Elettra Sincrotrone Trieste

ESS WS control system development status

Introduction - 1

preamble

Software is just a tool that lets you to shift a task from the conceptual side (model) to the computing side (practical implementation of the model)

Head first: before coding anything define a model describing what you want to be performed by an automatic system

Defining a good (enough) model is a complex task that requires to identify the input, the output and the action (transfer function) that have to be performed in order to reach a goal

goal

measuring the transverse beam density profile by WS

Introduction - 2

what is a WS

WS: an electro-mechanical device which measures the transverse beam density profile in a particle accelerator by means of a moving thin wire (CERN)
WS input: signal generated by a moving wire (wire: primary sensing element)
WS output: transverse beam density profile
➔ beam density profile = f(wire position, time, ...)

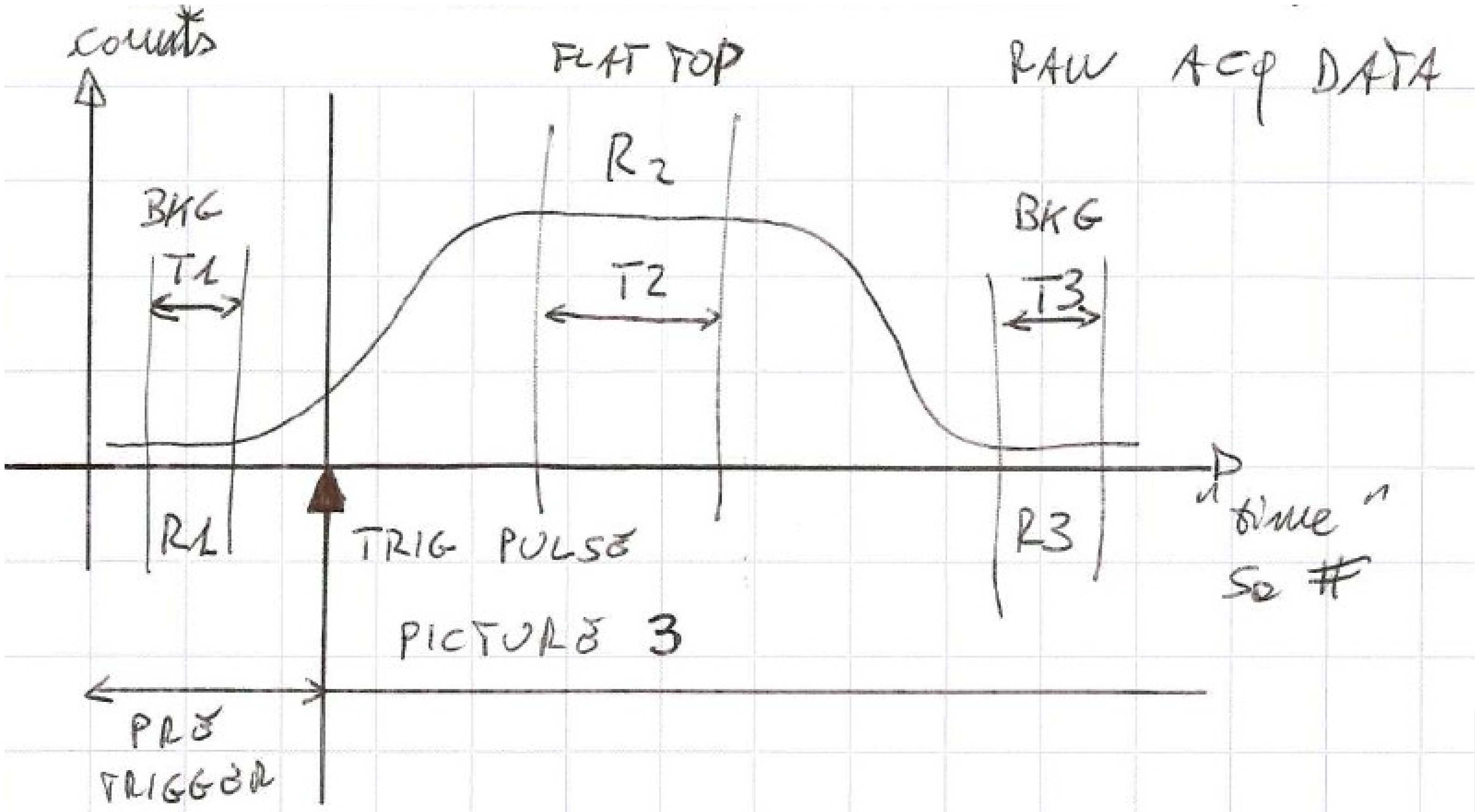
main SW goal

to implement f() by the identification, development and connection of suited functional blocks that can be built by code

Introduction - 3

Expected input pulse (theoretical shape)

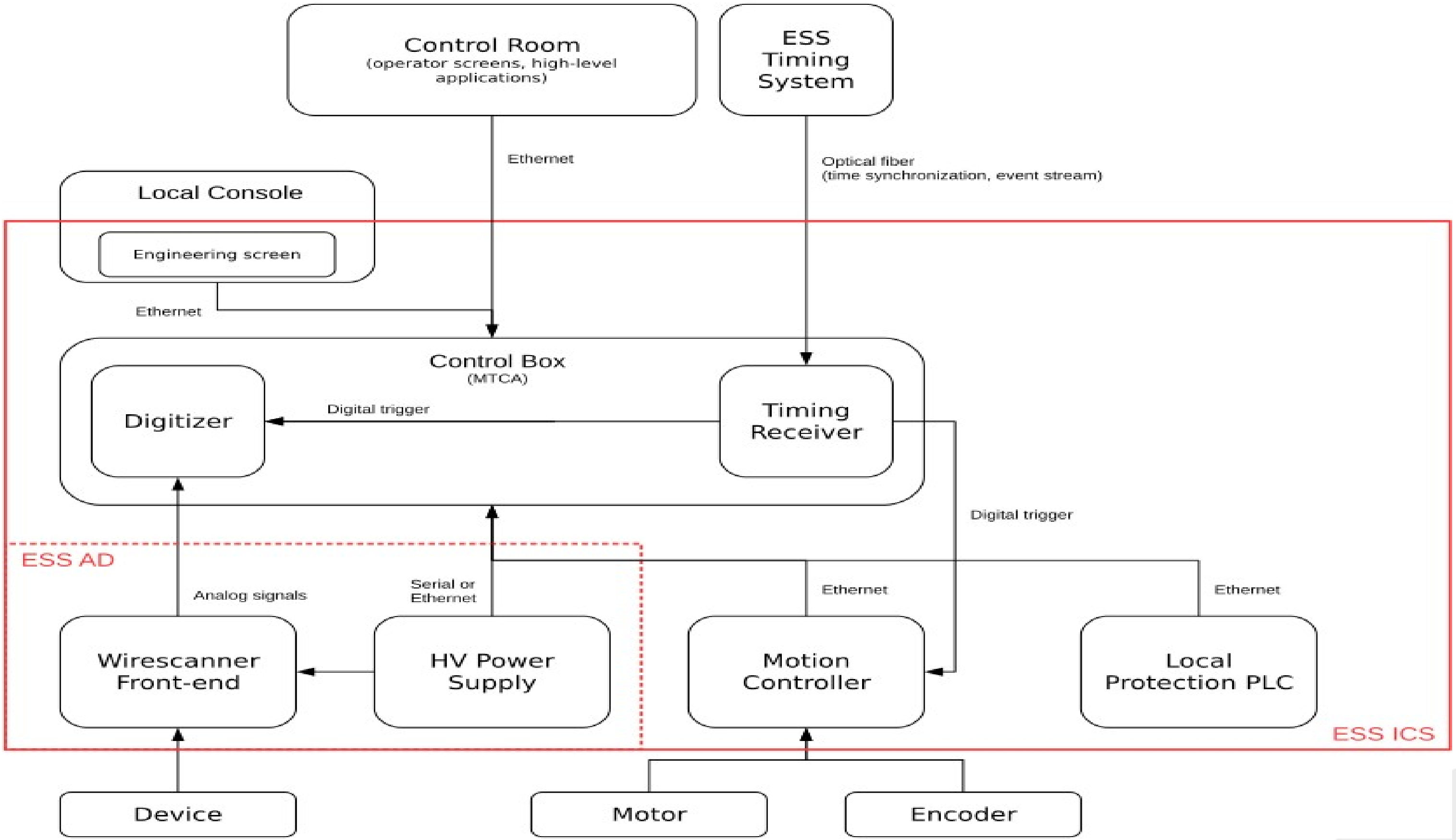
- per channel;
- per mechanical position;
- synchronized by timing system;



SW - 1

Conceptual HW-SW architecture

WS conceptual HW layout suggests a two layers client/server SW architecture



SW - 2

Two layers SW architecture

The high level layer (client side) is in charge of human interaction activities (control panel (OPI) and Engineering Screen)

The low level layer (server side) is in charge of all kind of HW access (device driver) and the logical functionality strictly related to the implementation of the WS transfer function

Ancillary low level tasks are protection and servicing

Constrain: EPICS based

SW - 3

Expected SW functions – raw estimation

Motion control: scan “trajectory” (step/fly mode) synchronized with the timing system

Data acquisition synchronized with the timing system

Elaboration of the acquired raw data accordingly to specified algorithms

Data presentation

Data archiviation

Auto and procedure driven testing

Data exchange (communication protocols – CA, TCP/IP, Ethercat, ...)

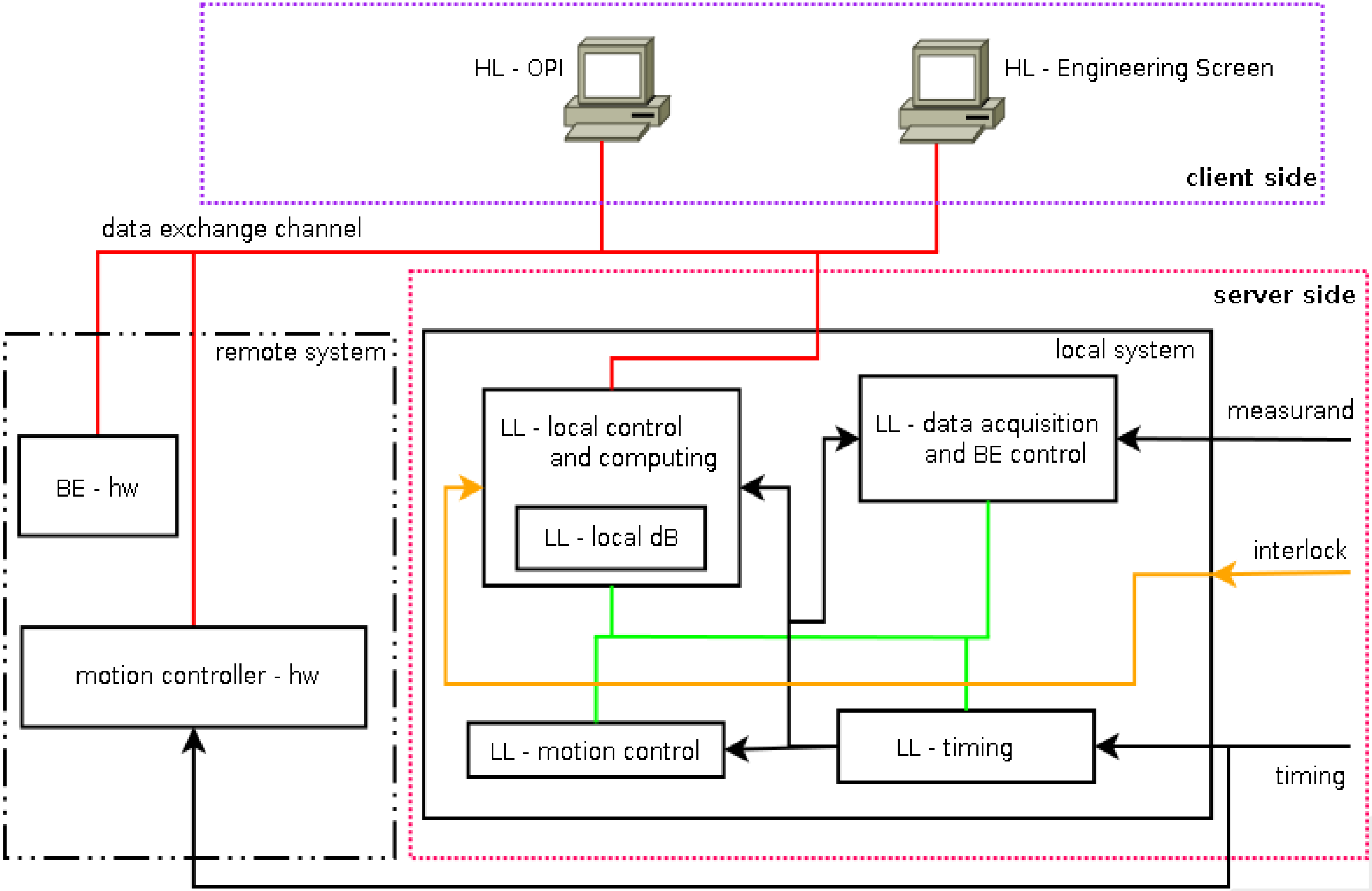
State management (normal operation, alarms, …)

Local control

HL: HMI

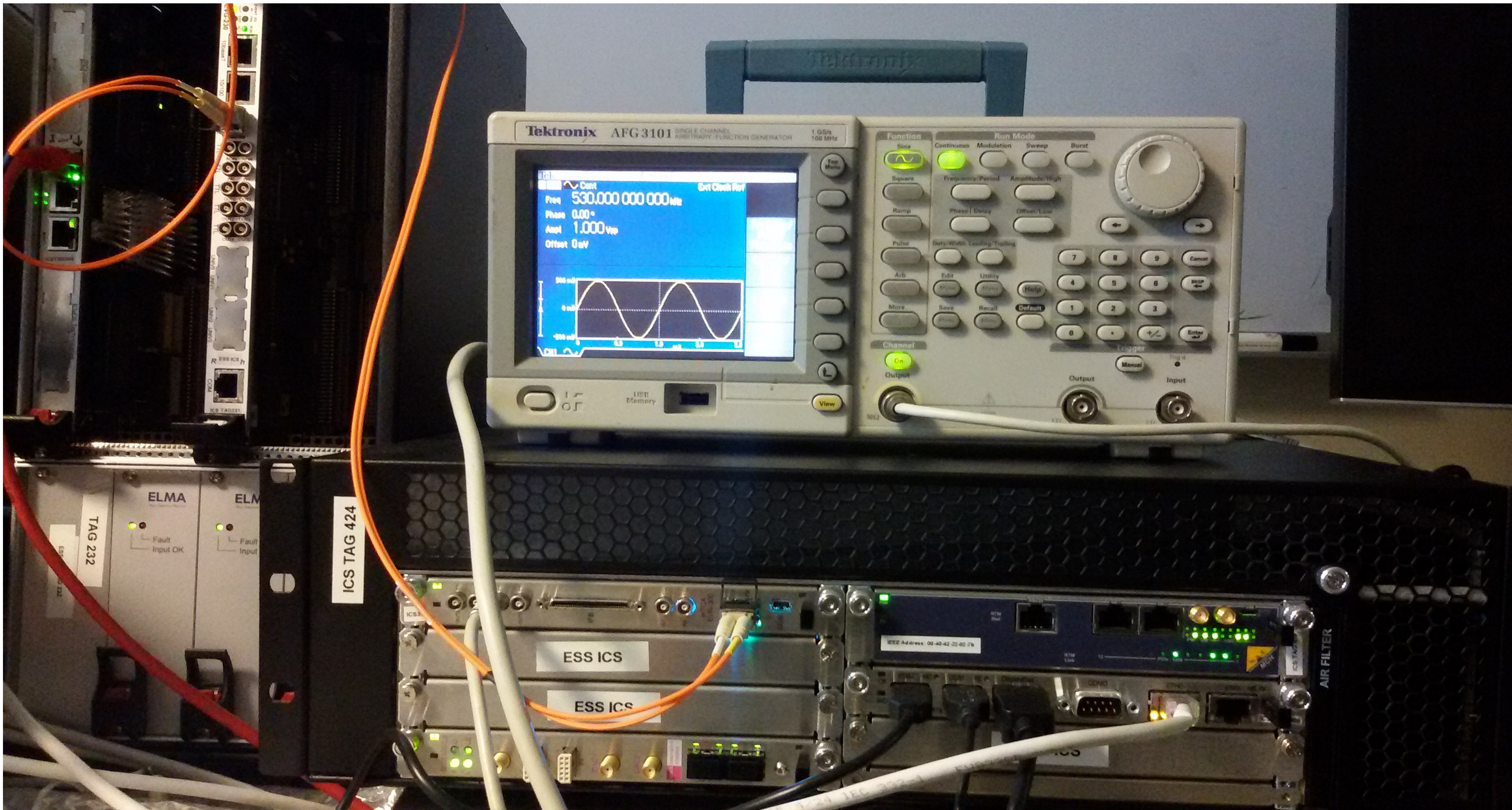
SW - 4

Putting all together: tasks/functions



HW overview

Real equipment by ESS



HW components

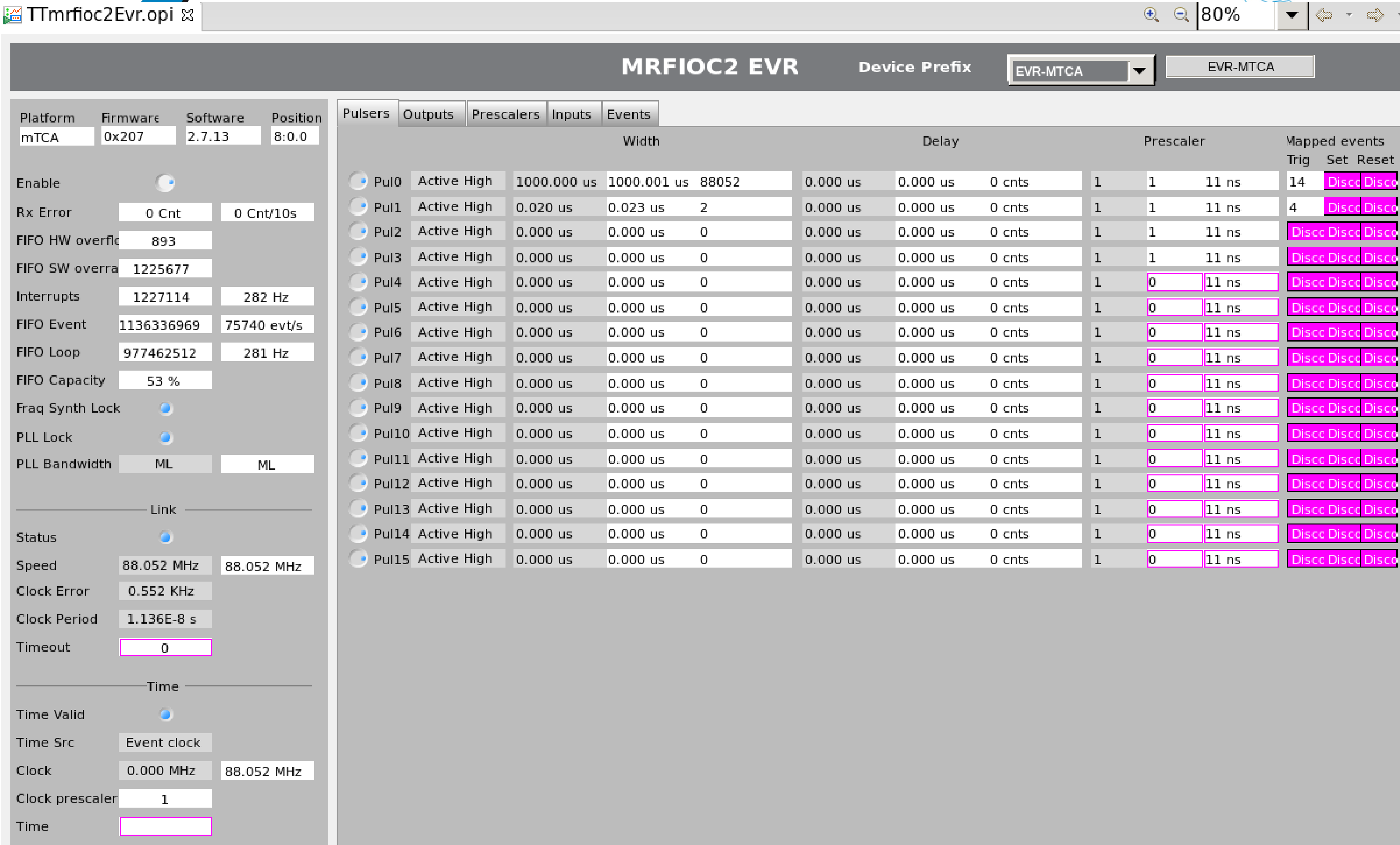
VME crate

- CPU board;
- EVG board;

MicroTCA crate

- CPU board (i7);
- EVR board;
- Struck 8300 L2 board (digitizer);

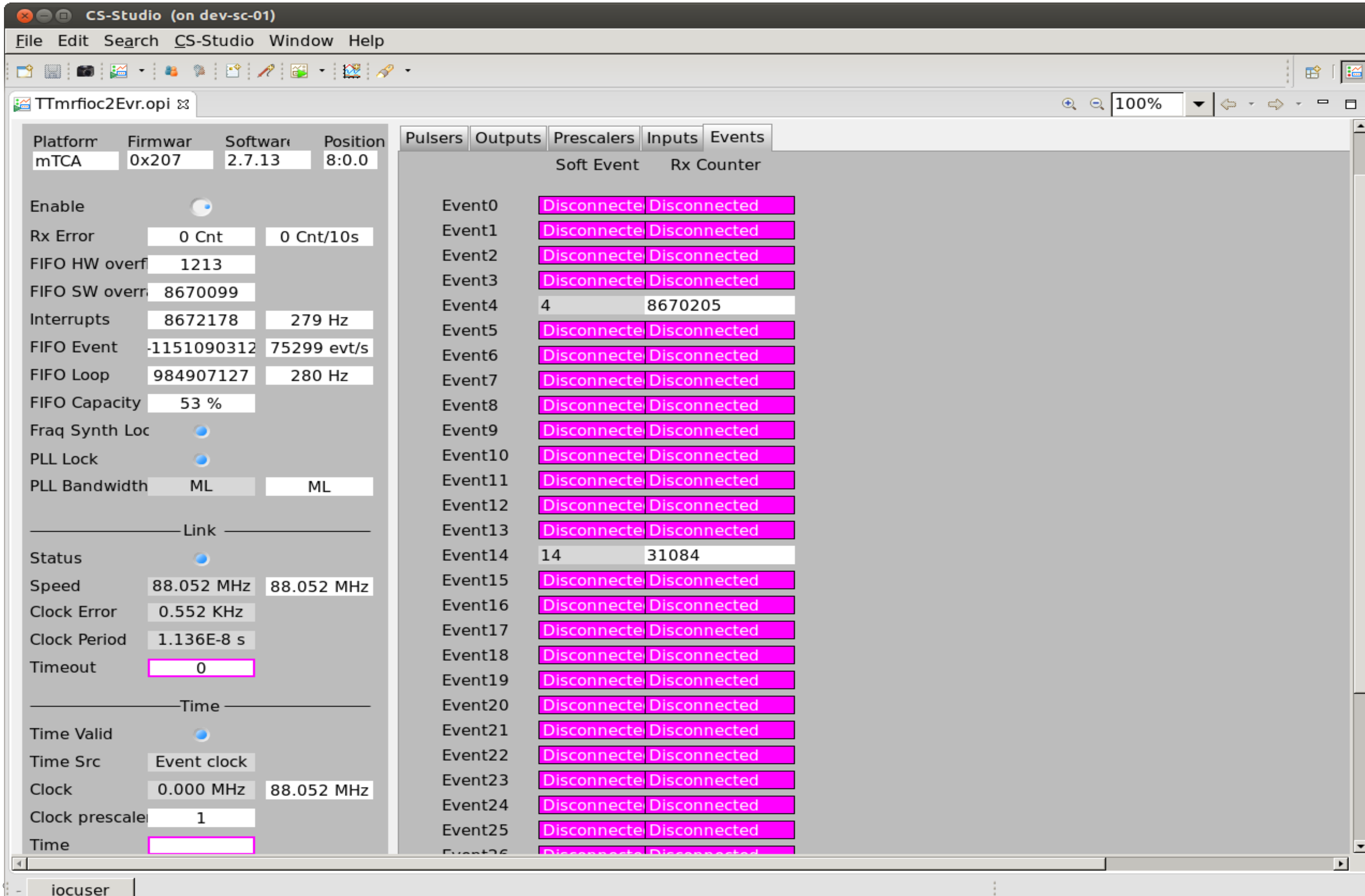
Timing OPI - 1


TTmrfioc2Evr.opi
80%
MRFIOC2 EVR
Device Prefix
EVR-MTCA
EVR-MTCA
Platform Firmware Software Position
mTCA 0x207 2.7.13 8:0.0
Enable
Rx Error 0 Cnt 0 Cnt/10s
FIFO HW overflc 893
FIFO SW overra 1225677
Interrupts 1227114 282 Hz
FIFO Event 1136336969 75740 evt/s
FIFO Loop 977462512 281 Hz
FIFO Capacity 53 %
Fraq Synth Lock
PLL Lock
PLL Bandwidth ML ML
Link
Status
Speed 88.052 MHz 88.052 MHz
Clock Error 0.552 KHz
Clock Period 1.136E-8 s
Timeout 0
Time
Time Valid
Time Src Event clock
Clock 0.000 MHz 88.052 MHz
Clock prescaler 1
Time
Pulsers Outputs Prescalers Inputs Events
Width Delay Prescaler Mapped events
Trig Set Reset
Pul0 Active High 1000.000 us 1000.001 us 88052 0.000 us 0.000 us 0 cnts 1 1 11 ns 14 Disco Disco
Pul1 Active High 0.020 us 0.023 us 2 0.000 us 0.000 us 0 cnts 1 1 11 ns 4 Disco Disco
Pul2 Active High 0.000 us 0.000 us 0 0.000 us 0.000 us 0 cnts 1 1 11 ns Disco Disco Disco
Pul3 Active High 0.000 us 0.000 us 0 0.000 us 0.000 us 0 cnts 1 1 11 ns Disco Disco Disco
Pul4 Active High 0.000 us 0.000 us 0 0.000 us 0.000 us 0 cnts 1 0 11 ns Disco Disco Disco
Pul5 Active High 0.000 us 0.000 us 0 0.000 us 0.000 us 0 cnts 1 0 11 ns Disco Disco Disco
Pul6 Active High 0.000 us 0.000 us 0 0.000 us 0.000 us 0 cnts 1 0 11 ns Disco Disco Disco
Pul7 Active High 0.000 us 0.000 us 0 0.000 us 0.000 us 0 cnts 1 0 11 ns Disco Disco Disco
Pul8 Active High 0.000 us 0.000 us 0 0.000 us 0.000 us 0 cnts 1 0 11 ns Disco Disco Disco
Pul9 Active High 0.000 us 0.000 us 0 0.000 us 0.000 us 0 cnts 1 0 11 ns Disco Disco Disco
Pul10 Active High 0.000 us 0.000 us 0 0.000 us 0.000 us 0 cnts 1 0 11 ns Disco Disco Disco
Pul11 Active High 0.000 us 0.000 us 0 0.000 us 0.000 us 0 cnts 1 0 11 ns Disco Disco Disco
Pul12 Active High 0.000 us 0.000 us 0 0.000 us 0.000 us 0 cnts 1 0 11 ns Disco Disco Disco
Pul13 Active High 0.000 us 0.000 us 0 0.000 us 0.000 us 0 cnts 1 0 11 ns Disco Disco Disco
Pul14 Active High 0.000 us 0.000 us 0 0.000 us 0.000 us 0 cnts 1 0 11 ns Disco Disco Disco
Pul15 Active High 0.000 us 0.000 us 0 0.000 us 0.000 us 0 cnts 1 0 11 ns Disco Disco Disco

Elettra Sincrotrone Trieste

Timing OPI - 2

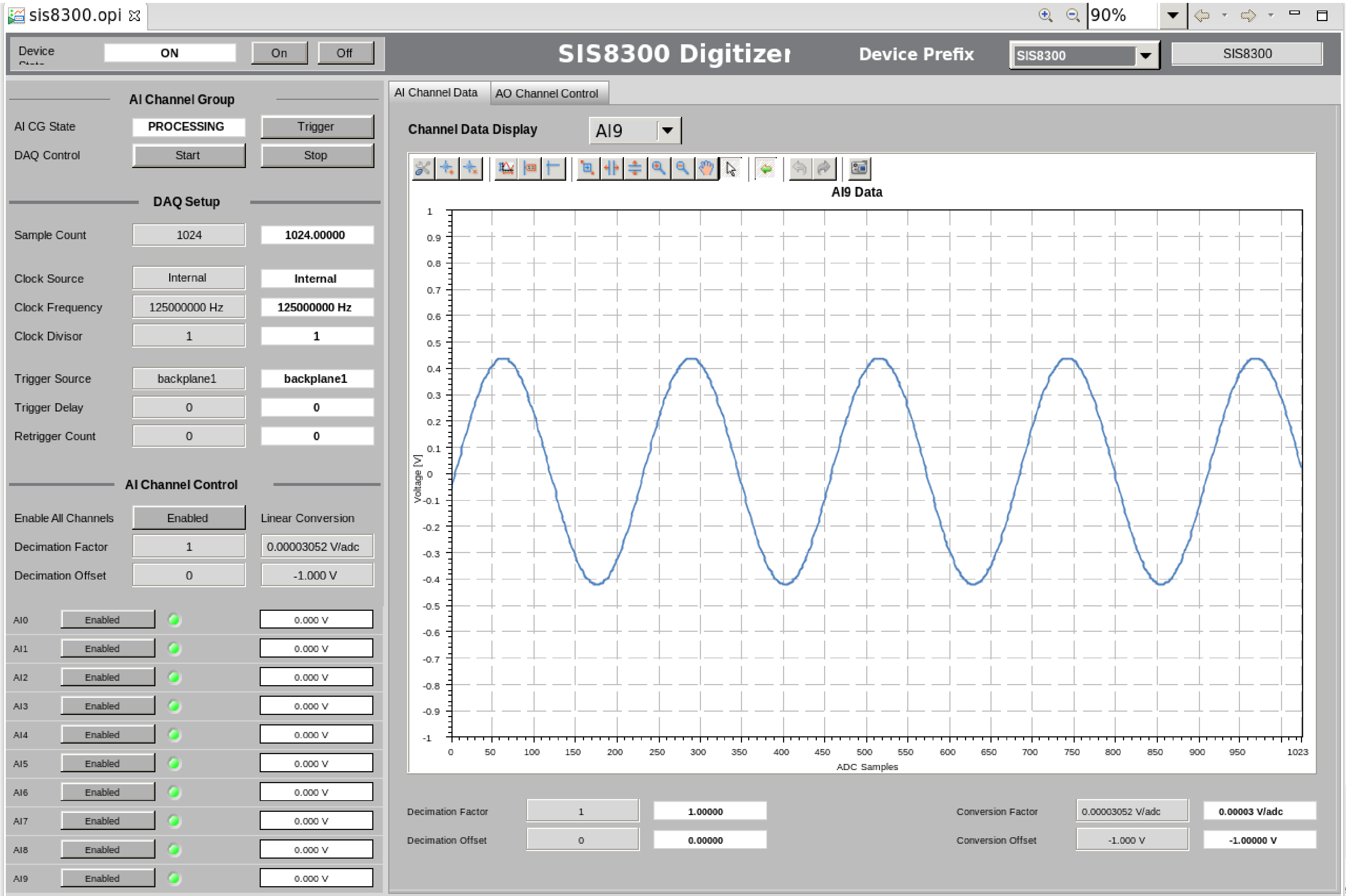
EUROPEAN SPALLATION SOURCE



Elettra Sincrotrone Trieste

SIS 8300 opi

EUROPEAN SPALLATION SOURCE



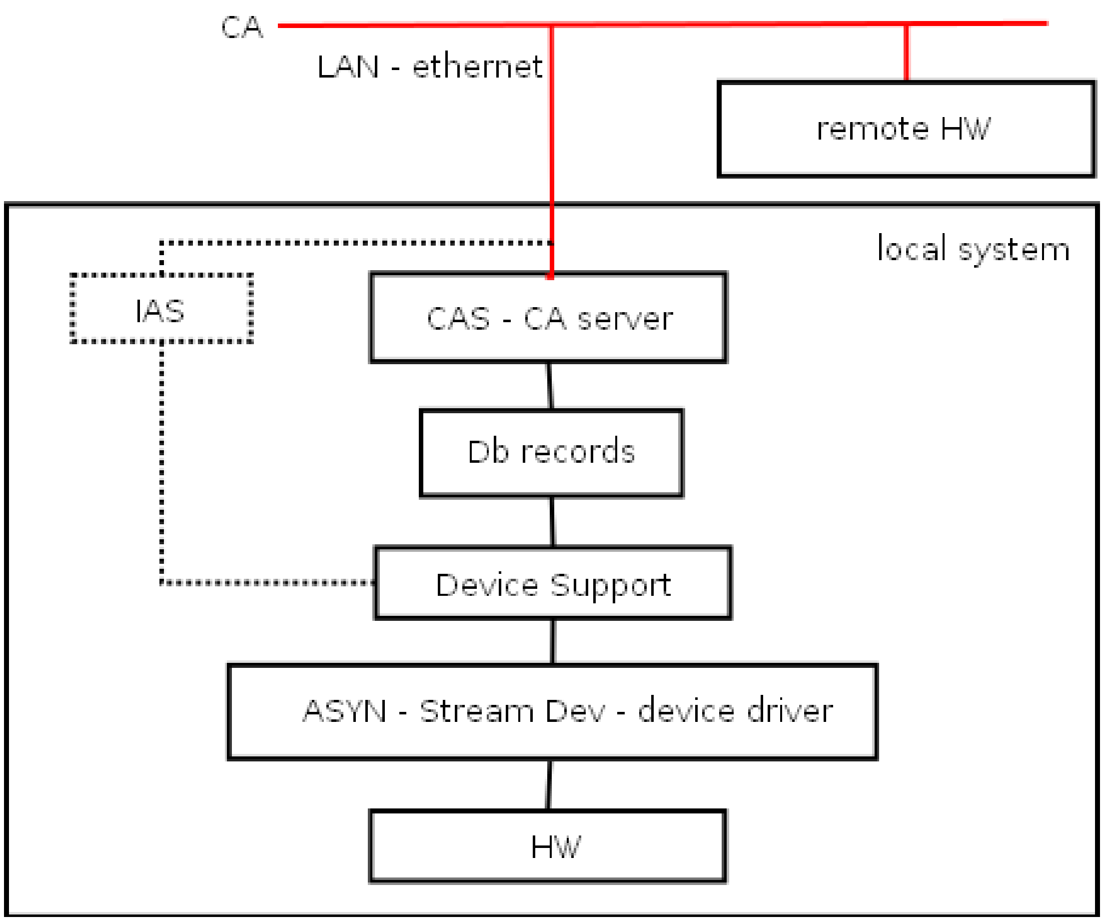
Elettra Sincrotrone Trieste

SW + EPICS - 1

EUROPEAN SPALLATION SOURCE

EPICS IOC

LL local control runs as EPICS IOC



SW + EPICS - 2

PVs - 1

1 wire generates 2 Analog Input signals (measurand) that are acquired by 2 ADCs:

- low gain wide range raw current;
- high gain small range raw current;

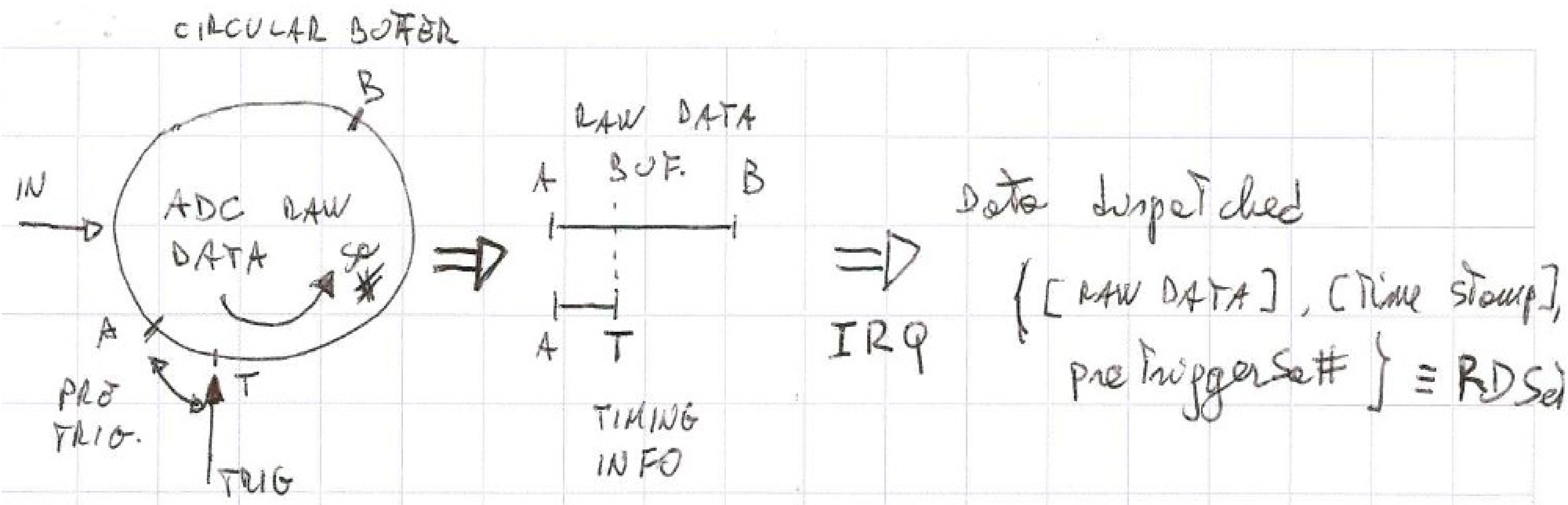
Interlock signal: digital input

Timing signal: digital input

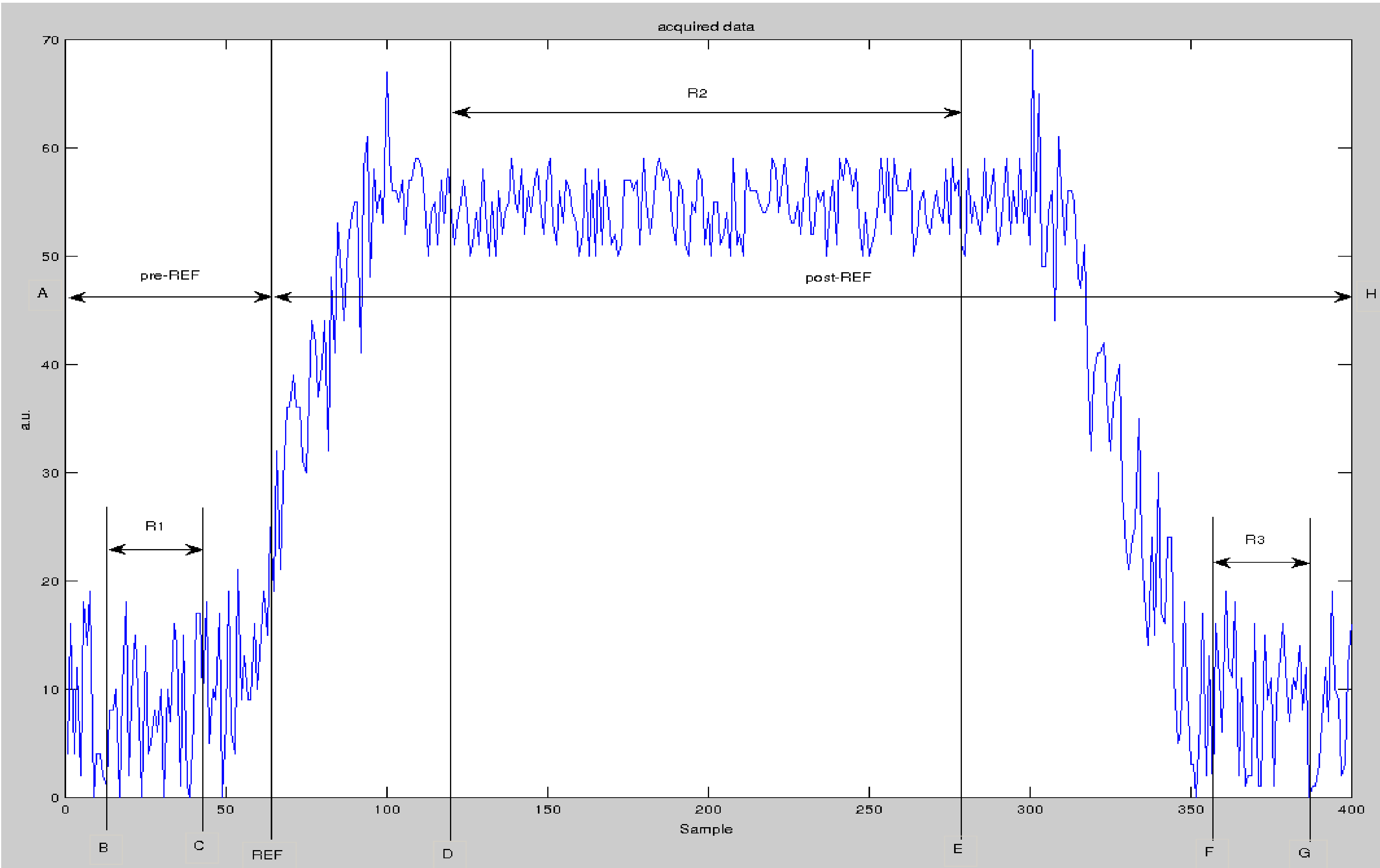
Waveform output: beam shape, obtained following one ore more assigned data processing algorithms that define the transfer function

SW + EPICS - 3

Sampling and Timestamping



SW + EPICS – 4: Pulse shape (simulated)



Elettra Sincrotrone Trieste

SW + EPICS - 5

EUROPEAN SPALLATION SOURCE

PVs - 2

Controlled remote “Analog” Outputs:

- AFE amplifiers gain (set by ethernet messages sent to BE);
- AFE HVPS set point (set by ethernet messages sent to BE);
- Wire position, 1 or 2 axis (set by ethernet messages);

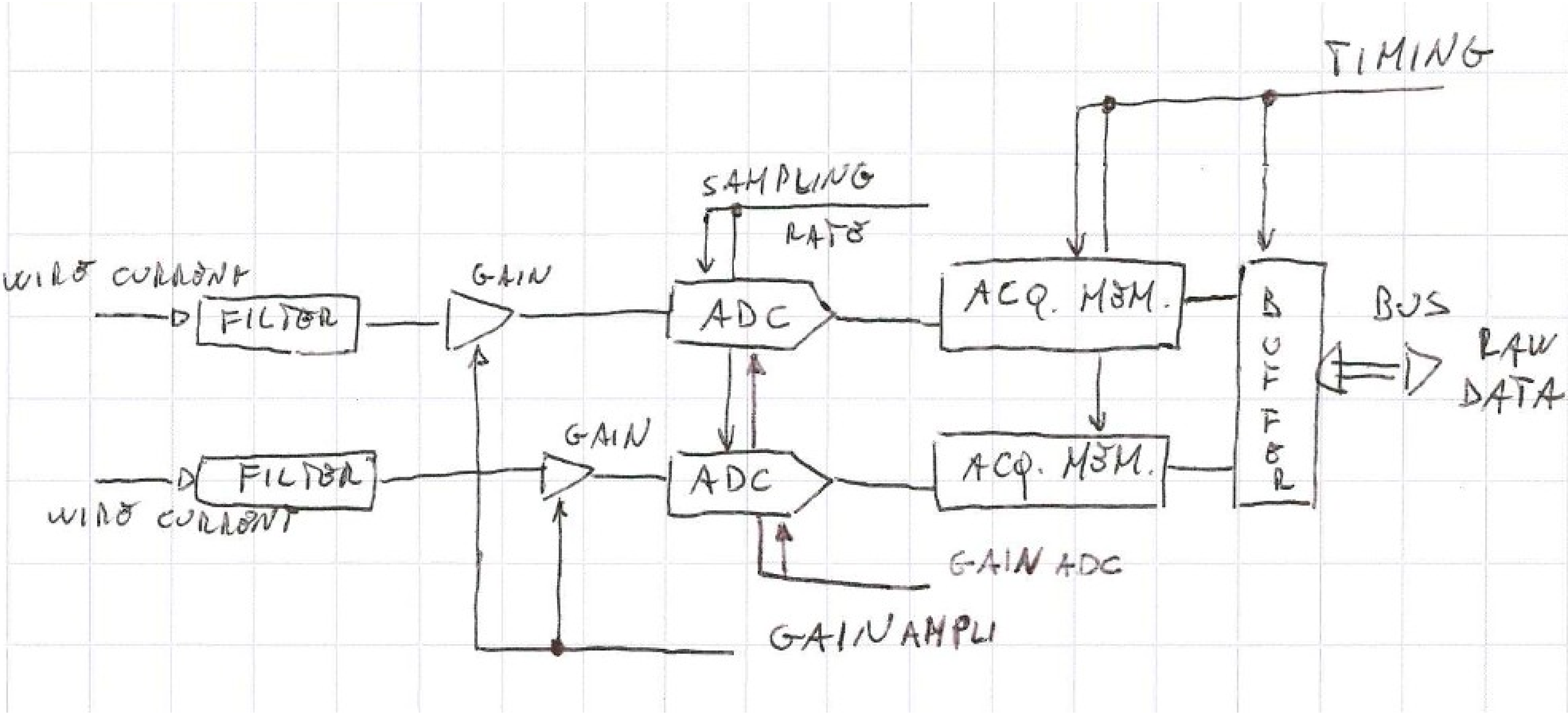
Acquired remote “Analog” Inputs:

- Wire position (encoders), 1 or 2 axis (get by ethernet messages);
- Limit Switches (get by ethernet messages);

Elettra Sincrotrone Trieste

SW + EPICS - 6

EUROPEAN SPALLATION SOURCE



BE: protocol file

```
Terminator = CR;

ReadStatus {
    out "RS?";
    in "%s";
    ExtraInput = Ignore;
}

SetBits {
    out "SB:%s";
    in "%{OK|KO}";
    ExtraInput = Ignore;
}

ReadBits {
    out "RB?";
    in "%s";
    ExtraInput = Ignore;
}

ReadVoltage {
    out "RV?";
    in "%s";
    ExtraInput = Ignore;
}

ReadTemp {
    out "RT?";
    in "%s";
    ExtraInput = Ignore;
}

SetVoltage {
    out "SV:%s";
    in "%{OK|KO}";
    ExtraInput = Ignore;
}

Abort {
    out "AB";
    in "%{OK|KO}";
    ExtraInput = Ignore;
}

Unsupported {
    out "fc";
    in "%s";
    ExtraInput = Ignore;
}

get_IDN {
    out "*IDN?";
    in "%s";
    ExtraInput = Ignore;
}
```

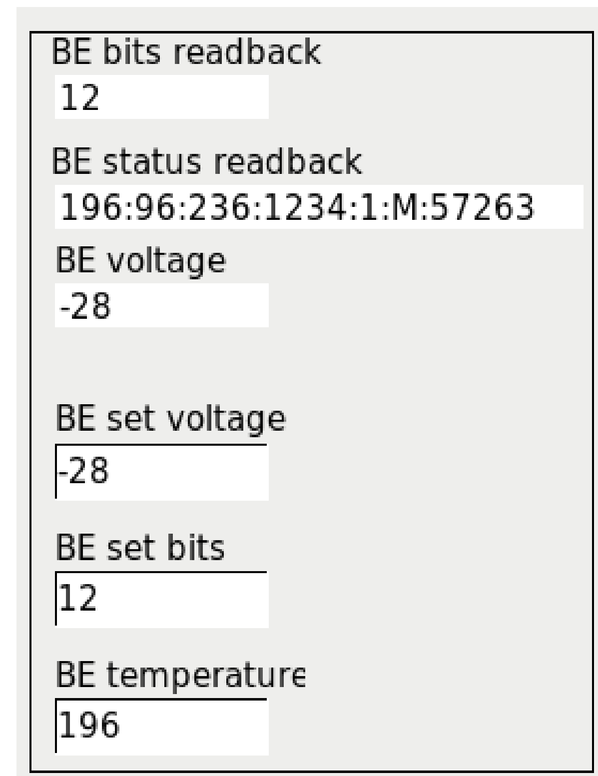
SW + EPICS - 8

BE: protocol simulation

Example: status readback
SETVOLTAGE:READVOLTAGE:BITS:SWVERSION:
SWTYPE:BESTATUS:CHECK_DIGIT

reply = str(counter) + ':' + str(counter - 100)
+ ':' + str(counter + 40) + ':' + '1234'
+ ':' + '1' + ':' + 'R' + ':' + str(counter1) + '\r'

SWTYPE : 1 = WS   5=SCINT



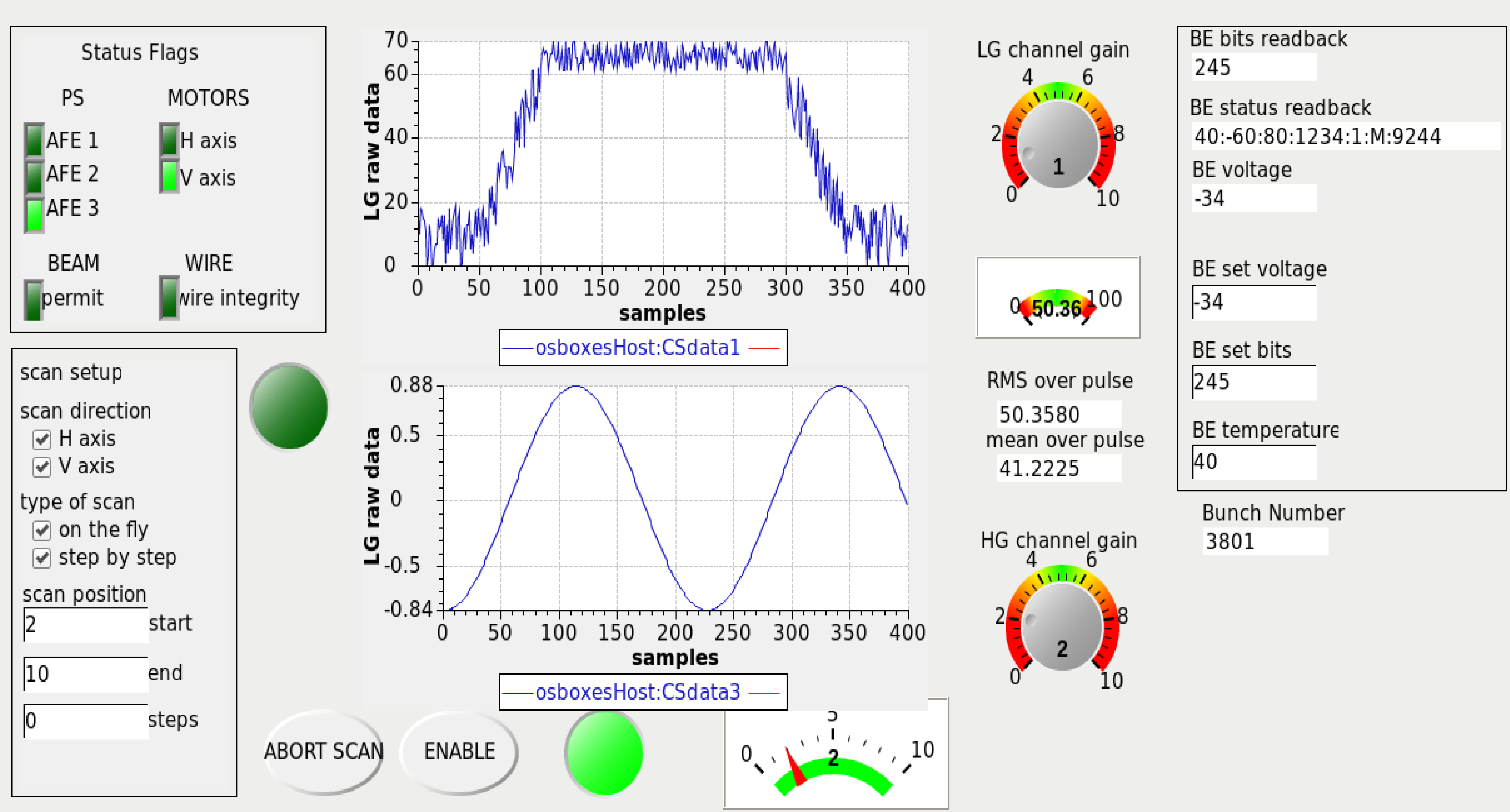
SW + EPICS - 9

PVs - 3

Several auxiliary PVs are foreseen in order to fully control the WS acquisition system:

- ADCs gain (if required/available);
- System integrity detection signals (e.g. dark current, wire integrity, ...);
- System integrity auto testing signals (simulated inputs);
- SW generated triggers and interlocks;
- ...

SW + EPICS - 10


Status Flags
PS
MOTORS
AFE 1
AFE 2
AFE 3
H axis
V axis
BEAM
WIRE
permit
wire integrity
LG raw data
samples
osboxesHost:CSdata1
osboxesHost:CSdata3
LG channel gain
RMS over pulse
50.3580
mean over pulse
41.2225
HG channel gain
50.36
BE bits readback
245
BE status readback
40:-60:80:1234:1:M:9244
BE voltage
-34
BE set voltage
-34
BE set bits
245
BE temperature
40
Bunch Number
3801
scan setup
scan direction
H axis
V axis
type of scan
on the fly
step by step
scan position
2
start
10
end
0
steps
ABORT SCAN
ENABLE

Current IOC module development status

```
.
└── ws_simul1
    ├── builddir
    │   ├── 3.14.12.5
    │   │   ├── dbd
    │   │   │   └── ws_simul1.dbd
    │   │   └── lib
    │   │       ├── centos7-x86_64
    │   │       │   ├── libws_simul1.so
    │   │       │   └── ws_simul1.dep
    │   │       └── eldk56-e500v2
    │   │           ├── libws_simul1.so
    │   │           └── ws_simul1.dep
    │   ├── 3.15.4
    │   │   ├── dbd
    │   │   │   └── ws_simul1.dbd
    │   │   └── lib
    │   │       ├── centos7-x86_64
    │   │       │   ├── libws_simul1.so
    │   │       │   └── ws_simul1.dep
    │   │       └── eldk56-e500v2
    │   │           ├── libws_simul1.so
    │   │           └── ws_simul1.dep
    │   ├── db
    │   │   ├── dbSubExample.db
    │   │   ├── dbWFM.db
    │   │   ├── userHost.db
    │   │   ├── userHost.substitutions
    │   │   ├── ws_dbfile.db
    │   │   └── ws_protocol.db
    │   ├── misc
    │   │   └── ws_protocol.proto
    │   └── opi
    │       └── engpanelBKP.opi
```

SW + EPICS - 12

Current IOC module development status

```
├── db
│   ├── dbSubExample.db
│   ├── dbSubExample.db~
│   ├── dbWFM.db
│   ├── dbWFM.db~
│   ├── userHost.substitutions
│   ├── userHost.substitutions~
│   ├── user.substitutions~
│   ├── wd_dbfile.db~
│   ├── ws_dbfile.db
│   ├── ws_dbfile.db~
│   ├── ws_protocol.db
│   └── ws_protocol.db~
├── doc
├── Makefile
├── misc
├── opi
│   └── engpanelBKP.opi
├── protocol
│   ├── ws_protocol.proto
│   └── ws_protocol.proto~
├── src
│   ├── dbSubExample.c
│   ├── dbSubExample.c~
│   ├── dbSubExample.dbd
│   ├── dbSubExample.dbd~
│   ├── initTrace.c
│   └── initTrace.dbd
├── startup
└── test
```

EPICS integration

Signal Processing

Several measurement procedures and models to be fit to the acquired data are foreseen in order to compute the beam profile;

Data acquisition auto switching between channels (HG vs LG);

Local Protection System

Its primary function should be to avoid damages generated by user actions: several details have to be defined

Scanning Logic and Motion Control

Several details have to be defined

Conclusions

The global picture describing the SW is now clear enough to develop the first release (NON Real-Time) of the controlling code of the WS and it's OPI.

Thanks to the availability of the real HW the porting of the original simulated code/IOC has been ported to reality but…

… a lot of details have to be still investigated, depending on:

- HW issues (e.g. motion controller driving, real time approach to the buffer acquisition and storage, …);
- Low level drivers issues (motion controller and its EPICS demo ioc);
- OPI and Engineering screen definitions;
- Code porting to Real Time operation issues;
- ...