CDAS e accesso ai dati

Fabio Convenga (INFN Lecce)

OVERVIEW

- Monitoring
- Estrazione ADC UUB
- Estrazione parametri trigger
- Estrazione istogrammi di calibrazione

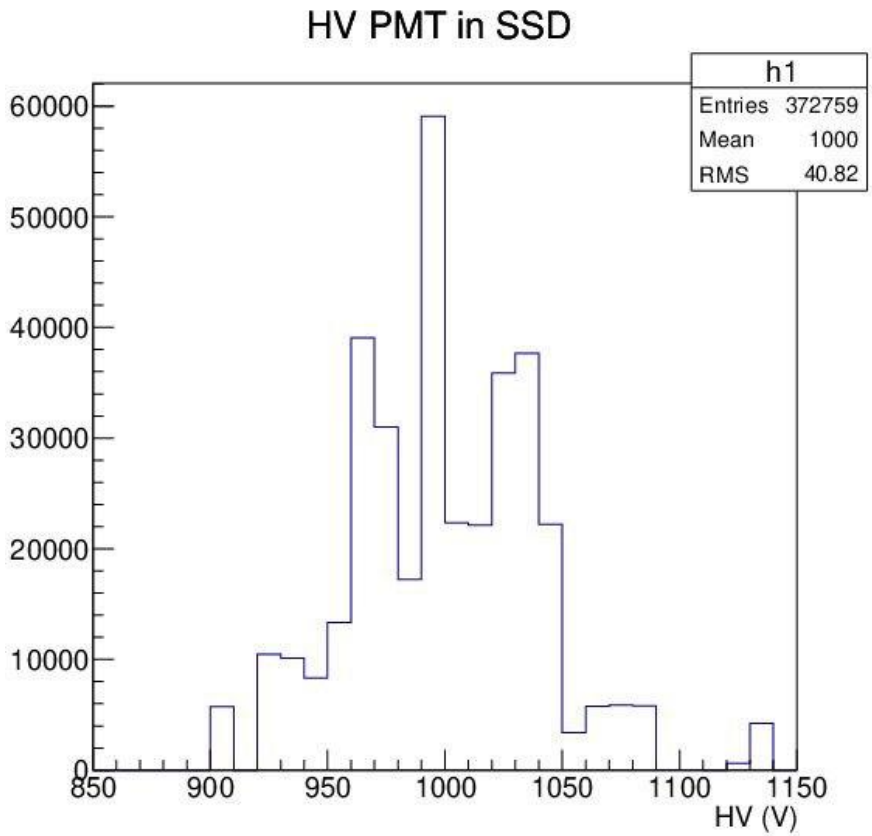
MONITORING

I dati di monitoring da cui vengono anche generati i grafici del sito sono nei file **mc_2021***

Per estrarre i dati può essere usato l'esempio scritto nel sorgente **fillUUB.cc**

Attualmente è presente la versione più aggiornata coerente con la versione ultima dello SC.

Per selezionare le UUB si usa la variabile **fRawMonitoring.fIsUUB (0 se UB, 1 se UUB)**



ESTRAZIONE ADC UUB

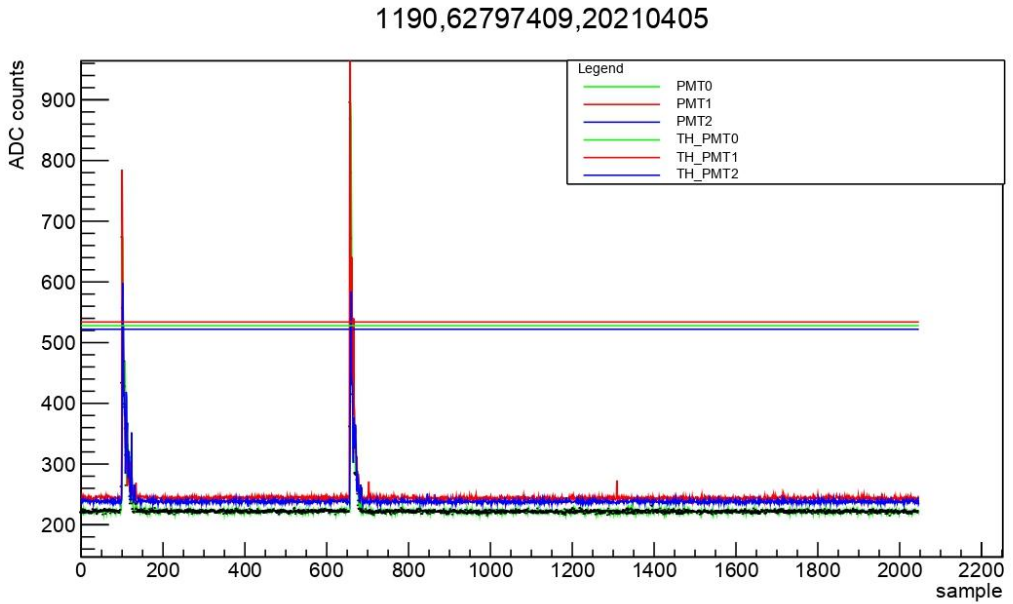
I dati raw **sd_2021*** possono essere estratti con la libreria **IoSd** di CDAS. Questa libreria può essere usata anche da OffLine.

Per il momento conviene usare la **IoSd** di CDAS DAQ per AugerPrime. CDAS user non è ottimizzato.

```
/// Upgraded Surface detector Flash ADC class
class IoUSdFadc {
public:
  IoUSdFadc() {
    NSample = 0;
    CodingVersion=0;
  }
  virtual ~ IoUSdFadc() {}
  UInt_t NSample;              ///< Number of samples
  UInt_t CodingVersion;        ///< How the FADC have bee
  UInt_t TraceStart;           ///< Internal, where the t
  UInt_t ShwrBufSt;
  UInt_t RdBufSt;
  vector <unsigned short> Traces;  ///< Buffer memory wh
  short int GetValue(int pmt, int gain, int bin); ///< r
  short GetValueRd(int pmt, int gain, int bin); ///< ret
  short GetRawValue(int pmt, int gain, int bin);


  bool GetValueParity(int pmt, int gain, int bin); ///<

  ClassDef(IoUSdFadc, 3)
};
```



TRIGGER

I trigger che sono definiti localmente per tank sono quattro:

- **Compatibility** single bin trigger
- **Compatibility** ToT
- **Compatibility** ToTd
- **Compatibility** MoPS
- Full bandwidth single bin trigger

Questi trigger funzionano solo sui PMT della tank. Funzionano sulle tracce ADC **filtrate** e **campionate** a 40 MHz.

Bits	Description
11:0	WCD PMT0 low gain ADC
15:12	Low order 4 bits of filtered PMT0
27:16	WCD PMT0 high gain ADC
31:28	Middle 4 bits of filtered PMT0

Funziona anche su SSD. Attualmente non è impostato nell'acquisizione dati. Viene usato unicamente durante la calibrazione.

ESTRAZIONE ADC UUB

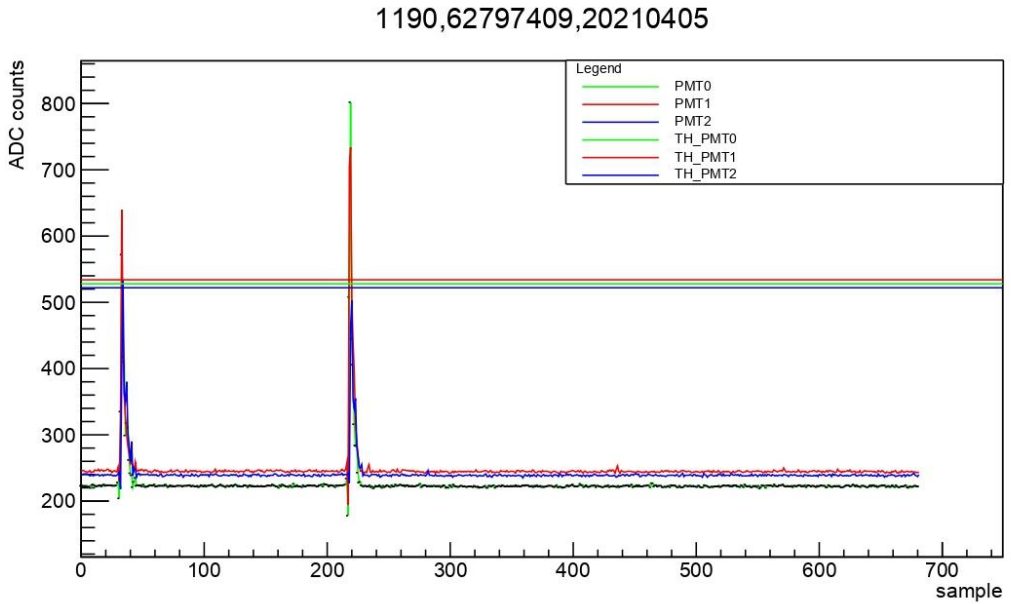
Nel caso di analisi sui trigger può essere necessario estrarre le tracce filtrate.

Attualmente non sono spacchettate. Bisogna estrarle dai buffer e campionarle usando ENABLE40

```
/// Upgraded Surface detector Flash ADC class
class IoUSdFadc {
public:
  IoUSdFadc() {
    NSample = 0;
    CodingVersion=0;
  }
  virtual ~ IoUSdFadc() {}
  UInt_t NSample;              ///< Number of samples
  UInt_t CodingVersion;        ///< How the FADC have bee
  UInt_t TraceStart;           ///< Internal, where the t
  UInt_t ShwrBufSt;
  UInt_t RdBufSt;
  vector <unsigned short> Traces;  ///< Buffer memory wh
  short int GetValue(int pmt, int gain, int bin); ///< r
  short GetValueRd(int pmt, int gain, int bin); ///< ret
  short GetRawValue(int pmt, int gain, int bin);


  bool GetValueParity(int pmt, int gain, int bin); ///<

  ClassDef(IoUSdFadc, 3)
};
```



ESTRAZIONE PARAMETRI TRIGGER

Con CDAS si possono estrarre i parametri di trigger ed inoltre si possono discriminare i vari tipi di trigger.

Analisi di rate per tipo di trigger.

Esempio di due eventi CSB trigger

DATE	Nevent	Tank	TH_0	TH_1	TH_2	VEMC_0	VEMC_1	VEMC_2	VEMP_0	VEMP_1	VEMP_2	ENABLE_REGISTER
20210412	62887754	1819	569	562	522	3171.2	3319.2	2971	172.3	172.2	160.2	248
DATE	**Nevent**	**Tank**	**TH_0**	**TH_1**	**TH_2**	**VEMC_0**	**VEMC_1**	**VEMC_2**	**VEMP_0**	**VEMP_1**	**VEMP_2**	**ENABLE_REGISTER**
20210413	62907626	840	540	521	554	2975.9	2853.7	2842.5	160.4	154.3	154.2	248

Bit(s)	Bit Mask (or << Shift)	Description
3	COMPATIBILITY_SB_TRIG_INCL_PMT0	Include PMT0 in multiplicity logic if set
4	COMPATIBILITY_SB_TRIG_INCL_PMT1	Include PMT1 in multiplicity logic if set
5	COMPATIBILITY_SB_TRIG_INCL_PMT2	Include PMT2 in multiplicity logic if set
7:6	<<COMPATIBILITY_SB_TRIG_COINC_LVL_SHIFT	Coincidence level for multiplicity logic sub-trigger
9		Require 2 consecutive bins above threshold if set

ESTRAZIONE PARAMETRI TRIGGER

```
class IoUSdTrigParam {
public:
  IoUSdTrigParam() {
    PL_version=0;
  }
  virtual ~ IoUSdTrigParam() {}

  UInt_t PL_version;
  UInt_t TrigMask;

  //Compatibility Single bin trigger
  UInt_t csbt_th[3];
  UInt_t csbt_enable;

  //Compatibility TOT
  UInt_t ctot_th[3];
  UInt_t ctot_enable;
  UInt_t ctot_occ;

  //Compatibility TOTD
  UInt_t ctotd_thmin[3];
  UInt_t ctotd_thmax[3];
  UInt_t ctotd_enable;
  UInt_t ctotd_occ;
  UInt_t ctotd_fd;
  UInt_t ctotd_fn;
  UInt_t ctotd_int;

  //Compatibility MOPS
  UInt_t cmops_thmin[3];
  UInt_t cmops_thmax[3];
  UInt_t cmops_enable;
  UInt_t cmops_occ;
  UInt_t cmops_ofs;
  UInt_t cmops_int;

  //Single bin trigger (full band width mode)
  UInt_t sbt_th[4]; // thresholds . 0-2: WCD PMTs; 3- SSD
  UInt_t sbt_enable;

  //Led flasher
  UInt_t led_ctrl;

  //random trigger.
  UInt_t random_mode;

  ClassDef(IoUSdTrigParam, 1)

};
```

- **ToT:**
  - TH, Occupancy
- **ToTd:**
  - THmin, THmax, Occupancy, FD, FN, INT
- **MoPS:**
  - THmin, THmax, Occupancy, OFS, INT
- **FBW single bin:**
  - TH (quattro questa volta)

Tutti i trigger hanno un registro ENABLE per le impostazioni sulle coincidenze.

ESTRAZIONE ISTOGRAMMI DI CALIBRAZIONE

In IoSd sono salvati gli istogrammi di carica e di picco per dei PMT e SSD

```
/// Surface detector Muon histograms
class IoSdHisto {
public:
  IoSdHisto() {}
  virtual ~ IoSdHisto() {}
  Int_t type; ///< 1-> normal SSD histogram;
              ///< 2 -> SSD working with SiPM
              ///< 3 -> SiPM calibration histogram

  UShort_t Offset[kIoSd::NB_HISTO_CALIB];
  UShort_t Base[kIoSd::NPMT][20];     ///< Histogram of baseline
  UShort_t Peak[kIoSd::NPMT][150];    ///< Peak histogram
  UShort_t Charge[kIoSd::NPMT + 1][600];      ///< Charge histogram (4th histogram is for the sum of the PMTs)
  UInt_t Shape[kIoSd::NPMT][kIoSd::SINGLE_MUON_SIZE]; ///< Average muon shape
  // UUB: have 5 channels, but right now SPMT is not in slow buffers (normal, never has signal)
  // So we end up with 4 channels
  // 3 usual PMTs
  // SSD is like a normal PMT, but sharper
  // no sum histogram, as it is useless
  // FIXME: we need for the UUB:
  // 4 baseline histograms (we are missing the SSD right now)
  // 4 peak/charge histograms, with same binning as before (missing 1 peak, and eventually 1 charge)
  // 4 shapes with ~3 times the number of bins (Set in FPGA, currently 70), need to replace the Shape
  //
  // EA version: has extra UShape
  //
  Int_t UShape[4][70]; ///< currently only 69 are used...
  UShort_t Base3[20];  ///< SSD baseline histogram, would be Base[3][20] if ROOT allowed evolution...
  UShort_t Peak3[150];  ///< SSD peak histogram, would be Peak[3][20] if...
  UShort_t Offset3[3]; ///< SSD baseline offset, peak offset, charge offset

  float BaseAvg[4];

  Char_t Pk_bit_shift[4],Ch_bit_shift[4];
  UShort_t Pk_bin[2][4],Ch_bin[2][4];

  Int_t tStart,tEnd,Entries;
  ClassDef(IoSdHisto, 4)
};
```

Gli istogrammi SSD sono salvati nella quarta posizione del vettore Charge al posto dell'istogramma della somma dei PMT.

Il picco è salvato in un vettore separato da quello degli altri PMT.

Questi vettori non possono essere plottati perché l'informazione all'interno di essi è hardcoded da Ricardo. Attualmente in CDAS user non ci sono funzioni in grado di leggere questi vettori e restituire gli istogrammi per la parte di UUB. Ricardo, tuttavia, ha rilasciato una versione di IoSd in cui vi sono le funzioni aggiornate per UUB.