



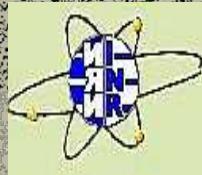
# TO Calibration Status Report

ALICE offline week

Tomasz Mankiewicz & Alla Maeuskaya



HIP - Department of Physics  
University of Jyväskylä  
Finland



Institute for Nuclear Research  
Moscow

**TO in ALICE**

The diagram illustrates the TO (Time-Of-Flight) detector setup in the ALICE experiment. It shows two main components: T0-A (left) and T0-C (right). T0-A consists of two vertical layers of scintillators with photomultiplier tubes (PMTs) at the top and bottom. T0-C is a cylindrical detector with a central beam pipe and several concentric layers of scintillators. Blue arrows point from the labels 'T0-A' and 'T0-C' to their respective components in the diagram. To the right of the diagram is a photograph of the physical hardware, showing the cylindrical T0-C detector and its support structure.

**Online :**

- Trigger signals:
  - T0-mean
  - T0-vertex
  - T0-C
  - T0-A
  - T0-centrality
- TRD 'wake-up' trigger

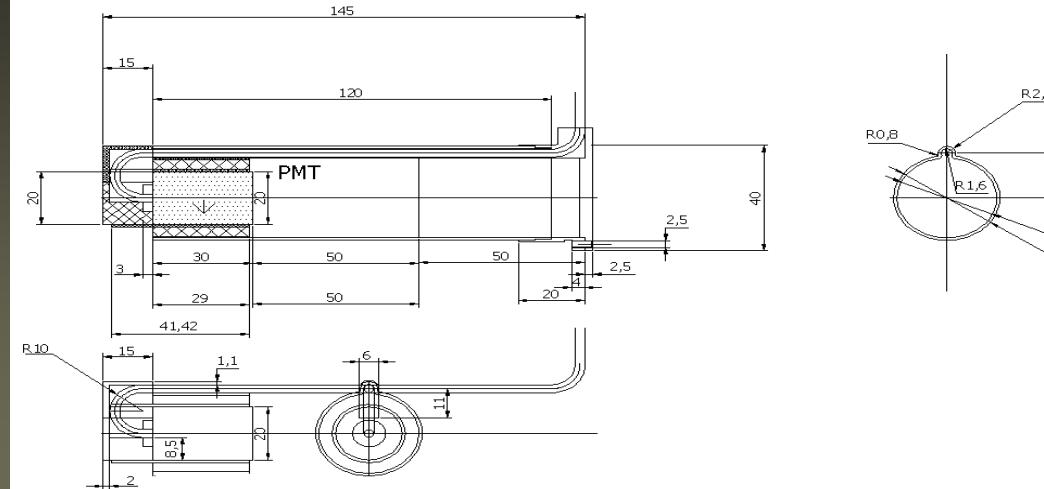
**Offline:**

- Vertex
- Interaction time

Tomasz Malkiewicz & Alla Maevskaia      ALICE offline week      October 2005



# TO Calibration with LCS



## Laser calibration system

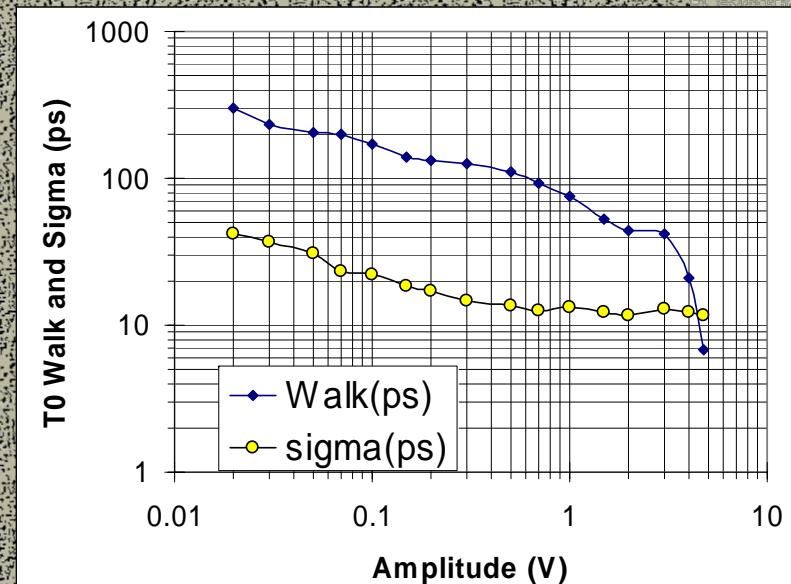
- laser pulses send to PMT
- data acquired as during normal event
- Ongoing tests of PMT
- During long gap between BC – one or more laser pulses send to all PMTs



# Calibration objects for DB used in reconstruction

For each channel

- time delay
- time vs amplitude calibration curve for accurate time definition



Time-amplitude dependence.

*Full list of objects is not clear at present*



# Using calibration data in reconstruction procedure

- For each channel correct time signal using time-amplitude curve;
- take into account time delay in cables and FE electronics;
- choose channels with smallest time on each side;
- calculate T0 and Zvertex using time information from chosen channels;
- write T0 and Zvertex in AliESD



# AliSTARTCalibData.h

```
class AliSTARTCalibData: public TNamed {  
public:  
    AliSTARTCalibData();  
    AliSTARTCalibData(const char* name);  
    AliSTARTCalibData(const AliSTARTCalibData &calibda);  
    AliSTARTCalibData& operator= (const AliSTARTCalibData &calibda);  
    virtual ~AliSTARTCalibData();  
    void Reset();  
    virtual void Print() const;  
    Float_t GetTimeDelay(Int_t channel) const {return fTimeDelay[channel];}  
    Float_t* GetTimeDelay() const {return (float*)fTimeDelay;}  
    void SetTimeDelay(Float_t val, Int_t channel) {fTimeDelay[channel]=val;}  
    void SetTimeDelay(Float_t* TimeDelay);  
    TH1F* GetHistTimeDelay() const {return fHistTimeDelay;}  
    void CleanHistos();  
    // amplitude - time corrections  
    TH1F* GetCurve (Int_t channel) {return fCorrCurve[channel];}  
    void SetCurve (Int_t channel, TF1 *corrCurve) {fCorrCurve[channel]=corrCurve; }  
protected:  
    Float_t fTimeDelay[24];      // Time delay for each channel  
    TH1F* fHistTimeDelay;       // time delay histogram  
    TH1F* fCorrCurve[24];       // correction curve time-amplitude  
    ClassDef(AliSTARTCalibData,1) // START Sensor Calibration data  
};  
#endif
```



# CBD objects used for control only

- *Optical properties of Cherenkov radiator*
- *PMT efficiency*
- *Equalising of amplitudes of each PMT*
- *Gain vs. HV curves*

## For centrality trigger

- *Measurements of characteristics of all 24 CFDs in full dynamic range*
- *Measurements of the output signals of linear adders*
- *Adjustment of multiplicity thresholds*



# Summary

- First implementation of calibration classes
- Current DB interface works smoothly

## Do be done:

- Include calibration procedure in reconstruction code
- Real data will be included after measurements
- All decalibration – recalibration business will be done in November