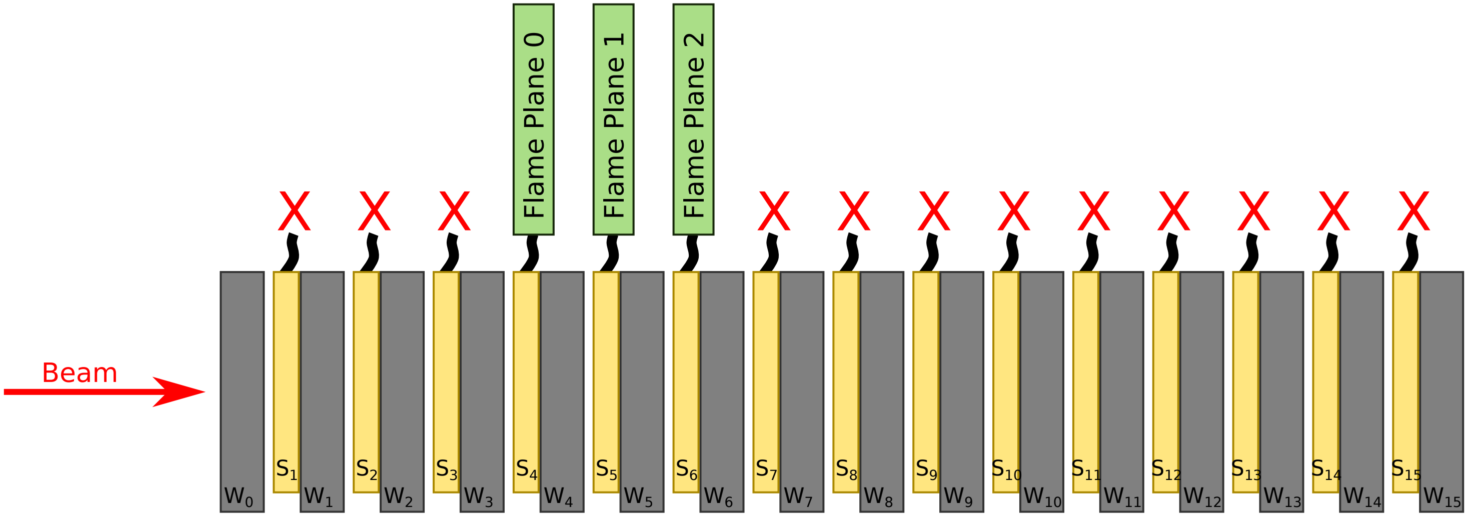
Main purpose

- Shower development accros the calorimeter.

- **Testbeam setup – regular**
- **LumiCal setup:**
  - Splitted into sub-configurations since only 3 Flame readout were available

---

- **LumiCal Configuration B (runs: 697 - 745)**
  - 697–737 → Energy Scan (1, 2, 3, 4, 5 Gev)
  - 738–745 → Flame *debug* data – containing additional informations in output tree



RUNS OVERVIEW - RUN 661 – 881: FLAME STANDALONE

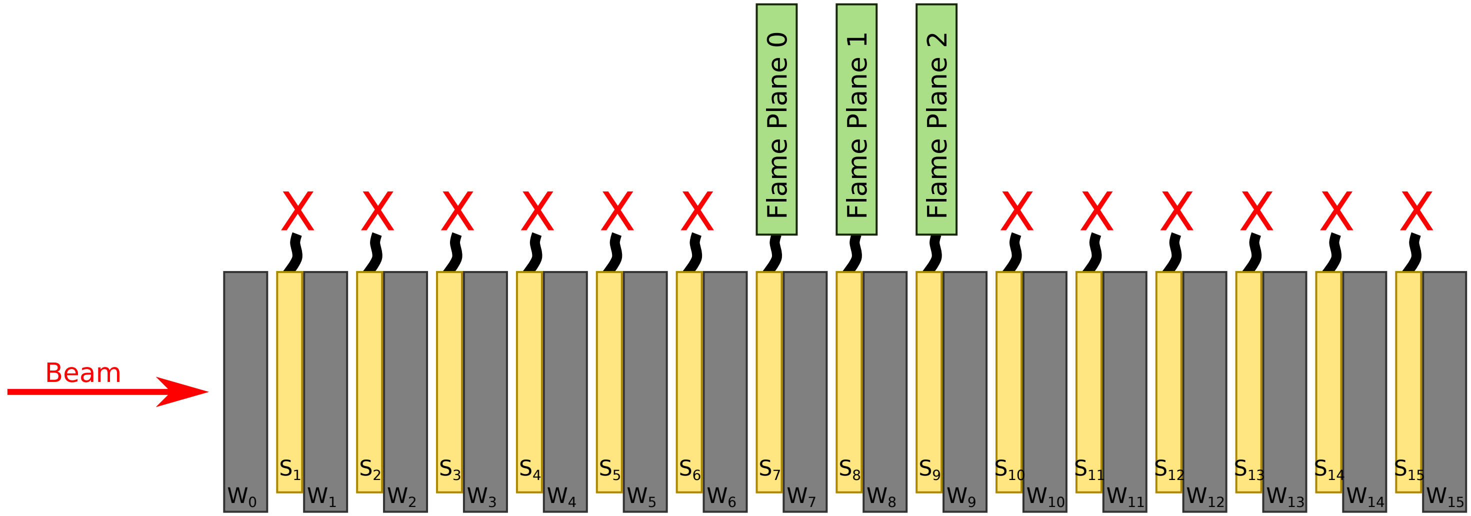
Main purpose

- Shower development accros the calorimeter.

- **Testbeam setup – regular**
- **LumiCal setup:**
  - Splitted into sub-configurations since only 3 Flame readout were available

---

- **LumiCal Configuration C (runs: 746 - 755)**
  - 746–755 → 5 GeV only
    → **No signal from sensor 8, only noise**



RUNS OVERVIEW - RUN 661 – 881: FLAME STANDALONE

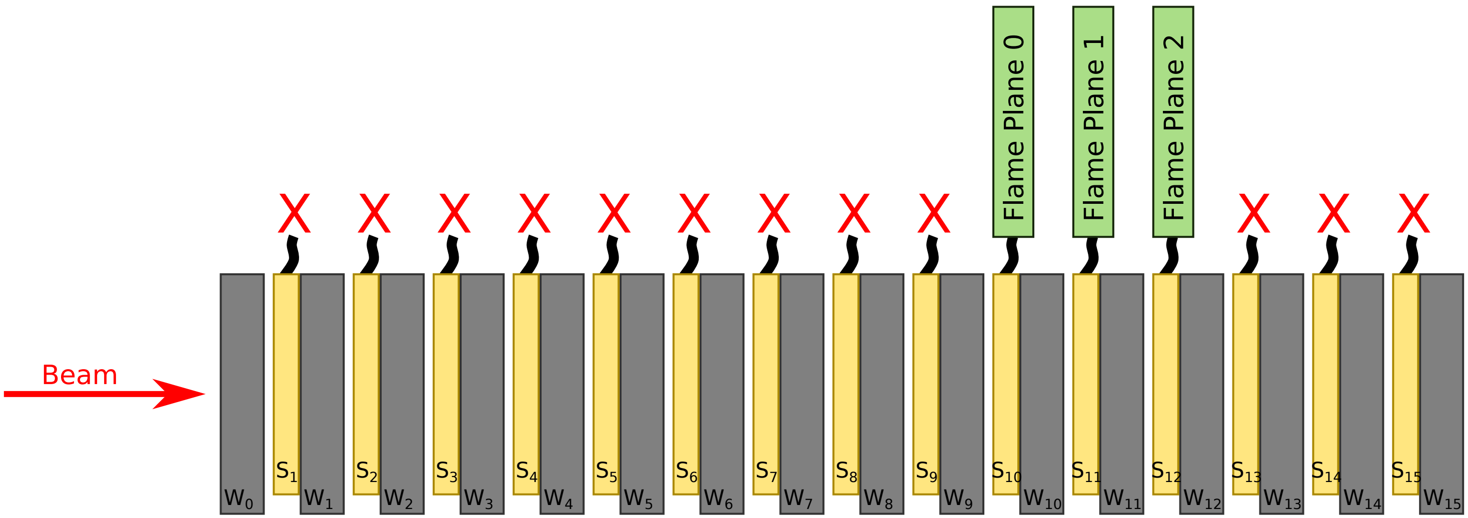
Main purpose

- Shower development accros the calorimeter.

- **Testbeam setup – regular**
- **LumiCal setup:**
  - Splitted into sub-configurations since only 3 Flame readout were available

---

- **LumiCal Configuration D (runs: 757 - 764)**
  - 757–764 → 5 GeV only



RUNS OVERVIEW - RUN 661 – 881: FLAME STANDALONE

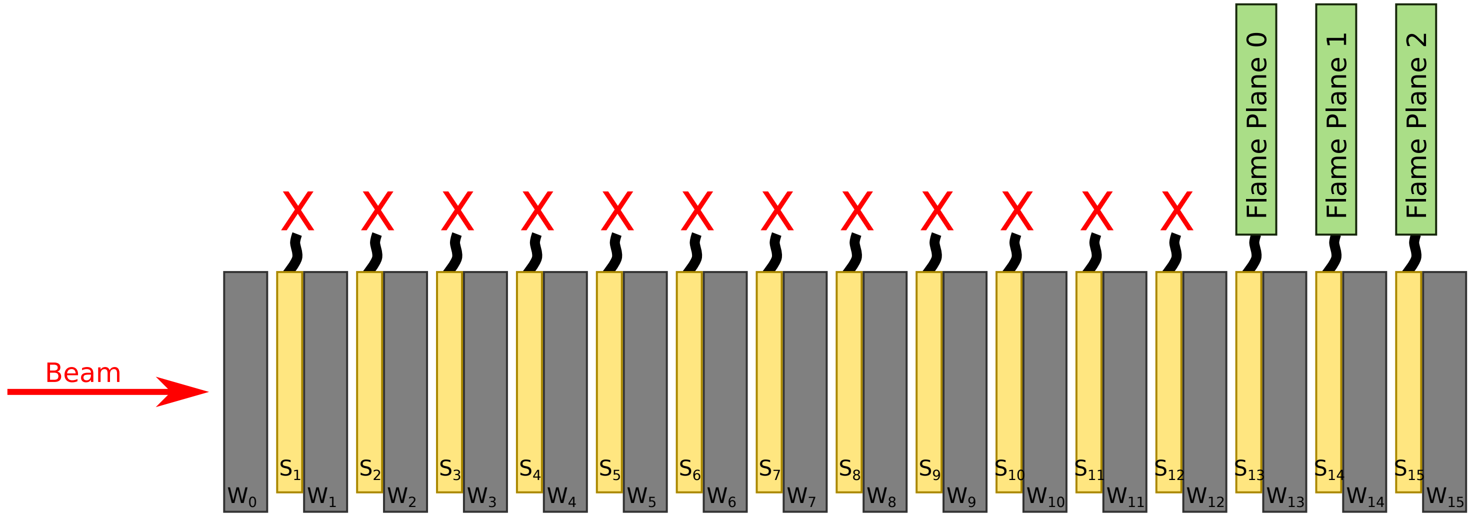
Main purpose

- Shower development accros the calorimeter.

- **Testbeam setup – regular**
- **LumiCal setup:**
  - Splitted into sub-configurations since only 3 Flame readout were available

---

- **LumiCal Configuration E (runs: 869 - 875)**
  - 869–875 → 5 GeV only



RUNS OVERVIEW - RUN 661 – 881: FLAME STANDALONE

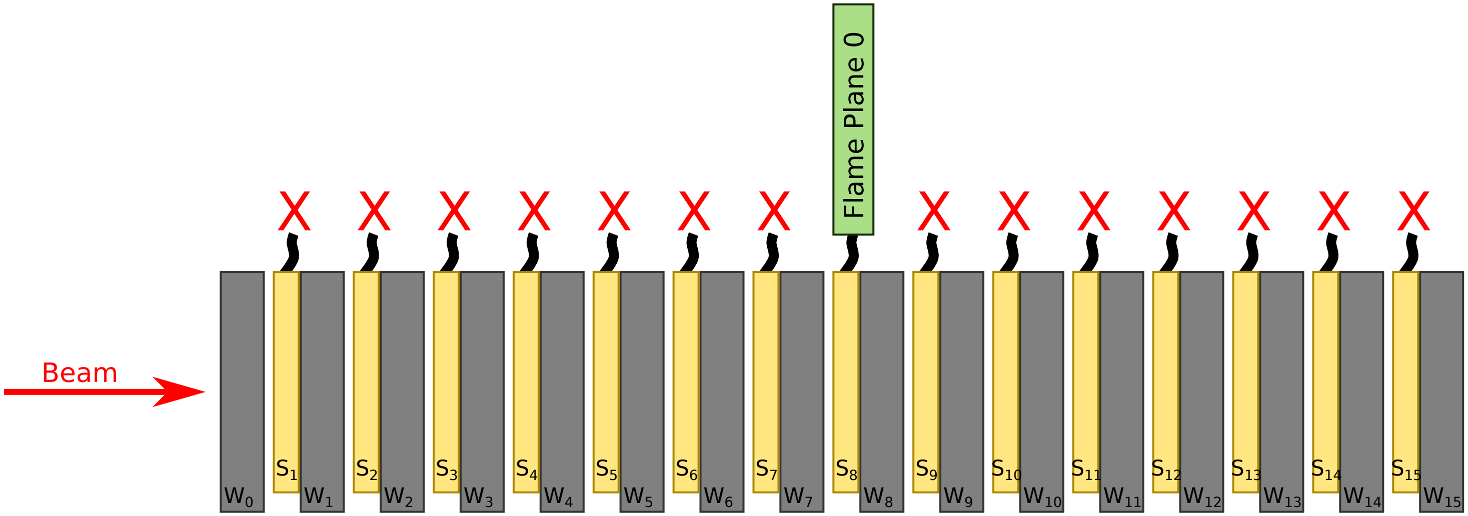
Main purpose

- Shower development accros the calorimeter.

- **Testbeam setup – regular**
- **LumiCal setup:**
  - Splitted into sub-configurations since only 3 Flame readout were available

---

- **LumiCal Configuration F (runs: 877 - 881)**
  - 877–881 → 5 GeV only
    → single flame bord connected to sensor 8 wchich was not responding in configuration C



RUNS OVERVIEW - RUN 661 – 881: FLAME STANDALONE

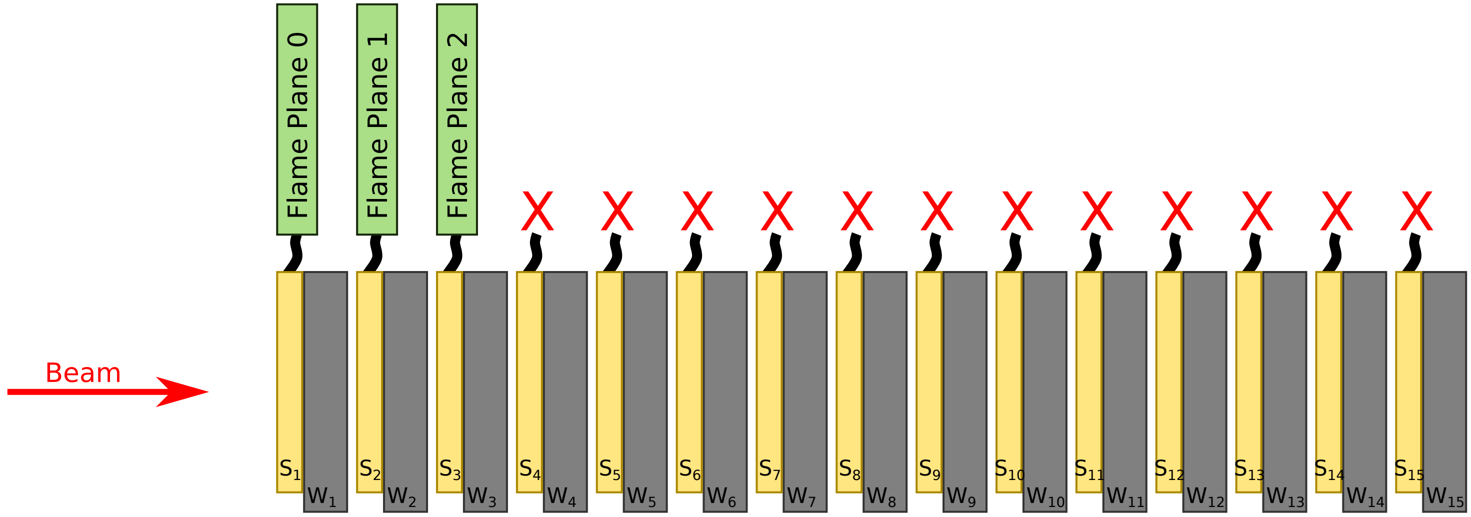
Main purpose

- Shower development accros the calorimeter.

- **Testbeam setup – regular**
- **LumiCal setup:**
  - Splitted into sub-configurations since only 3 Flame readout were available

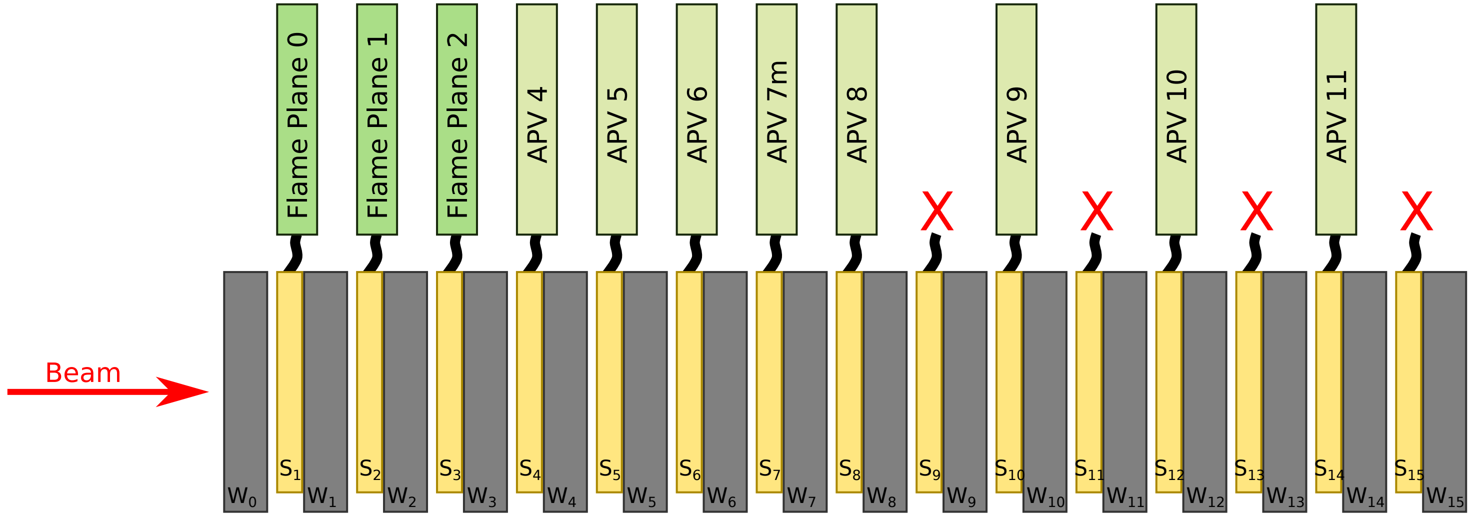
---

- **LumiCal Configuration A-- (runs: 765 – 868)**
  - Wo tungsten layer removed to directly see MIPs on first sensor
  - 765-826 → 5 GeV automatic overnight runs → Huge statistics (~60M events)
  - 827-830 → Flame *debug* data – containing additional informations in output tree
  - 833-868 → XY scan



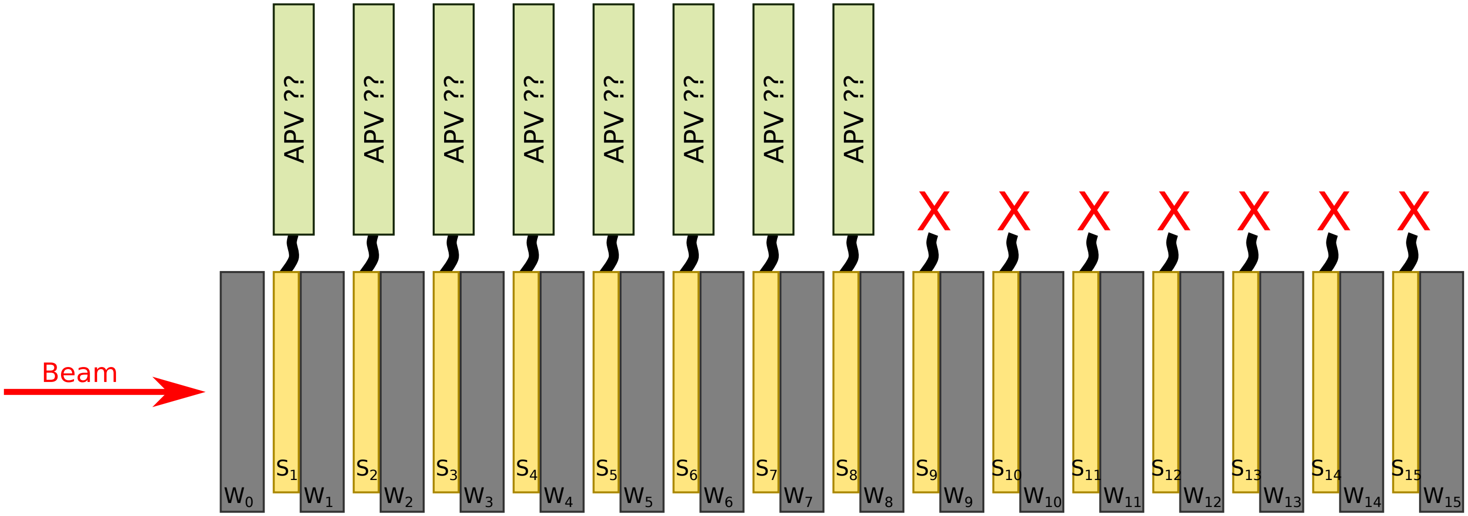
RUNS OVERVIEW - RUN 912 – 944: FULL SETUP

- **Testbeam setup – regular**
- **LumiCal setup:**
  - First 3 layers equiped with Flame, rest wirth APVs – according to plot
  - High noise spotted in both systems – mainly at the interface (layers S3/S4)
  - Energy scan (1, 2, 3, 4, 5 GeV)
  - Beam rate lowered ~10 times for SRS



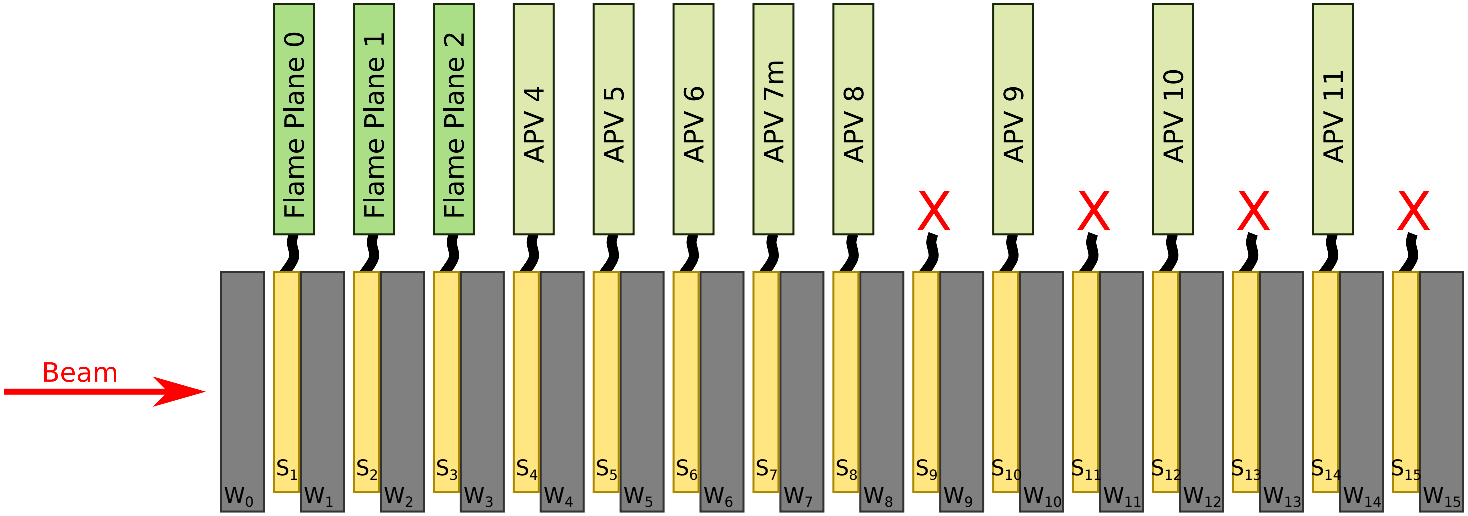
RUNS OVERVIEW - RUN 948 – 944: APV ONLY

- **Testbeam setup – regular**
- **LumiCal setup:**
  - Flame unmounted since two systems do not wanted to work toogether (no grounding scheme satisfying both systems found at that point)
  - First 8 layers equiped with APVs
  - **LumiCal tilted by 2, 4, 6 degrees**
  - 948–966 → 2 deg – XY scan
  - 968–974 → 4 deg – two XY positions
  - 981–994 → 6 deg – two XY positions

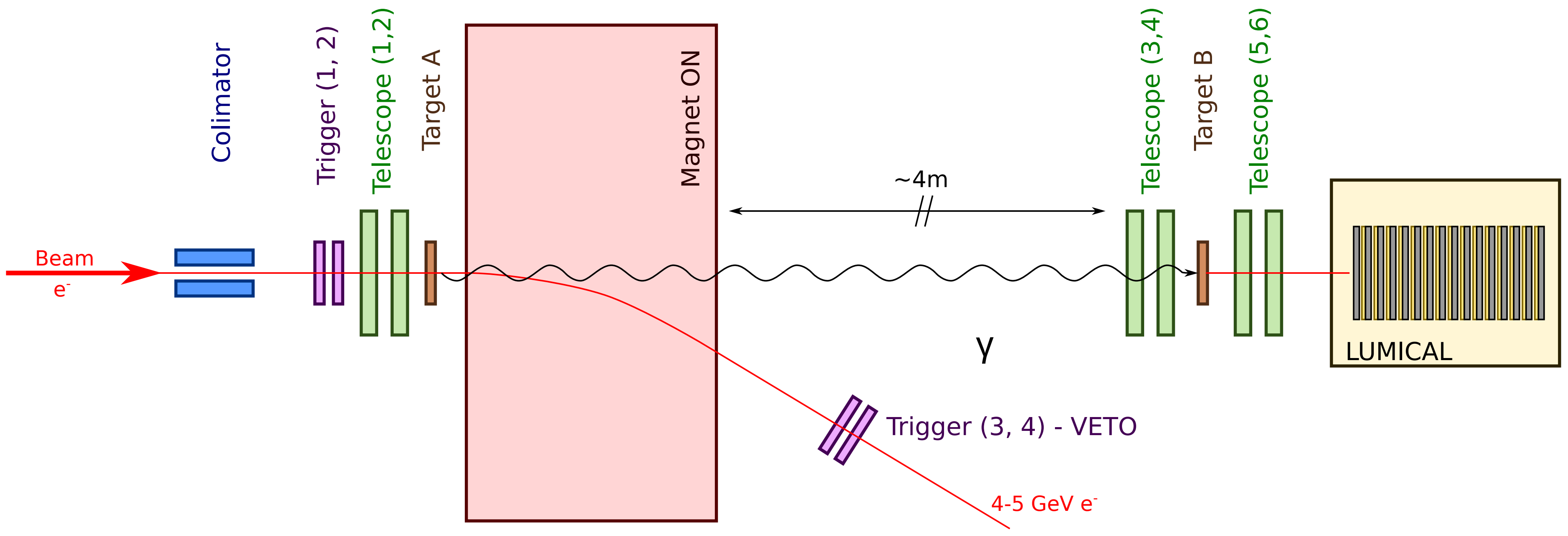


RUNS OVERVIEW - RUN 1002 – 1069: FULL SETUP

- **Testbeam setup – regular**
- **LumiCal setup:**
  - *Optimal* grounding schame founded – get back to the full setup Flame + APV
  - **XY scan – approaching calorimeter edge**
  - **Eergy scan (1, 2, 3, 4, 5, 6 GeV)**



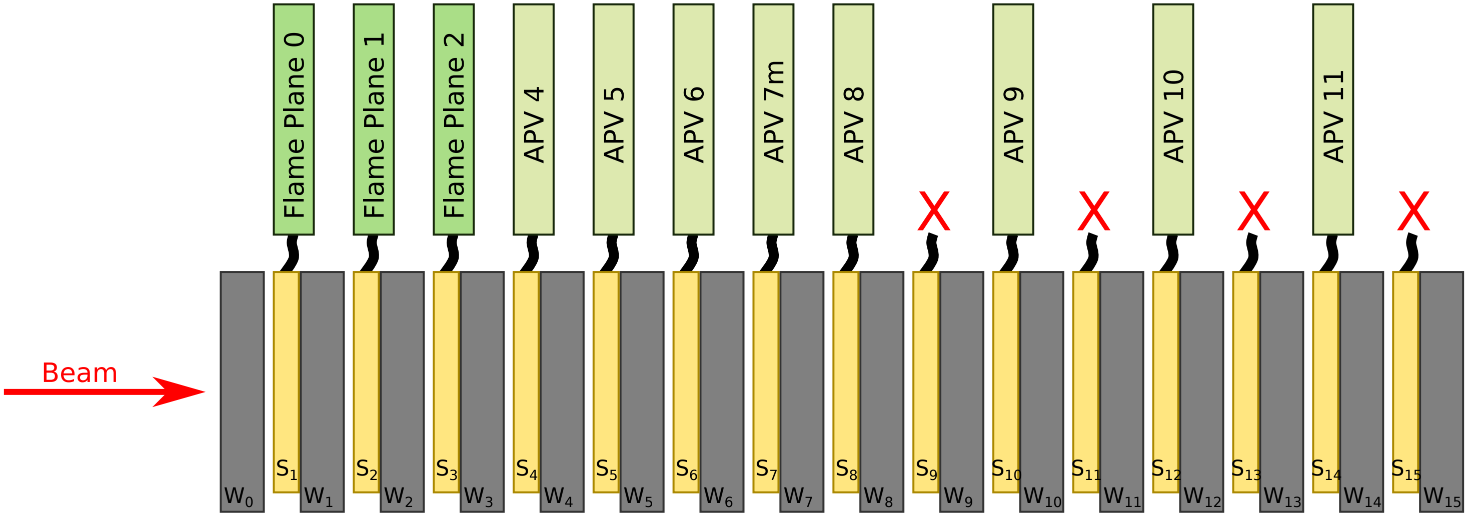
TESTBEAM SETUP – LUXE CONFIGURATION



- Beam spot after the colimator ~5mm x 5mm
- Trigger logic:
  - Innitially only scintilator 1 and 2 working in coincidence (T = T1 & T2)
  - Finally scintilators 3 and 4 included in anty-coincidence mode (T = T1 & T2 & (~T3) & (~T4))
- Second telescope arm equipped with additional 6th layer
- Second telescope arm and LumiCal moved to the second movable table located ~4m from the magnet
- Magnet swithecd ON (200-300A)
- Different targer A/B configuration → see the eLogbook

RUNS OVERVIEW - RUN 1125 – 1543: LUXIE SETUP

- **Testbeam setup – LUXIE**
- **LumiCal setup:**
  - Full setup: Flame + APV
  - 1125–1286 → basic trigger setup (T1 & T2)
  - 1507–1543 → High energy e⁻ rejection trigger (T1 & T2 & (~T3) &(~T4))
  - **For target configuration check the eLog (https://docs.google.com/spreadsheets/d/1qS55WGUZW4g3UOgdNJo9VvmmejKlH7q3CJ_wasWfozQ/edit#gid=0)**



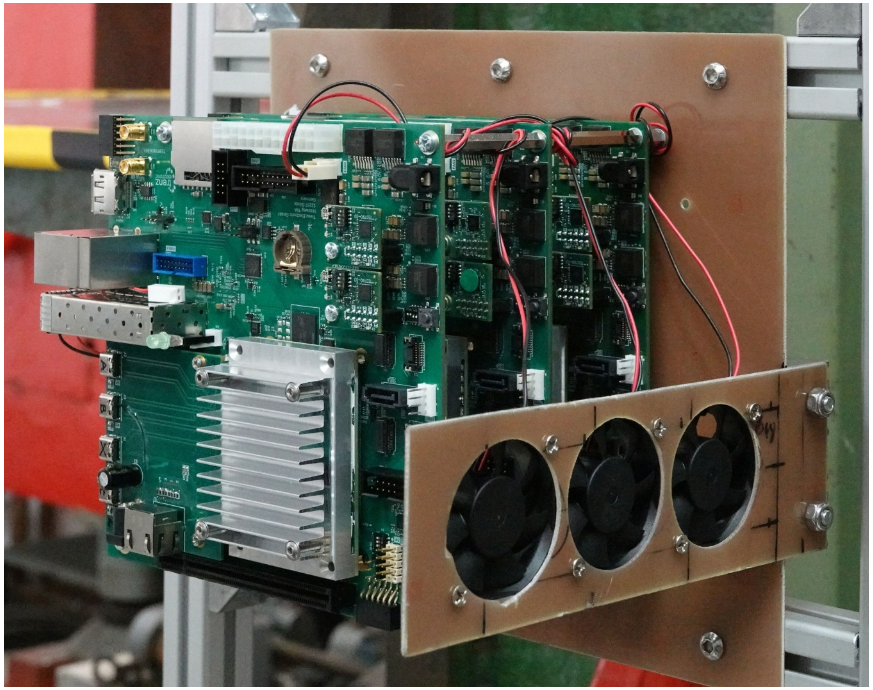
FLAME DATA

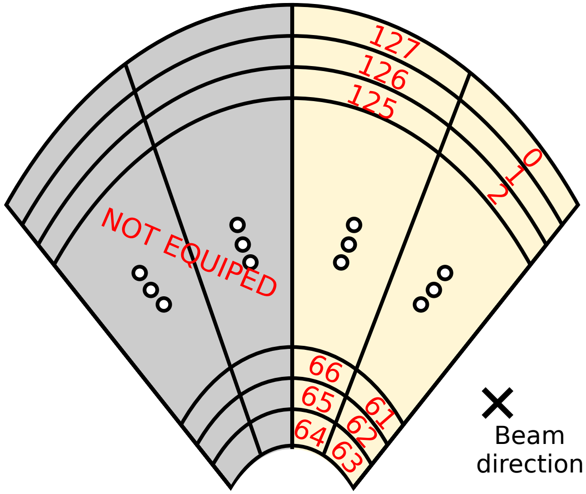
FLAME READOUT – DATA PROCESSING

- There are no 'raw' data samples stored for Flame (except debug runs)

- Signal processing/extraction is made already on the FPGA level, which includes:
  - Pedestal Subtraction
  - Common Mode Subtraction
  - Signal Extraction
  - Zero Suppresion

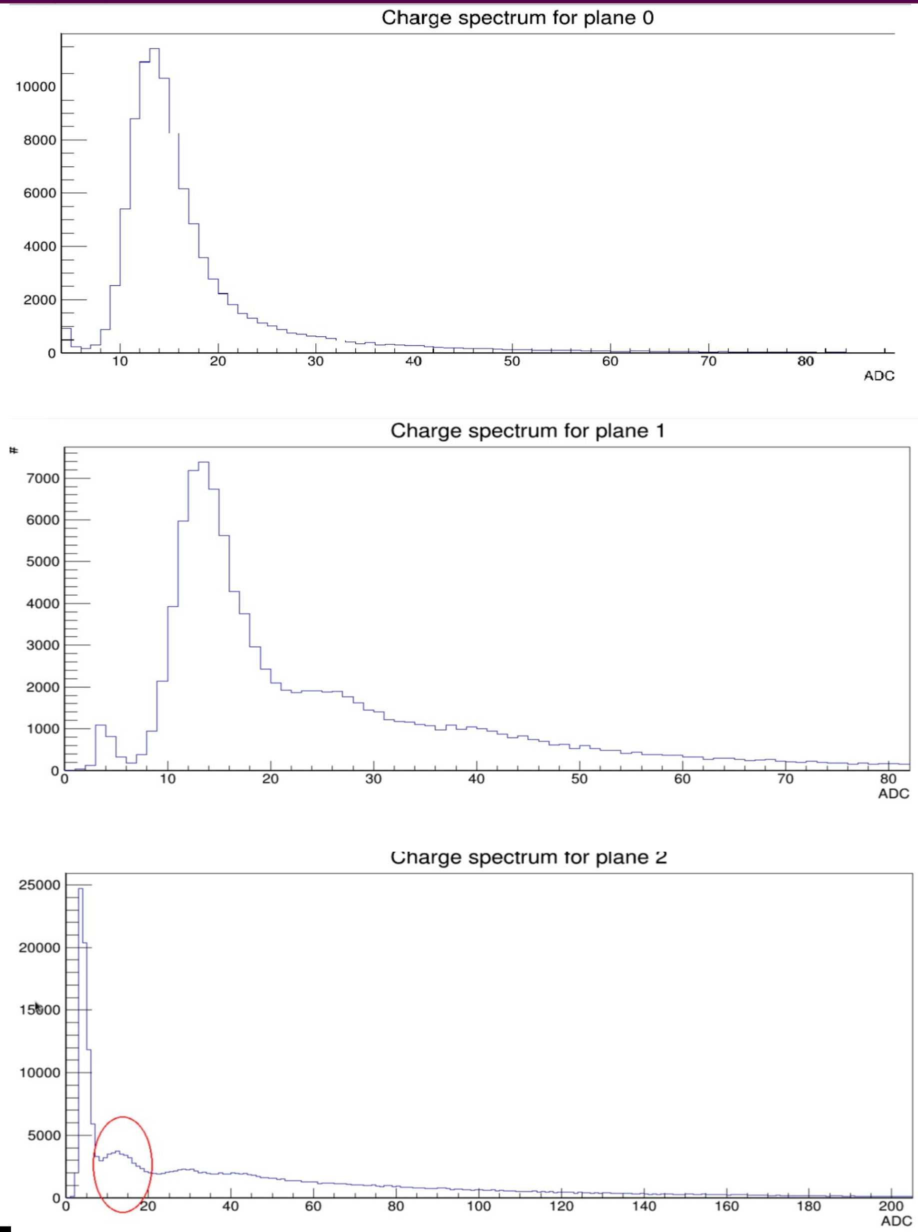
- Output informations corresponding to each single hit are:
  - Chanel number (corresponding to the picture)
  - Signal amplitude (in ADC units)
  - Time of Arrival
  - Chanel gain (constant during whole testbeam)





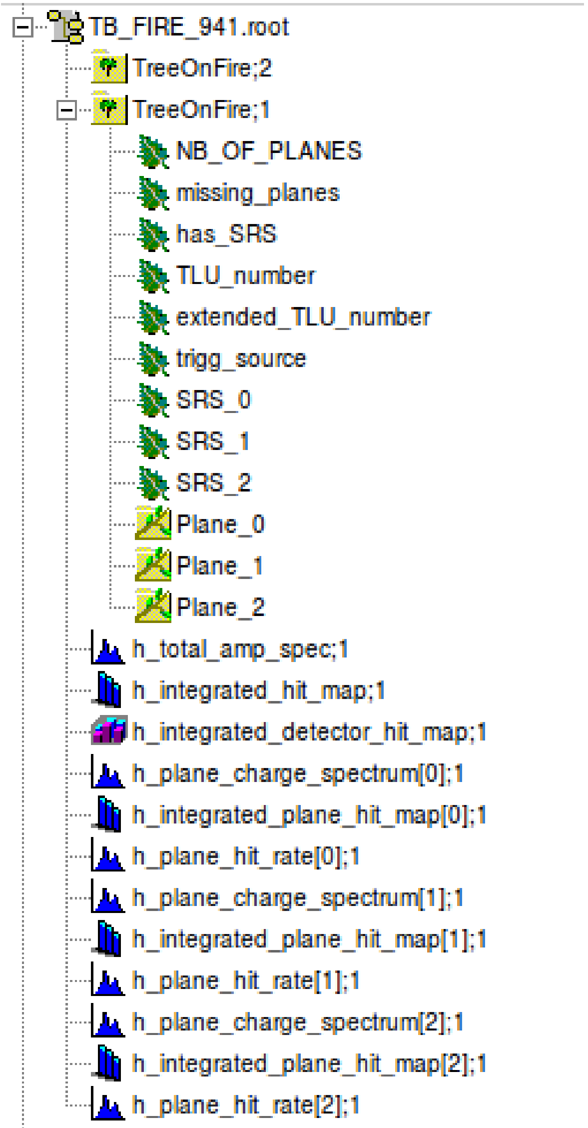
FLAME READOUT – DATA PROCESSING

- There are no 'raw' data samples stored for Flame (except debug runs)
- Signal processing/extraction is made already on the FPGA level, which includes:
  - Pedestal Subtraction
  - Common Mode Subtraction
  - Signal Extraction
  - Zero Suppresion
- Output informations corresponding to each single hit are:
  - Chanel number (corresponding to the picture)
  - Signal amplitude (in ADC units)
  - Time of Arrival
  - Chanel gain (constant during whole testbeam)

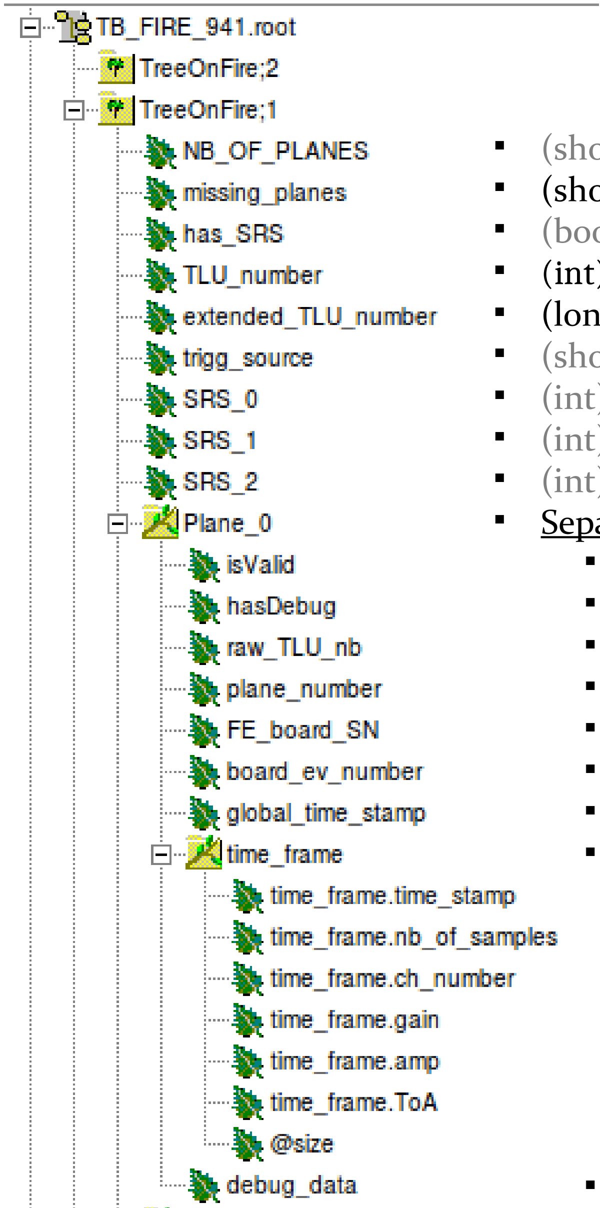


FLAME DATA STRUCTURE

- For each run a pair of **TB_FIRE_#run_nb#.root** and **.log** files is created
- **#run_nb#** is directly taken from the telescope run number
- **.log** file contains the flame settings
- **Flame data** are storred in a **.root** files
- One can easyly browse the file through TBrowser
- **Each .root file contains:**
  - Tree with „raw" data „ TreeOnFire"
  - Several basic plots like:
    - Signal spectrum for each plane (h_plane_charge_spectrum[x])
    - Hit map for each plane (h_plane_hit_map[x])



FLAME TREE STRUCTURE



- (short) Nb of Flame planes → always 3 ...
- (short) Nb of planes from which data packages were missing in given event (should be 0)
- (bool) Not used
- (int) 15-bit 'raw' TLU nb – provided by TLU
- (long) TLU nb extended to long (64b) by the DAQ software
- (short) Flame trigger source
- (int) Not used
- (int) Not used
- (int) Not used
- Sepatare branch for each plane
  - (bool) Value telling if data package from this plane was available
  - (bool) Value telling if there are some additional debug information
  - (short) Basically repetition of TLU_number
  - (short) Plane number
  - (short) Not used
  - (int) Event number – counted by each FPGA board individualy
  - (long) Value of timestamp counter – counted by each FPGA board individualy
  - Vector of time frames
    - (long) Value of the timestamp counter for given time_frame
    - (short) Number of hits in certain time_frame (==time_frame.amp.size())
    - (vec<short>) Vector containing chanel numbers of each hit in given time_frame
    - (vec<short>) Vector containing chanel gain of each hit in given time_frame
    - (vec<short>) Vector containing signal amplitude of each hit in given time_frame
    - (vec<short>) Vector containing time of arrival of each hit in given time_frame
    -
  - Debug data → (vec<int>) additional data stored only for several runs

READING FLAME TREE

- To properly read the Flame Tree one should provide the definitions of structures used for root file composition:
  - → Plane
  - → TimeFrame
- Listning of the plane_str.h file attached to the presentation →
- To write your own file reader one need to create shered libraries specifing these structures
- This can be done adding **this** to your main function:

```
#if !defined(__CINT__)
if (!(gInterpreter->IsLoaded("plane_str")))
gInterpreter->ProcessLine("#include \"plane_str.h\"");
gInterpreter->GenerateDictionary("Plane", "plane_str.h");
gInterpreter->GenerateDictionary("TimeFrame", "plane_str.h");
#endif /* !defined(__CINT__) */

#ifdef __MAKECINT__
#pragma link C++ class Plane+;
#pragma link C++ class TimeFrame+;
#endif
```

- The example of FireTreeReader is attached to the presentation
- To run it one need to specify the path to the root tree (in main)
- Compile it with:

```
`root-config --cxx` `root-config --cflags`  -O2 -W TreeOnFireReader.cpp -o
offline_analysis `root-config --ldflags` `root-config --glibs`
```

- And run: ./offline_analysis
- It shoud produce some basic plots (signal spectum / hit maps)

```
#ifndef PLANE_STR_h
#define PLANE_STR_h

#include <vector>

// Definition of structures defining the outpou tree shape
struct TimeFrame
{
  long time_stamp;
  short nb_of_samples;
  std::vector<short> ch_number;
  std::vector<short> gain;
  std::vector<short> amp;
  std::vector<short> ToA;
   TimeFrame()  // initialization list
             : time_stamp(-1),
               nb_of_samples(0){}
};

struct Plane
{
  bool isValid;
  bool hasDebug;
  short raw_TLU_nb;
  short plane_number;
  short FE_board_SN;
  int board_ev_number;
  long global_time_stamp;
  std::vector<TimeFrame> time_frame;
  std::vector<int> debug_data;
  Plane()  // initialization list
             : isValid(0),
               hasDebug(0),
               raw_TLU_nb(-1),
               plane_number(-1),
               FE_board_SN(-1),
               board_ev_number(-1),
               global_time_stamp(-1){ }

};

#endif
```

FLAME ↔ APV ↔ ALPIDE DATA CORELATION

- **FLAME ↔ ALPIDE**
  - Both FLAME and ALPIDE(telescope) events contains the same unique TLU number so the corelation should be straightforward, but as far as I know not yet werified ...

- **FLAME/ALPIDE ↔ APV**
  - APV does not store the TLU number, so one need to try to corelate events besing on the event number (assuming that all systems are working perfectly – not loosing any events)
  - **Thanks to Bohdan, we have already succeeded to corelate several FLAME ans APV runs! Discovering several issues...**

SPOTTED ISSUES:

- **Flame:**
  - Missing events on the begining of run (up to ~30 events)
    (first stored Flame event does not have TLU number = 0)
  - Missing events on the end of the run (arount 1000 events)
    (this issue was fixed during the testbeams so later runs shoul have last TLU number the same as last ALPIDEs TLU number)
- **SRS:**
  - Contains some duplicated events, that makes a corealtion by event number impossible without proper procedure of removing them
- **ALPIDE:**
  - No valid telescope data if at least one telescope plane stoped responding

FLAME ↔ APV CORELATION PROCEDURE

- **FLAME ↔ APV Correlation procedure proposal:**
  1) Clean APV Data sample, by removing all duplicated events (but nothing more)
  2) Check the value of the first TLU_numbrer stored by Flame (= N)
  3) Skipp N first APV events
  4) From this point events form both systems should be corelated
  5) One should also always check if the Flame TLU_number is incremented by 1 in each ceonsecutive event (as it should) and if not, then one should skip some APV events also.

- **Such a procedure seemed to work for several runs checked during the test beam**

- **In case of any troubles, Bohdan has already implemented a more fancy procedure that do not need any assuptions – simply returns the shift between the FLAME and APV entries**

THANK YOU!