```
TransferLineOpen:  # Single line transfer
  clocks:          P1, P2, P3, P4
  slices:
     BufferP     = 0,  1,  1,  0
     OverlapP    = 0,  1,  1,  1
     TimeP       = 0,  0,  1,  1
     OverlapP    = 1,  0,  1,  1
     TimeP       = 1,  0,  0,  1
     OverlapP    = 1,  1,  0,  1
     TimeP       = 1,  1,  0,  0
     OverlapP    = 1,  1,  1,  0
     TimeP       = 0,  1,  1,  0
     5000 ns     = 0,  1,  1,  0 # made it
  constants:     S1=1, S2=1, SHU=1

ReverseLineOpen:  # Single line transfer i
  clocks:          P1, P2, P3, P4
  slices:
     BufferP     = 0,  1,  1,  0
     OverlapP    = 1,  1,  1,  0
     TimeP       = 1,  1,  0,  0
     OverlapP    = 1,  1,  0,  1
     TimeP       = 1,  0,  0,  1
     OverlapP    = 1,  0,  1,  1
     TimeP       = 0,  0,  1,  1
     OverlapP    = 0,  1,  1,  1
     TimeP       = 0,  1,  1,  0
  constants:     S1=1, S2=1, SHU=1
```

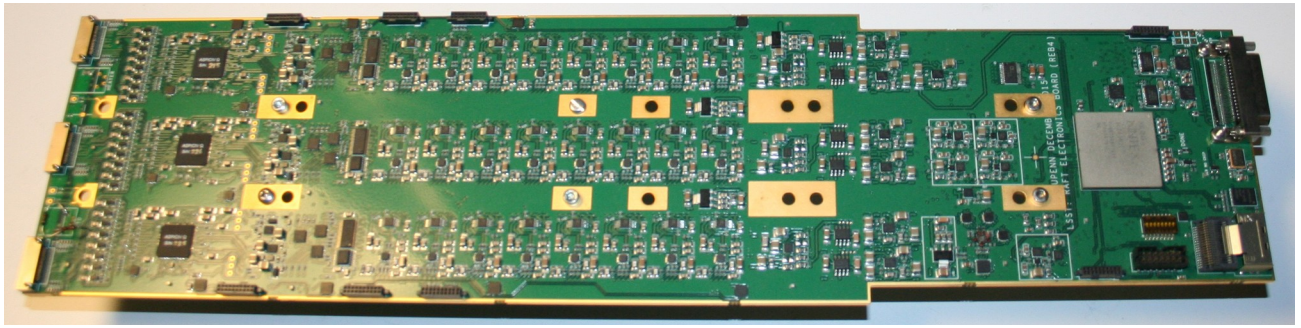
LSST REB Sequencer – Measuring linearity in one shot (a modest proposal)

Laurent Le Guillou
(Sorbonne Université / LPNHE)

LSST DESC Calibration Workshop
LPNHE, Paris, 2018-10-03

LPNHE : Pierre Antilogus, Pierre Astier, Claire Juramy-Gilles,
Laurent Le Guillou, Stefano Russo, Eduardo Sepulveda
APC : Eric Aubourg

LSST REB Sequencer

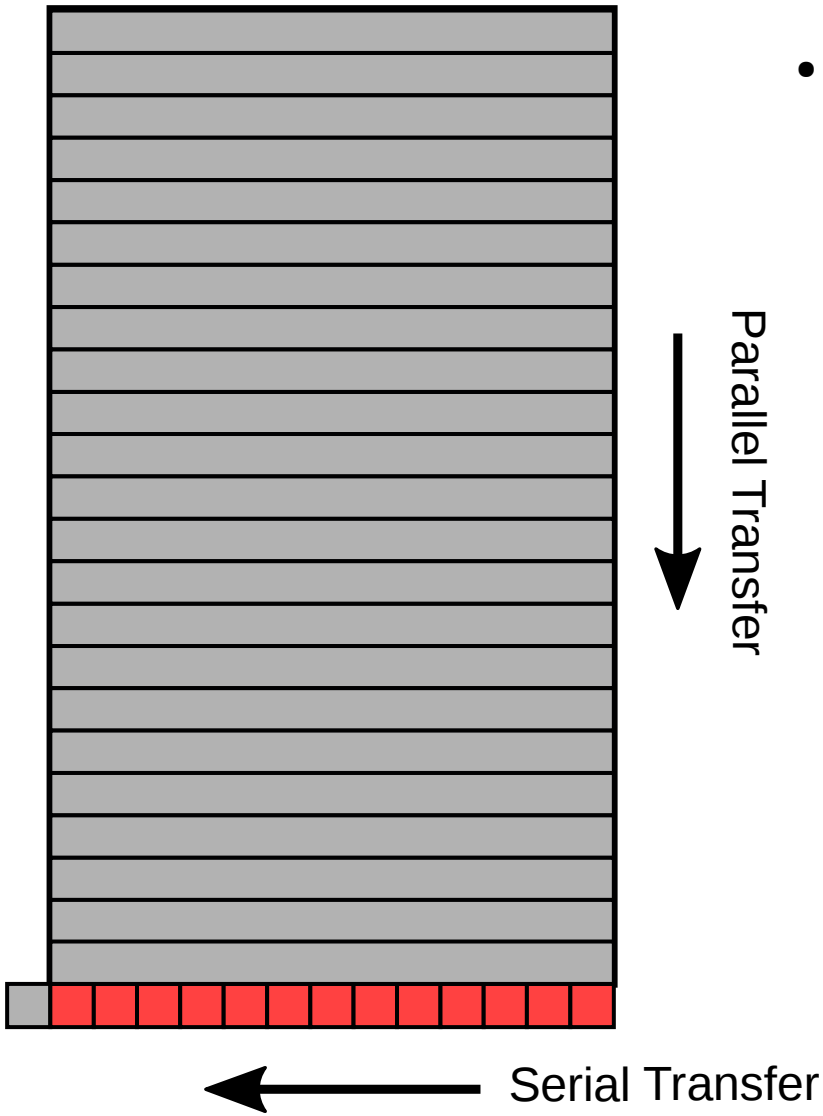


- **Very flexible sequencer programing** (REB FPGA, S. Russo *et al.*)
  - **Elementary chronograms** parts (sequencer « functions »)
  - **Subroutines** (« subroutines », could be nested to up to 16 levels)
  - **Direct / indirect addressing** for CALLs / JSRs of functions / subroutines
  - **High level seq. language** (assembly-like), python compiler → CCS

    https://github.com/lsst-camera-dh/sequencer-files/blob/master/sequencer-language.pdf
- **Huge amount of sequencer writing & optimisation by Claire Juramy et al.** (for E2V, ITL, rafts optimization, sensors & electronics problems, ...) **:**

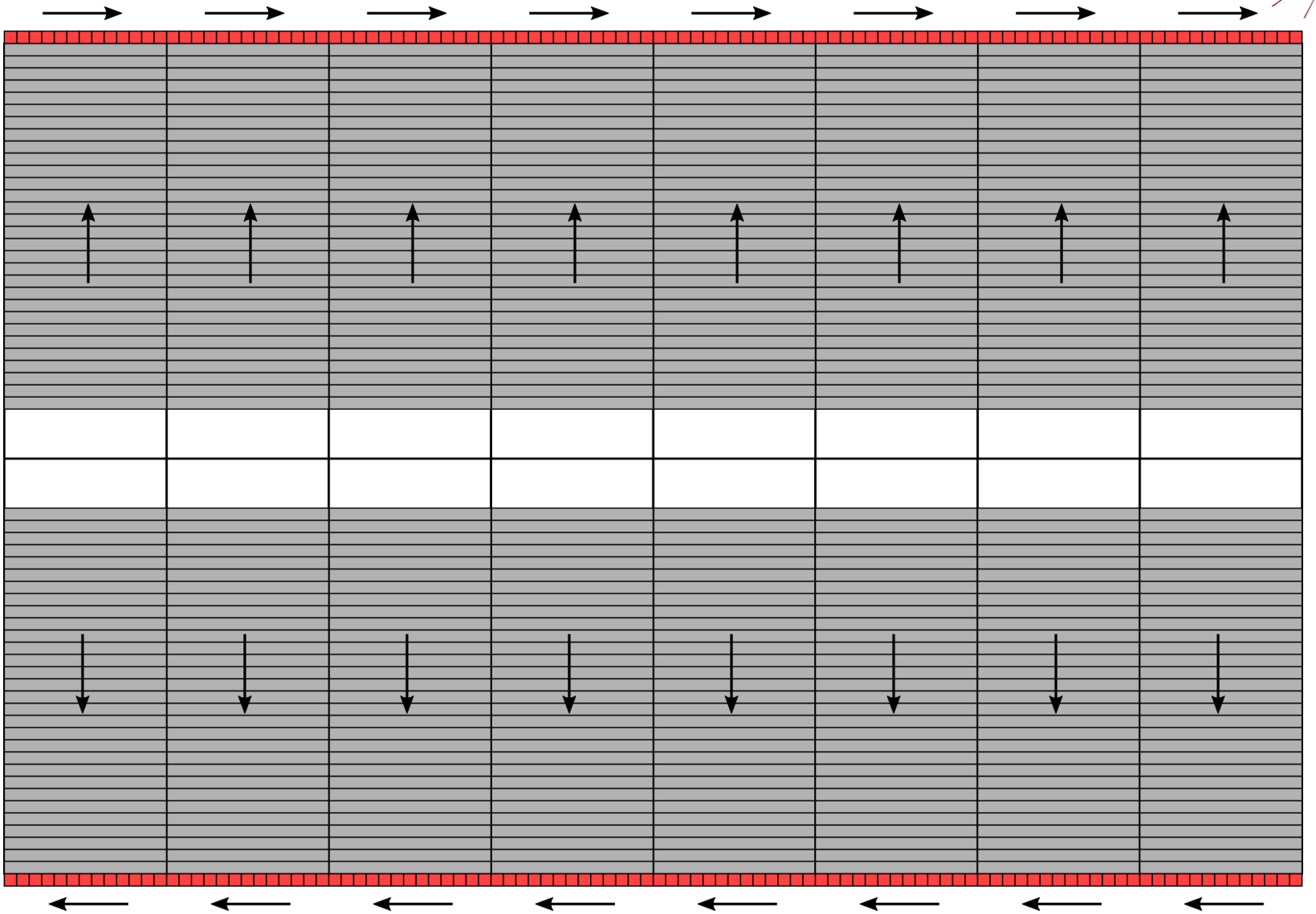
  https://github.com/lsst-camera-dh/sequencer-files

Usual way to read a CCD



- **« normal » frame readout :**
  - **Transfer each line down** (parallel transfer)
  - For each line in the serial register, **moving pixel charges,** pixel by pixel, towards the channel amplifier
  - **Sampling the charge**
  - Rince and repeat for all lines

LSST CCD have 16 channels



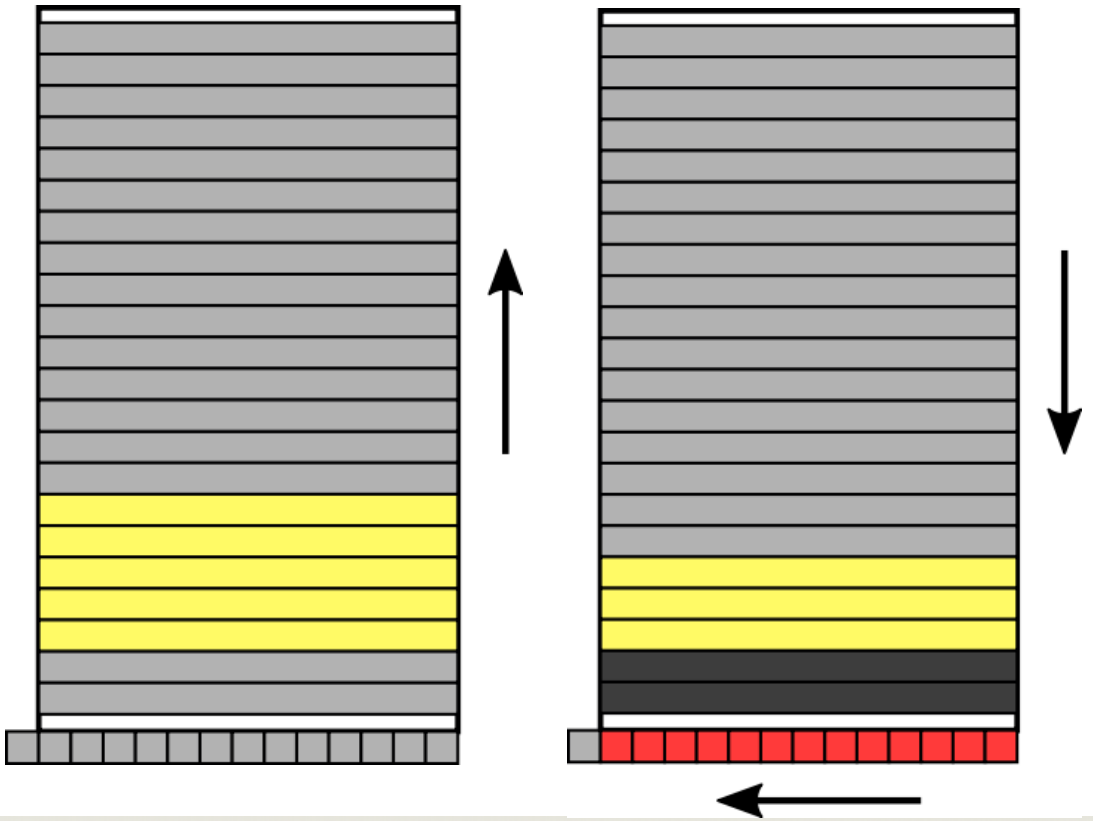
LSST REB Sequencer : extra stuff

- **Added extra possibilities** in the high level sequencer language
  - Preprocessing features (no changes in the FPGA interface) :

    Arithmetics, **local variables**, **conditional loops**, **IF tests**...
  - Already in our python compiler (2017)
  - currently added by Eric Aubourg to CCS
  - **Extra fun** when programming the REB sequencer ;-)
- **Make it easier** to program **non « regular » sequencer programs**
- **Motivations :**
  - Special sequences for the **LSST camera commissioning**
  - Calibration sequences for **regular monitoring of the camera**
  - Interesting alsoto test these sequences **AuxTel instrument**

Tentative for a “linearity frame”

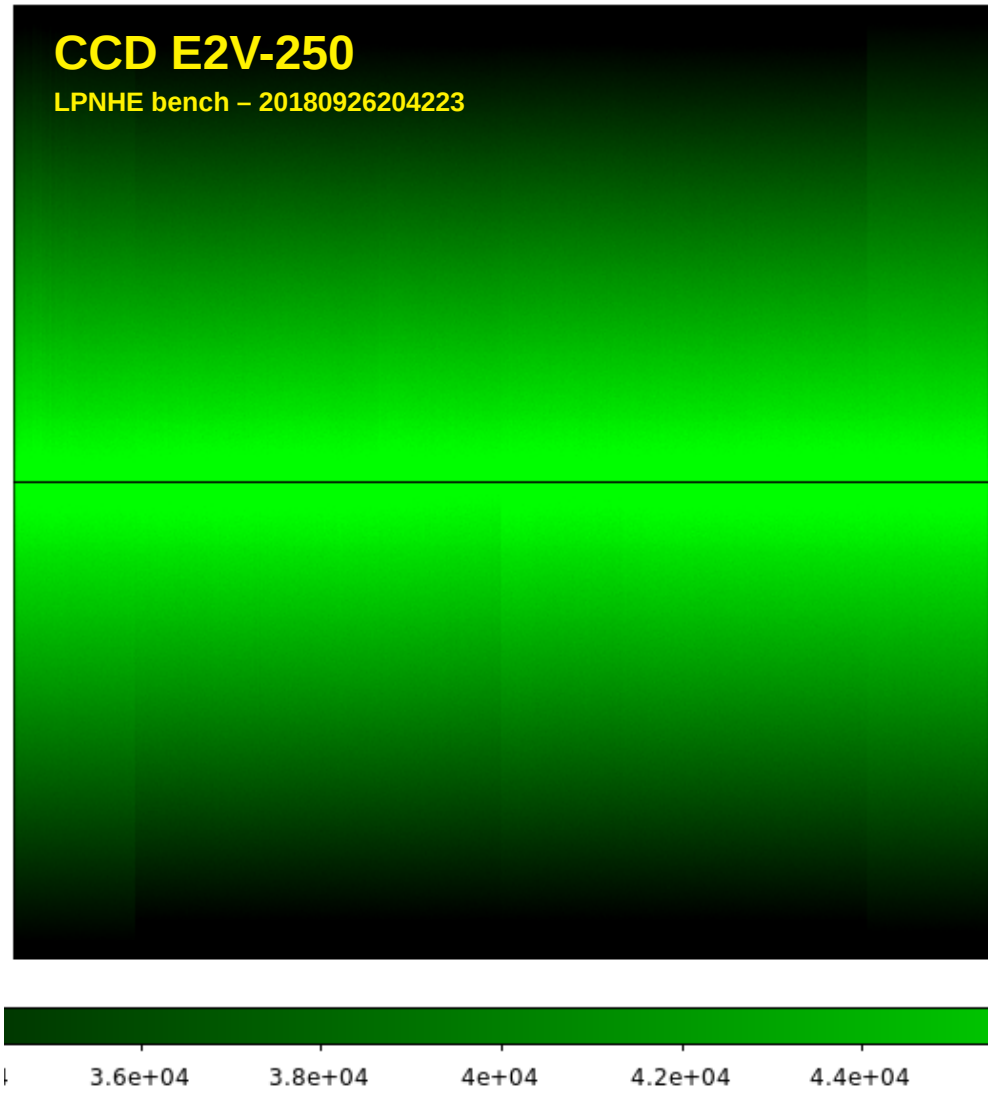
- **Motivation : study sensor & electronics linearity**
  - **At low flux** → may be important when stacking all LSST frames
  - Non-linearity feature ~42000 el (PTC study, Pierre & Pierre & ...)
- **Procedure** (tentative) **:**
  - Open the shutter
  - Repeat with increasing Δt :
    - **Move several lines UP**
    - **Wait (increasing) Δt**
    - **Trash some lines DOWN**
    - **Read a few (10) lines**
    - Clear the remaining
    - **Shutter stay open all time**
    - Repeat to fill the frame

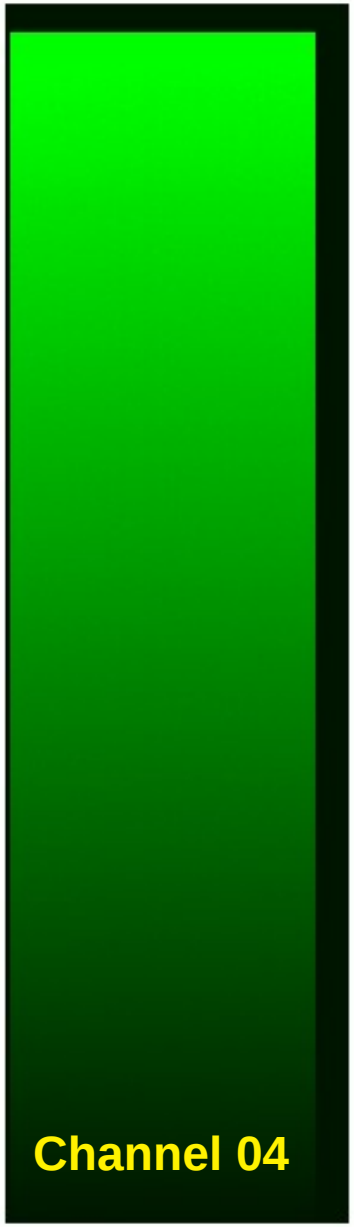


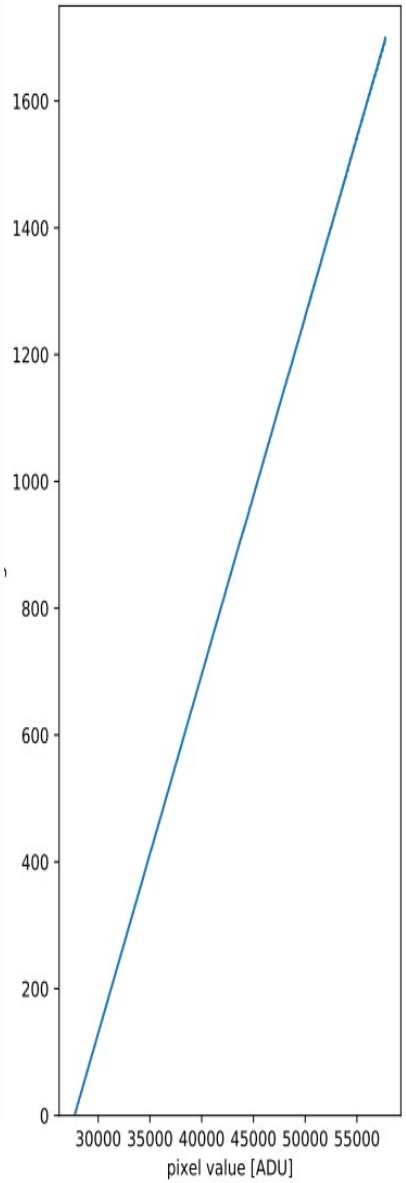
Tentative for a “linearity frame”

```
### ----- V2 ------------------------------------------------------------------

LinearityFlushFrame:  # Special frame to measure CCD linearity (v2, 20180926)
    JSR     FlushRegister
    CALL    StartOfImage
    SET     rep    0
    WHILE   rep < MaxRep   DO
        CALL    ParallelFlushOpen  repeat(DetectorRows)
        JSR     FlushRegisterOpen
        CALL    ReverseLineOpen    repeat(UpLines)
        CALL    FlushPixelOpen     repeat(6000*(rep))
        CALL    TransferLineOpen   repeat(TrashLines)
        JSR     FlushLineOpen
        JSR     WindowLineOpen     repeat(DownLines) # lines read out
        SET     rep    rep + 1
    DONE
    CALL    ReverseLine    repeat(2*DetectorRows)
    CALL    FlushPixel     repeat(DetectorCols)
    #
    # to fill the image and get the right size
    JSR     WindowLine     repeat(DetectorRows - MaxRep*DownLines)
    CALL    EndOfImage
    RTS

### ------------------------------------------------------------------------
```

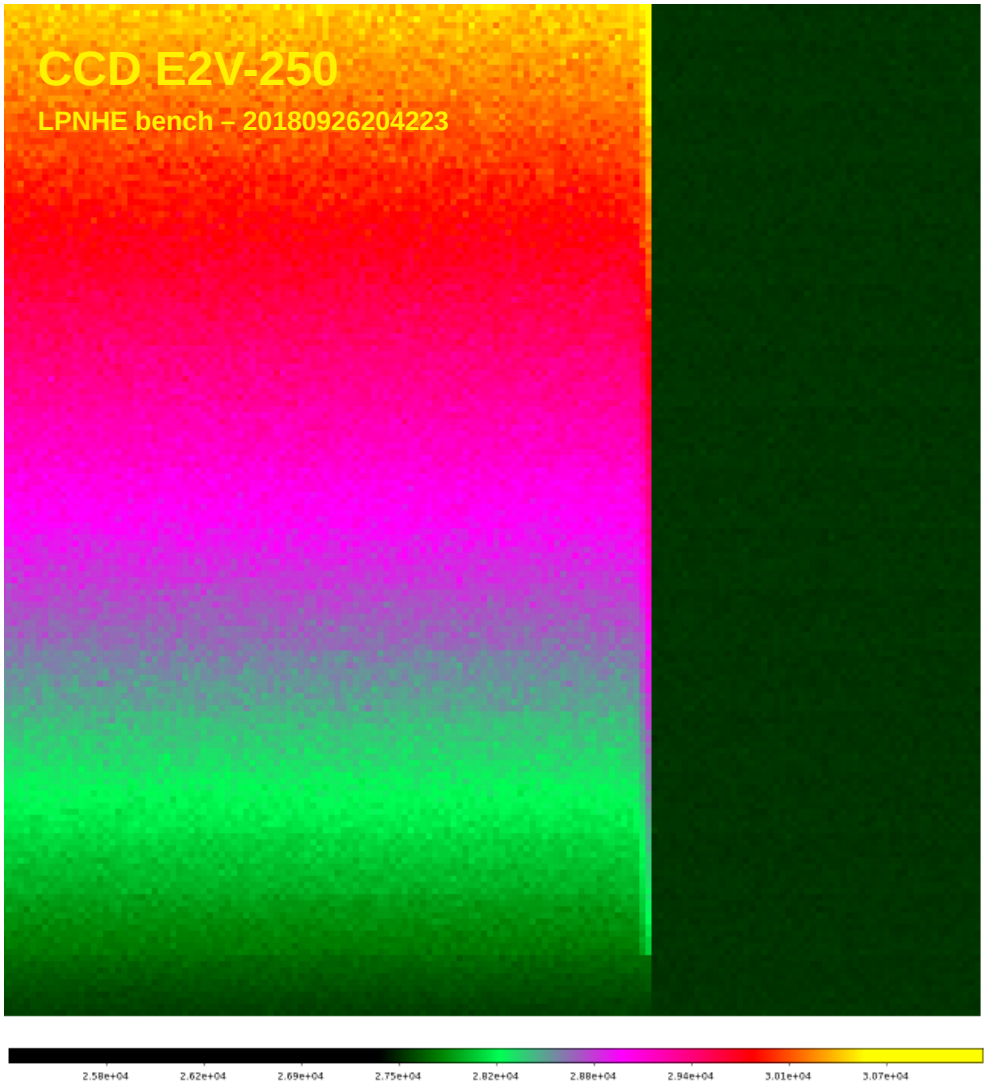
“Linearity” frames

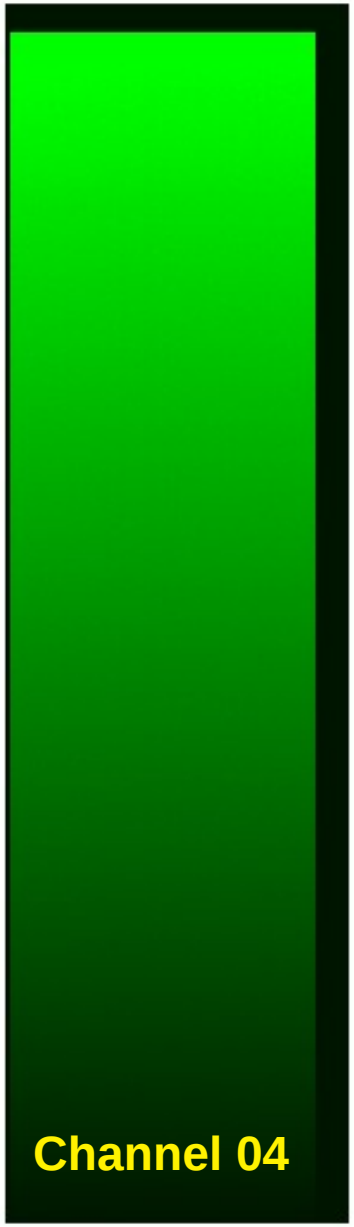


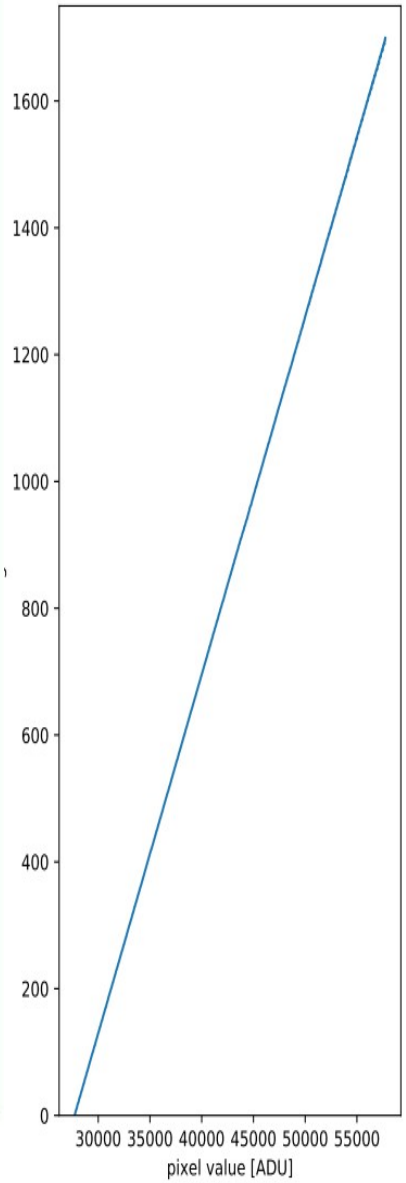




“Linearity” frames



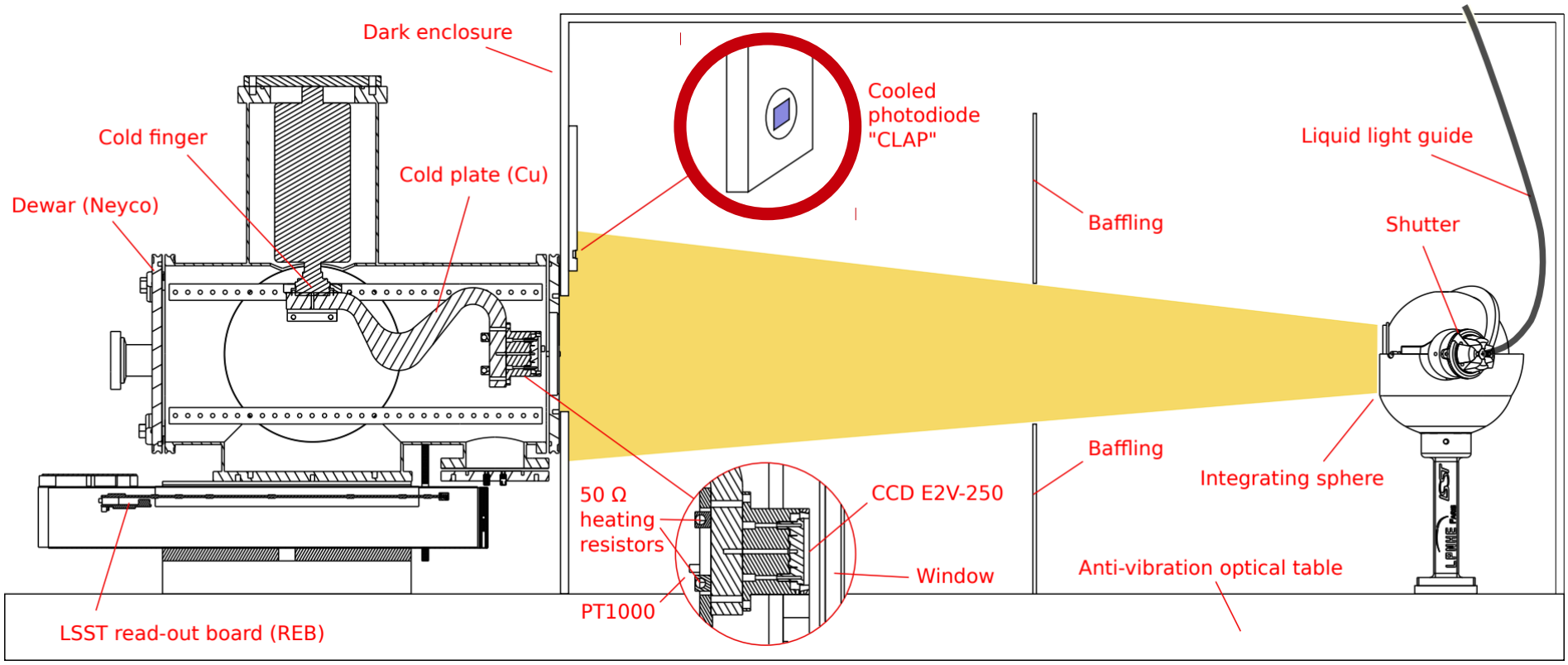




LPNHE LSST CCD testbench

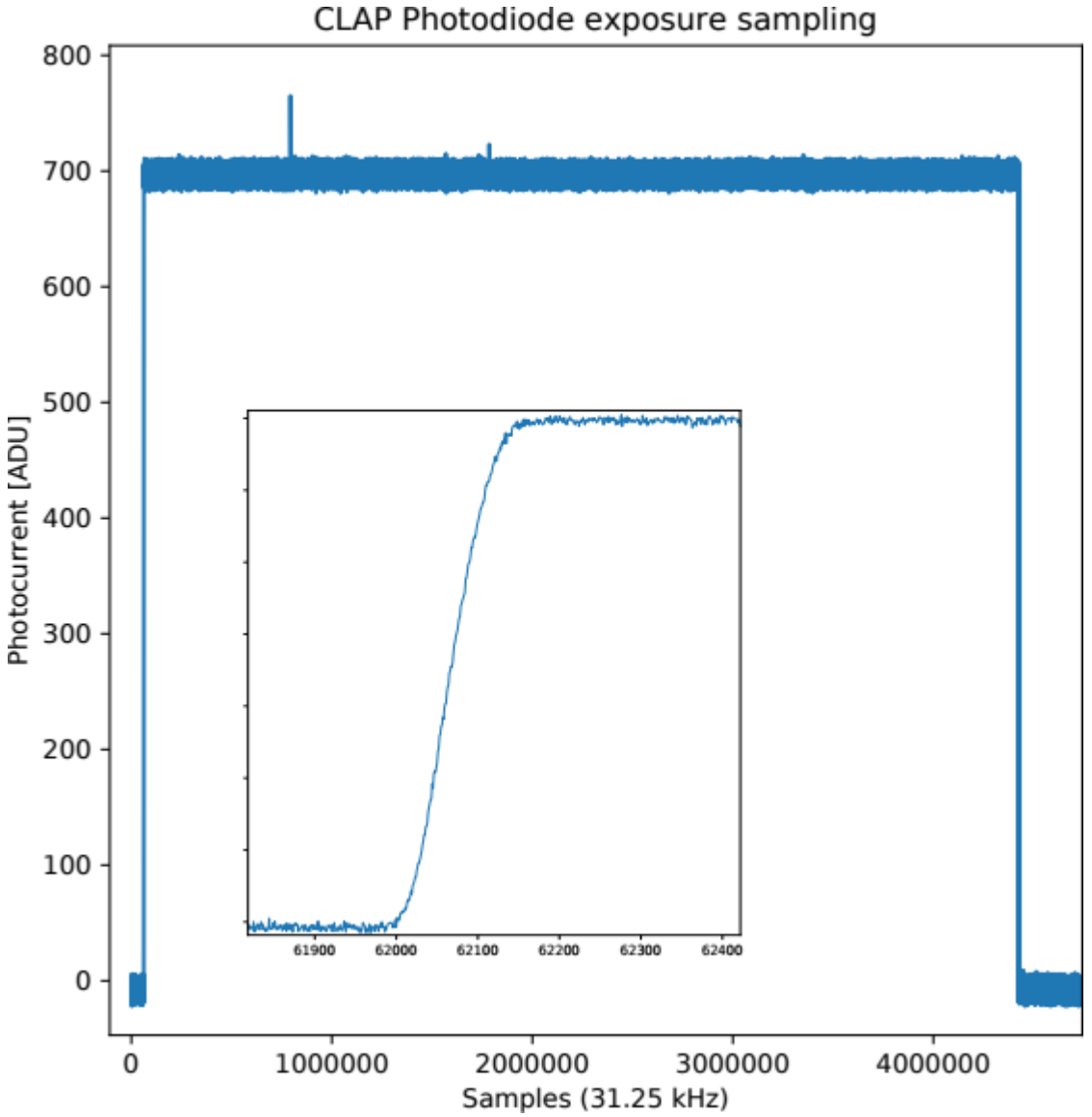
- **Monitoring the light flux during each exposure**

  → sampling at **31.25 kHz** with an home made photodiode electronics (CLAP)



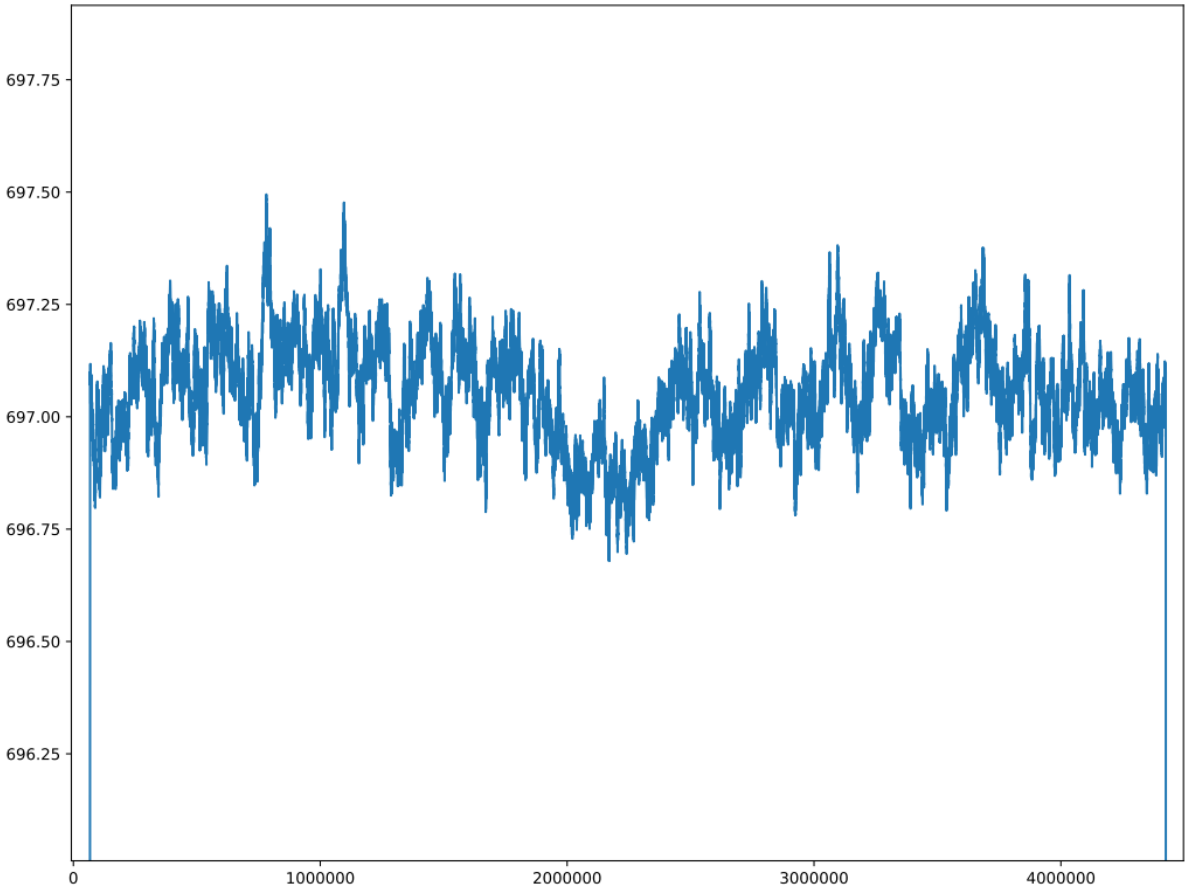
CLAP : Monitoring the light flux

- **Home made photodiode electronics** (CLAP)
- Monitoring the light level **during each exposure**
- Sampling at **31.25 kHz**
- Sensitive to
  - **Shutter opening / closing time**
  - **Illumination variations** during the exposure
- Allows to correct for the **effective exposure** integrated flux
- **Linearity studies**



CLAP : Monitoring the light flux

- **Home made photodiode electronics** (CLAP)
- Monitoring the light level **during each exposure**
- Sampling at **31.25 kHz**
- Sensitive to
  - **Shutter opening / closing time**
  - **Illumination variations** during the exposure
- Allows to correct for the **effective exposure** integrated flux
- **Critical for PTC & Linearity studies**



Smoothed CLAP data during a 128 s exposure

(moving averaged 5000 samples)

Tentative analysis (work in progress)

- **The Exposure Time**
  can be estimated **for each pixel (r,c)** and **per line**

```
exposure = 0

SerialRegisterFactor = 0.0
exposure += ( duration["FlushPixelOpen"] +
              c * duration["ReadPixelOpen"] * SerialRegisterFactor )

if c < PreScan:
    return exposure

exposure += ( (r + TrashLines) * duration["ReverseLineOpen"] +
              expwait * duration["FlushPixelOpen"] +
              TrashLines * duration["TransferLineOpen"] +
              duration["FlushLineOpen"] +
              r * duration["WindowLineOpen"] )
```

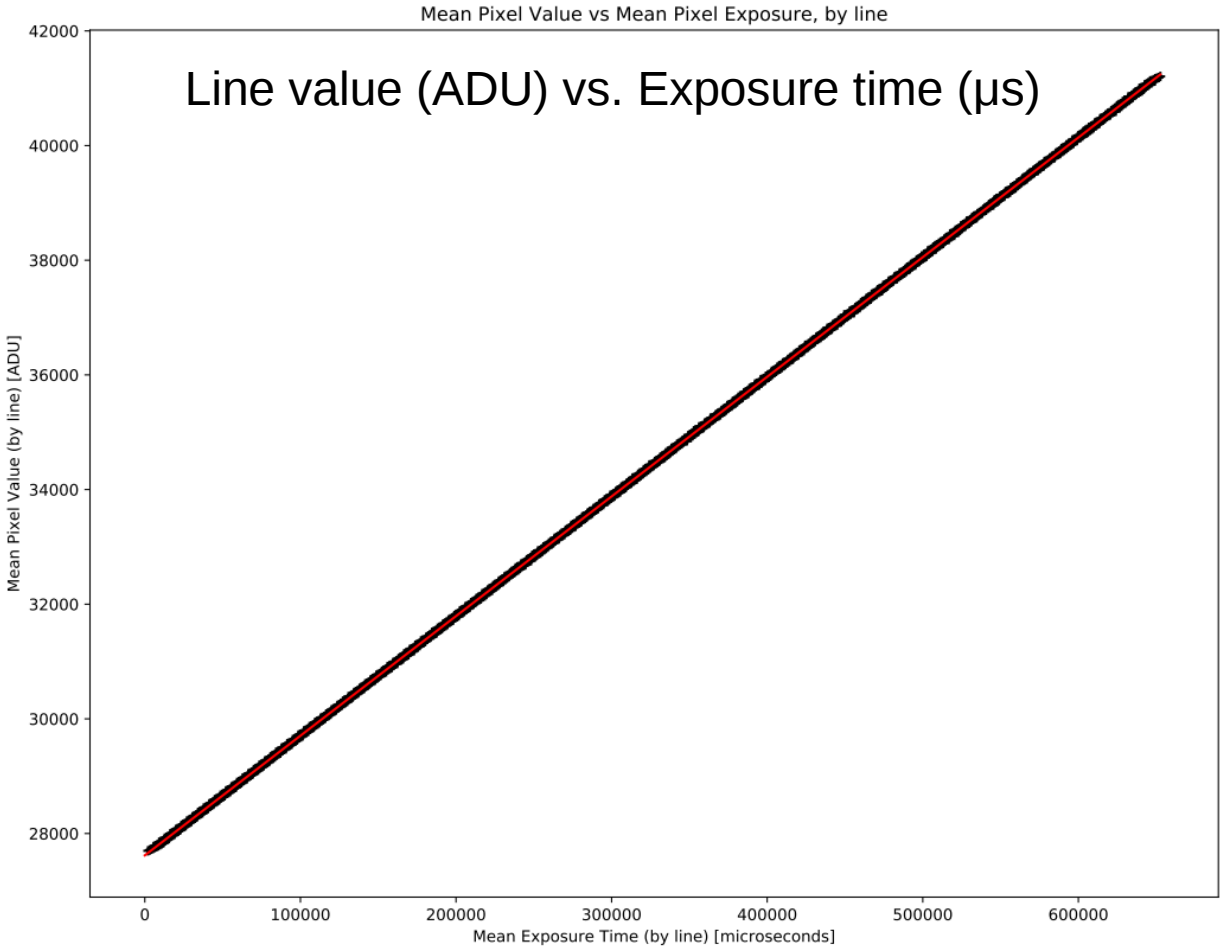
- **Illumination variations** could be **corrected** using the CLAP data

Tentative analysis (work in progress)

- **The Exposure Time** can be estimated **for each pixel (r,c)** and **per line**

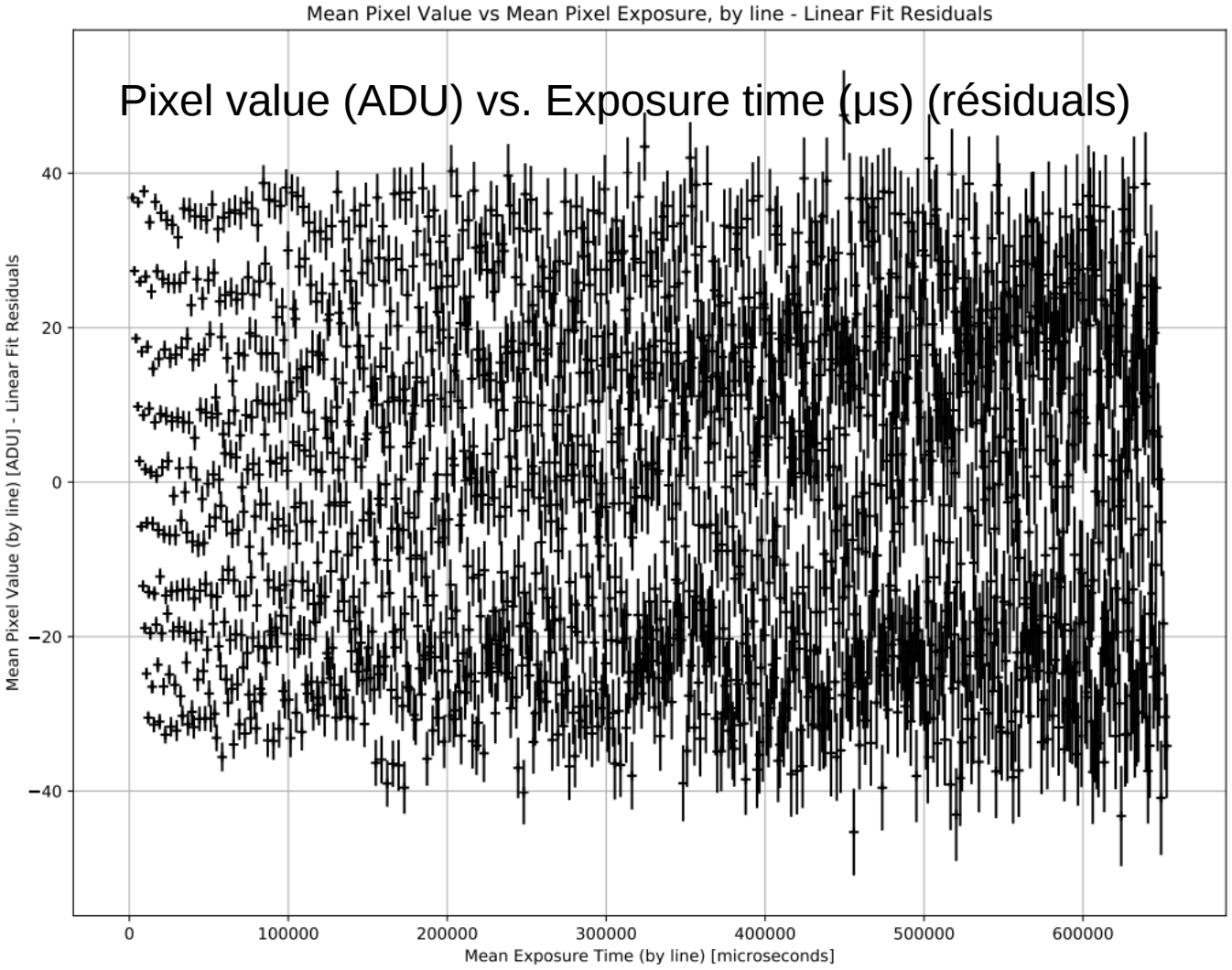
```
exposure = 0

SerialRegisterFactor = 0.0
exposure += ( duration["FlushPixelOpen"] +
              c * duration["ReadPixelOpen"] * SerialRe

if c < PreScan:
    return exposure

exposure += ( (r + TrashLines) * duration["ReverseLine
              expwait * duration["FlushPixelOpen"] +
              TrashLines * duration["TransferLineOpen"
              duration["FlushLineOpen"] +
              r * duration["WindowLineOpen"] )
```

- **Illumination variations** could be **corrected** using the CLAP data



Tentative analysis (work in progress)

- **Structured residuals**
- Reading blocks of **10 lines**…
- Hypothesis :
  - Shape of CCD lines **distorted** close to the sensor border (electrostatic)
  - Effective pixel size is different there
  - Resulting flux is affected during parallel transfers there ?
- Flux contributions in the **serial register** ?



Conclusion (work in progress)

- Development of **special sequencers programs**

  for **commissioning, instrument monitoring, calibrations**
- **« Linearity » frames** : tentative, still work to do
- To be tested with the CCS when ready (E. Aubourg)
- Analysis to take **electrostatic distorsions** and **other effects**
- **CLAP photodiode** : integration limited to 250 s in the current mode
- LSST DAQ timeout : DAQ has to be **fed with pixels regularly**

  → timeout of about 18 s (DAQ v 1.something at LPNHE)

- ***Creating and testing LSST REB sequencer programs is fun ;-)***