BAORadio TAcq software

R. Ansari

December 2012

Software requirements

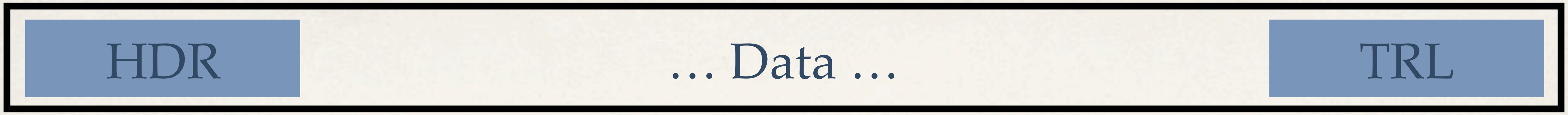
- An important component to ensure efficient collaboration
- Would be helpful to use a common base layer (library and associated tools) for developing the acquisition and key components of the data analysis software
- Required qualities: robustness, efficiency and reliability Efficiency, reliatibility, the question of maintenance & evolution should be addressed
- Acquisition software should handle different observation modes (electronic calibration & tests, sky source calibration, normal observation mode …)
- Data storage format & scheme, should cover all the experiment needs (Science Data, Housekeeping data) - should ensure easy exchange of the processed data
- Ensure an efficient implementation of the computing intensive part (visibility computation, sky map reconstruction …)
- Should be flexible enough to adapt to different hardware configuration (CPU, GPU …)
- Leave enough freedom to scientists for high level data analysis

BAORadio Acquisition/Processing software

- Multi-thread programs / object oriented (C++) architecture
- FITS files (+ directory structure) for data storage (raw data/Fourier coefficient/Visibilities/spectra)
- **BRPaquet** hardware / software data exchange unit
- **RAcqMemZoneMgr** (memory manager) insure thread synchronisation/ coordination
- Thread objects (ZThread) perform differents tasks
- Uses the **SOPHYA** C++ class library

http://www.sophya.org

BRPaquet

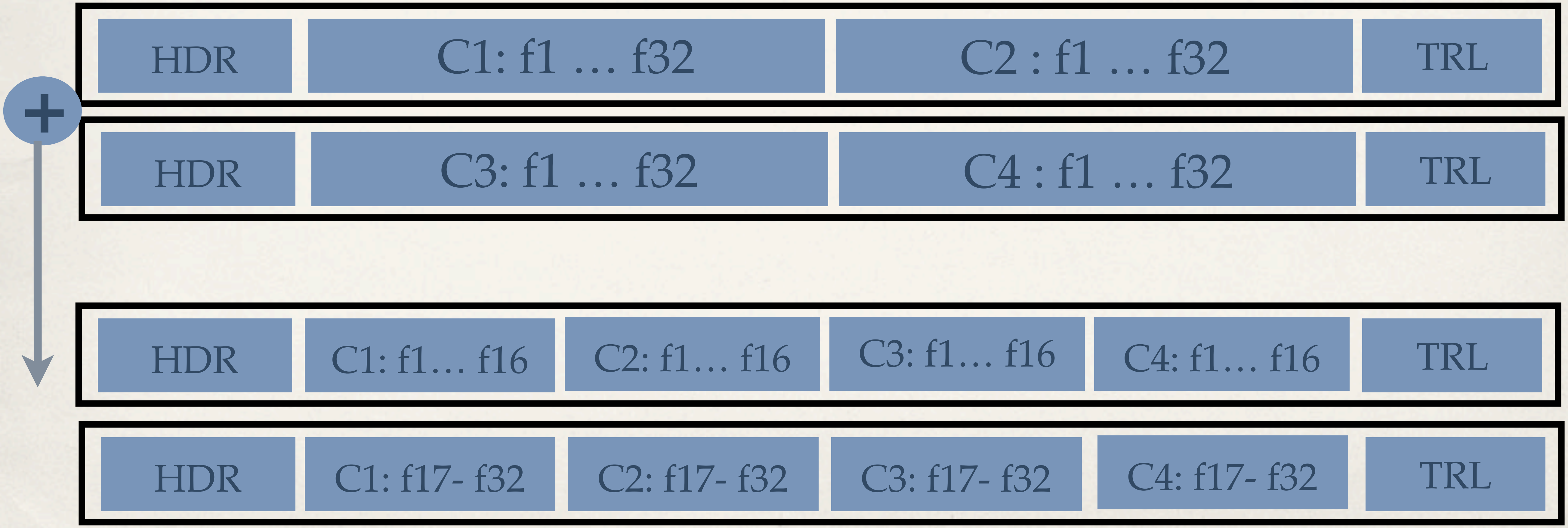


HDR | … Data … | TRL

- HDR/TRL: Data description, HW Time Tag, paquet length …
- Data : Raw or FFT coefficients, one or multiple channels
- HDR+TRL = 40 Bytes, Paquet length (variable)

HDR :

- Packet length
- DataDesc
- TimeTag
- FrameCounter



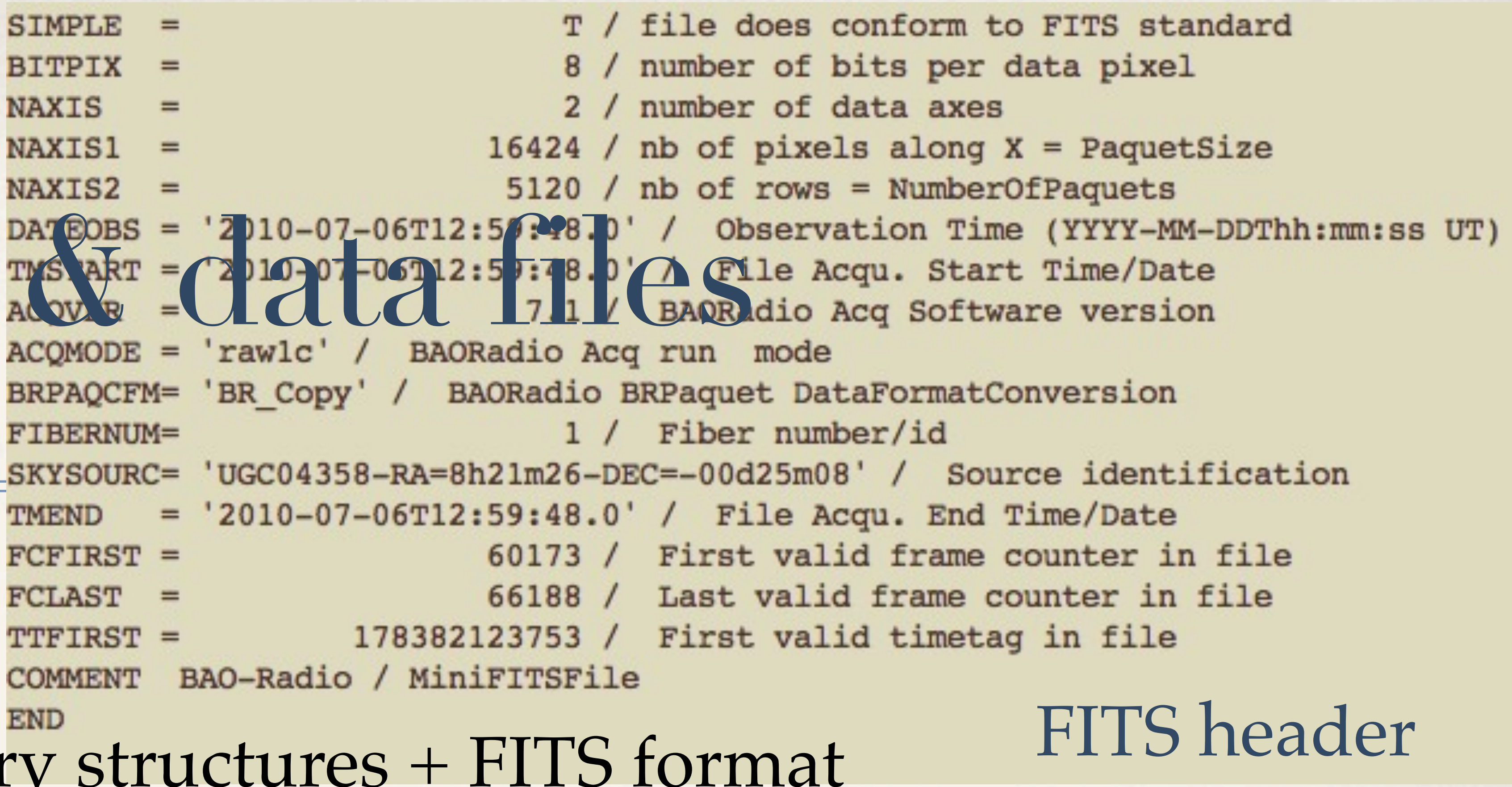
HDR | C1: f1 … f32 | C2 : f1 … f32 | TRL

HDR | C3: f1 … f32 | C4 : f1 … f32 | TRL

HDR | C1: f1… f16 | C2: f1… f16 | C3: f1… f16 | C4: f1… f16 | TRL

HDR | C1: f17- f32 | C2: f17- f32 | C3: f17- f32 | C4: f17- f32 | TRL

FITS format & data files



```
SIMPLE  =                    T / file does conform to FITS standard
BITPIX  =                    8 / number of bits per data pixel
NAXIS   =                    2 / number of data axes
NAXIS1  =                16424 / nb of pixels along X = PaquetSize
NAXIS2  =                 5120 / nb of rows = NumberOfPaquets
DATEOBS = '2010-07-06T12:59:48.0' /  Observation Time (YYYY-MM-DDThh:mm:ss UT)
TMSTART = '2010-07-06T12:59:48.0' /  File Acqu. Start Time/Date
ACQVER  =                 7.1 /  BAORadio Acq Software version
ACQMODE = 'rawlc' /  BAORadio Acq run  mode
BRPAQCFM= 'BR_Copy' /  BAORadio BRPaquet DataFormatConversion
FIBERNUM=                    1 /  Fiber number/id
SKYSOURC= 'UGC04358-RA=8h21m26-DEC=-00d25m08' /  Source identification
TMEND   = '2010-07-06T12:59:48.0' /  File Acqu. End Time/Date
FCFIRST =                60173 /  First valid frame counter in file
FCLAST  =                66188 /  Last valid frame counter in file
TTFIRST =         178382123753 /  First valid timetag in file
COMMENT  BAO-Radio / MiniFITSFile
END
```

FITS header

- Output files uses directory structures + FITS format
- Raw data (time samples), FFT data and visibility matrices written in FITS format
- BRPaquet complete structure written to FITS files for raw/fft data dumps
- FITS headers used to add auxiliary informations, such time time&date, start-end timetag, pointing/source identification …
- Visibility matrices written also in FITS format

http://heasarc.gsfc.nasa.gov/fitsio/

Some of the TAcq classes

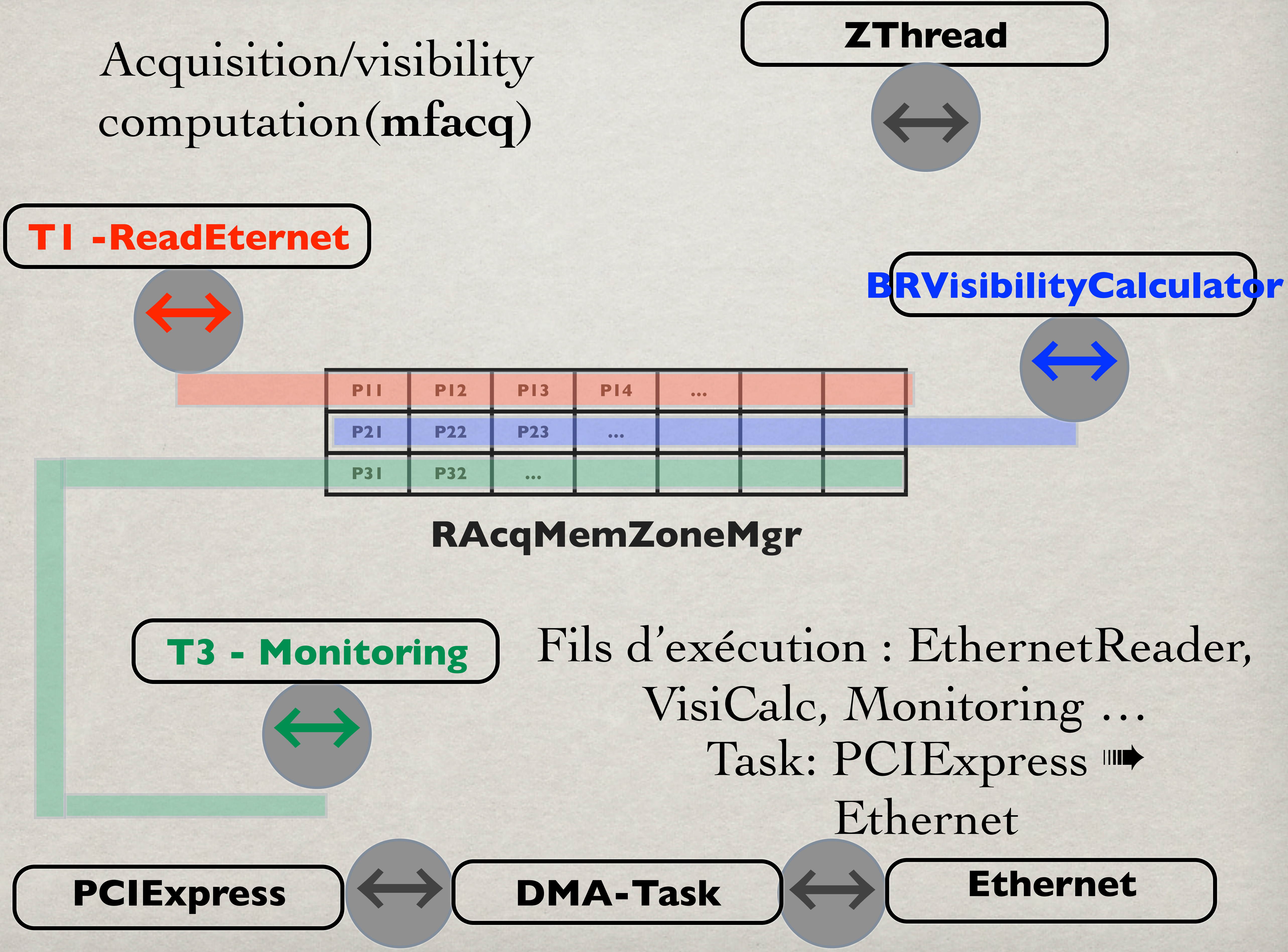
- Class **BRPaquet** & **BRPaqChecker**
- Class **RAcqMemZoneMgr** (Multi fiber / link managed memory zone)
- Thread (task) classes **(ZThread)** (see next slide)
- Class **MiniFITSFile**

http://bao.lal.in2p3.fr/TAcq/
http://www.sophya.org

Thread (Task) classes

- **PCIEMultiReader** (PCI-Express DMA to memory)
- **PCIEToEthernet** (PCI-Express DMA to ethernet)
- **MultiDataSaver** (Writes packet data to disk (FITS format))
- **EthernetReader** (Read packet from ethernet to memory)
- **BRMultiFitsReader** (Multi fiber fits reader, align in time)
- **BRVisibilityCalculator** (Computes visibilities)
- **BRFFTCalculator** (perform FFT on raw data)
- **MonitorProc(s)** (Monitoring during processing)

Acquisition/visibility computation(**mfacq**)



P11	P12	P13	P14	...		
P21	P22	P23	...			
P31	P32	...				

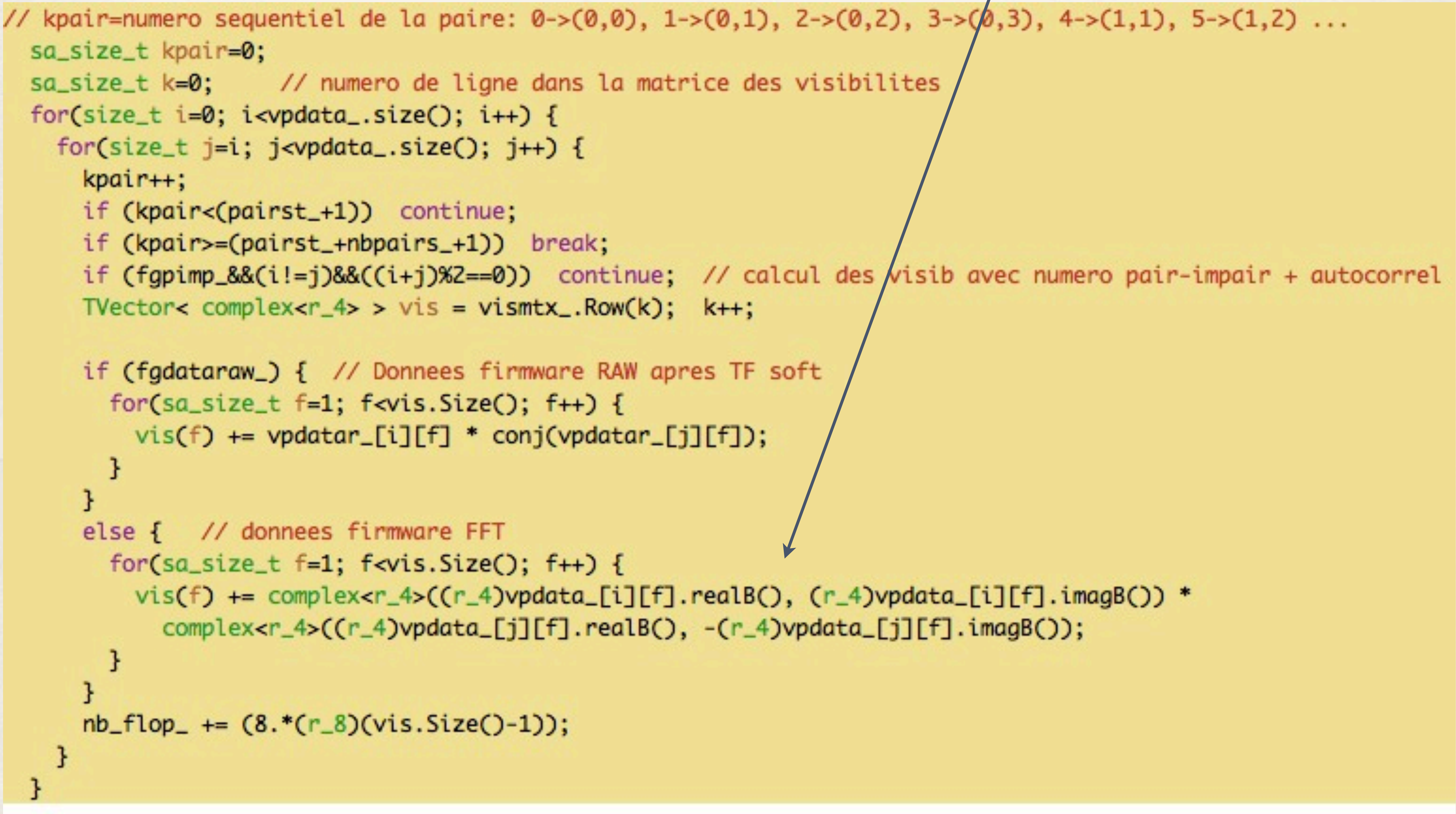
Fils d'exécution : EthernetReader, VisiCalc, Monitoring …
Task: PCIExpress ➠ Ethernet

Flexible CPU or GPU correlator

BRVisibilityCalculator can easily be adapted to perform the heavy calculation on GPU …

BRVisibilityCalculator code has > 800 lines

less than 20-50 lines has to be executed on GPU …



```
// kpair=numero sequentiel de la paire: 0->(0,0), 1->(0,1), 2->(0,2), 3->(0,3), 4->(1,1), 5->(1,2) ...
  sa_size_t kpair=0;
  sa_size_t k=0;     // numero de ligne dans la matrice des visibilites
  for(size_t i=0; i<vpdata_.size(); i++) {
    for(size_t j=i; j<vpdata_.size(); j++) {
      kpair++;
      if (kpair<(pairst_+1))  continue;
      if (kpair>=(pairst_+nbpairs_+1))  break;
      if (fgpimp_&&(i!=j)&&((i+j)%2==0))  continue;  // calcul des visib avec numero pair-impair + autocorrel
      TVector< complex<r_4> > vis = vismtx_.Row(k);  k++;

      if (fgdataraw_) {  // Donnees firmware RAW apres TF soft
        for(sa_size_t f=1; f<vis.Size(); f++) {
          vis(f) += vpdatar_[i][f] * conj(vpdatar_[j][f]);
        }
      }
      else {   // donnees firmware FFT
        for(sa_size_t f=1; f<vis.Size(); f++) {
          vis(f) += complex<r_4>((r_4)vpdata_[i][f].realB(), (r_4)vpdata_[i][f].imagB()) *
            complex<r_4>((r_4)vpdata_[j][f].realB(), -(r_4)vpdata_[j][f].imagB());
        }
      }
      nb_flop_ += (8.*(r_8)(vis.Size()-1));
    }
  }
```