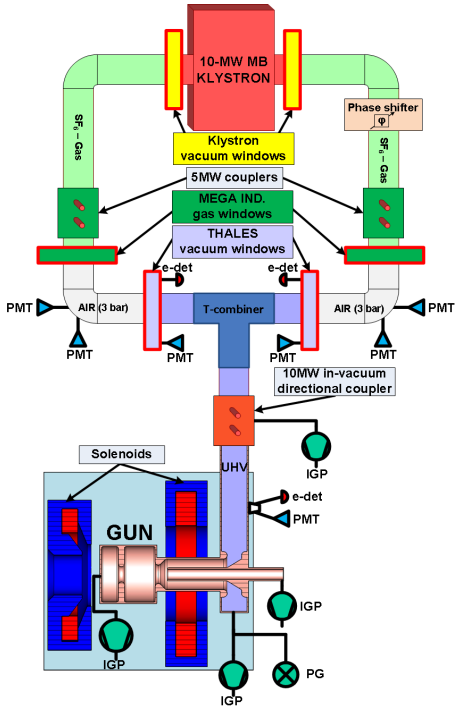
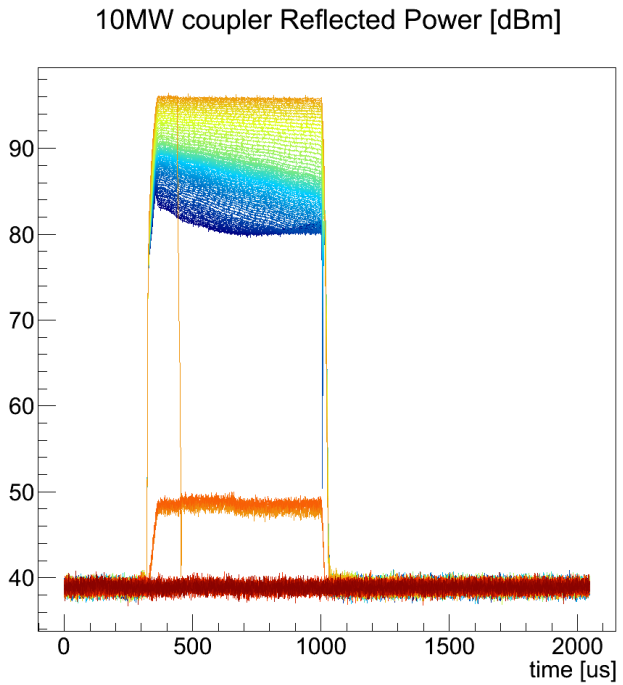
GUN 4.2 INTERLOCK DATABASE.





Yves Renier

Gun 4.2 Interlock database

PITZ Physics Seminar, 22nd of January 2016

HELMHOLTZ ASSOCIATION

DESY

Outline

Get the Data

Classes' Description

Analysis in Detail

GUI for summary of all ILs

Conclusion

1 Get the Data

2 C++ Classes

3 Analysis

4 GUI

5 Conclusion

Table of Contents

Get the Data

Classes' Description

Analysis in Detail

GUI for summary of all ILs

Conclusion

1 Get the Data

2 C++ Classes

3 Analysis

4 GUI

5 Conclusion

Interlock date and time

1 Get the Data
2 C++ Classes
3 Analysis
4 GUI
5 Conclusion

Procedure

- Get IL flags every second from DAQ.
- Every transition from 0 to 1 of ”Sum Gun IL” or ”Preamp enable” considered as new IL.
- Save date and time of all ILs.

Software used

- *daqpr* to get IL flags every second.
- *analyze_ILs.m* to find ILs.
- *plot_ILs.m* to save the ILs date and time.

Pros and Cons

Pros

- Simple.
- Robust (all ILs are found) as long as IL flags are available in DAQ.

Cons

- Max. period length per *daqpr* run: $\simeq$ 2 weeks.
- *daqpr* usually crashes when DAQ data is missing.
- all files must be manually modified, then combined.
- $\simeq$1-2 days to gather data for 1 year run.
- Includes switch off of RF2 by operators.

1 Get the Data
2 C++ Classes
3 Analysis
4 GUI
5 Conclusion

Interlock data Gathering

For each IL, get data within a 10s window using *daqbr*.

Pros

- Existing software. Automatic script developped.
- Within a *daqbr* run, properties are synchronized.

Cons

- If 1 property is missing, no data is gathered at all.
- Non-synchronous prop. collected separatly: 3 *daqbr* runs per ILs.
- Then merging needed.
- Very long: $\simeq$2 weeks to gather data for 1 year run.

1 Get the Data
2 C++ Classes
3 Analysis
4 GUI
5 Conclusion

List of DAQ Properties

1 Get the Data
2 C++ Classes
3 Analysis
4 GUI
5 Conclusion

adc0 data

Scalar data:

- `"RfinfRF2CavityPower_20131024"`
- `"RF2___LLRF_CONTROLLER__FALL_TIME_20141122"`
- `"RF2__FLATTOP_LENGTH_EFFECTIVE_20141204"`
- `"RF2___LLRF_CONTROLLER__RISE_TIME_20141122"`
- `"GUN__IGP1_P" & "GUN__IGP2_P"`

Power from couplers:

- `"RF2Cpl10MWFW & "RF2Cpl10MWRE"`
- `"RF2WG1CavityFW" & "RF2WG2CavityFW"`
- `"RF2WG1CavityRE" & "RF2WG2CavityRE"`

PMTs:

- `"GUN__COUPLER__PMT_20140905"`
- `"GUN__WG1__THALES_PMT_VAC_20140905" & "GUN__WG2__THALES_PMT_VAC_20140905"`
- `"GUN__WG1__THALES_PMT_AIR_20140905" & "GUN__WG2__THALES_PMT_AIR_20140905"`
- `"GUN__WG1__RF_WINDOW_PMT_AIR_20140905" & "GUN__WG2__RF_WINDOW_PMT_AIR_20140905"`

Electron detectors:

- `"GUN__COUPLER__E_DET_20140905"`
- `"GUN__WG1__THALES_E_DET_VAC_20140905" & "GUN__WG2__THALES_E_DET_VAC_20140905"`

List of DAQ Properties

adc1 data

- "GUN__WATER__TF392"
- "GUN__MAGNETS__MAIN__ON"
- "GUN__MAGNETS__MAIN__POLARITY"
- "GUN__MAGNETS__MAIN__RB"
- "GUN__MAGNETS__MAIN__SP"
- "GUN__MAGNETS__BUCKING__ON"
- "GUN__MAGNETS__BUCKING__RB"
- "GUN__MAGNETS__BUCKING__SP"

IL flags data

- "GUN__ILOCK__DP1__SG1"
- "GUN__ILOCK__DP1__SG2"
- "I_LOCK__KLYS2__SDO_PL_96_119"
- "I_LOCK__KLYS2__SDO_PL_120_143"

1 Get the Data
2 C++ Classes
3 Analysis
4 GUI
5 Conclusion

Table of Contents

Get the Data

Classes' Description

Analysis in Detail

GUI for summary of all ILs

Conclusion

1 Get the Data

2 C++ Classes

3 Analysis

4 GUI

5 Conclusion

C++ data classes

1 Get the Data
2 C++ Classes
3 Analysis
4 GUI
5 Conclusion

scalar_data

This class describes the evolution of a scalar property and allows to:

- Load data from a ROOT file
- Find the index of the min and max values
- Plot (normal and log scale)
- Copy

C++ data classes

1 Get the Data
2 C++ Classes
3 Analysis
4 GUI
5 Conclusion

ILflag_data

This class describes the evolution of a IL flag property and allows to:

- Load data from a ROOT file
- Tell if the IL was triggered
- Plot

C++ data classes

1 Get the Data
2 C++ Classes
3 Analysis
4 GUI
5 Conclusion

spectra_data

This class describes the evolution of a spectral property and allows to:

- Load from a ROOT file or from arrays
- Find the index/channel of the max value
- Create spectral data from the max of each index
- Plot
- Copy/Addition/Substraction

C++ data classes

intelock_data

This class contains all data about an IL and allows to:

- Load from a ROOT file
- Find the 1^{st} and last channels of the RF flattop
- Find the 1^{st} entry and channel of the 1^{st} event
- Find the number of channels with ongoing event
- Find the 1^{st} entry/ the timestamp/ the date/ the time when an IL occurred
- Plot of the main IL flags
- Determine if the gun was lost.
- Determine if the event occured in the gun or at the RF windows or if it was a klystron IL
- Print a summary of the IL ...

1 Get the Data
2 C++ Classes
3 Analysis
4 GUI
5 Conclusion

C++ data classes

intelock_data members

```
char filename [255]
ILflag_data PreAmp_Enable, RF2Spark
ILflag_data Max_refl_WG1, Max_refl_WG2
ILflag_data gun_IGP1, gun_IGP2
ILflag_data PMT_coupler
ILflag_data PMT_TH_window_wg1_vac, PMT_TH_window_wg2_vac
ILflag_data PMT_TH_window_wg1_air, PMT_TH_window_wg2_air
ILflag_data PMT_RFgas_window_wg1_air, PMT_RFgas_window_wg2_air
ILflag_data e_det_coupler
ILflag_data e_det_TH_window_wg1_vac, e_det_TH_window_wg2_vac
ILflag_data temp_TH_window_wg1, temp_TH_window_wg2
ILflag_data IR_TH_window_wg1_air, IR_TH_window_wg2_air
ILflag_data Sum_gun_IL
scalar_data Pgun, temp_gun
scalar_data IGP1, IGP2
scalar_data rise_time, pulse_length, fall_time
scalar_data main_sol_status, main_sol_pol, main_sol_rdbk, main_sol_SP
scalar_data buck_sol_status, buck_sol_rdbk, buck_sol_SP,
spectra_data tenmwcoupl_fw, tenmwcoupl_refl
spectra_data fivemwcoupl_wg1_fw, fivemwcoupl_wg2_fw
spectra_data fivemwcoupl_wg1_refl, fivemwcoupl_wg2_refl
spectra_data pmt_guncoupl
spectra_data pmt_thal_wg1_vac, pmt_thal_wg2_vac
spectra_data pmt_thal_wg1_air, pmt_thal_wg2_air
spectra_data pmt_rf_wg1_air, pmt_rf_wg2_air
spectra_data edect_guncoupl
spectra_data edect_thales_wg1_vac, edect_thales_wg2_vac
```

1 Get the Data
2 C++ Classes
3 Analysis
4 GUI
5 Conclusion

Usage

1 Get the Data
2 C++ Classes
3 Analysis
4 GUI
5 Conclusion

- All these classes are compatible with ROOT (as a shared library).
- To use them in ROOT:

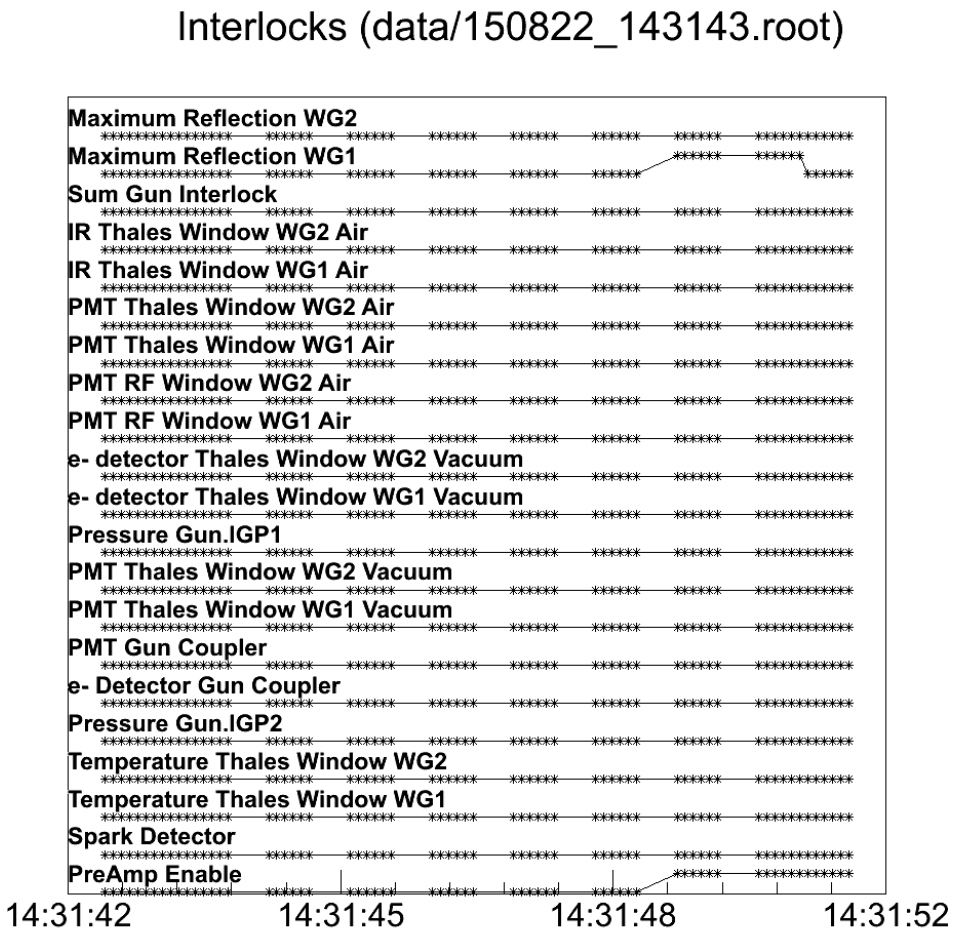
```
root [0] gSystem->Load("libclasses_data.so");
```

- That's it ... Let's play !

```
root [1] interlock_data test("data/150822_143143.root");
root [2] test.Plot_ILs();
root [3] test.tenmwcoupl_refl.Plot();
root [4] test.tenmwcoupl_refl.make_scalar_data_max().Plot();
root [5] test.Pgun.Plot();
root [6] Int_t min_channel=test.first_channel_flattop("10MW")+10;
root [7] Int_t max_channel=test.last_channel_flattop("10MW")-10;
root [8] spectra_data ratio_refl =test.tenmwcoupl_refl-test.tenmwcoupl_fw;
root [9] ratio_refl.make_scalar_data_max(min_channel,max_channel).Plot();
root [10] test.lost_gun();
(const bool)1
```

Example 1 - IL Flags

```
root [1] interlock_data test("data/150822_143143.root");
root [2] test.Plot_ILs();
```



1 Get the Data

2 C++ Classes

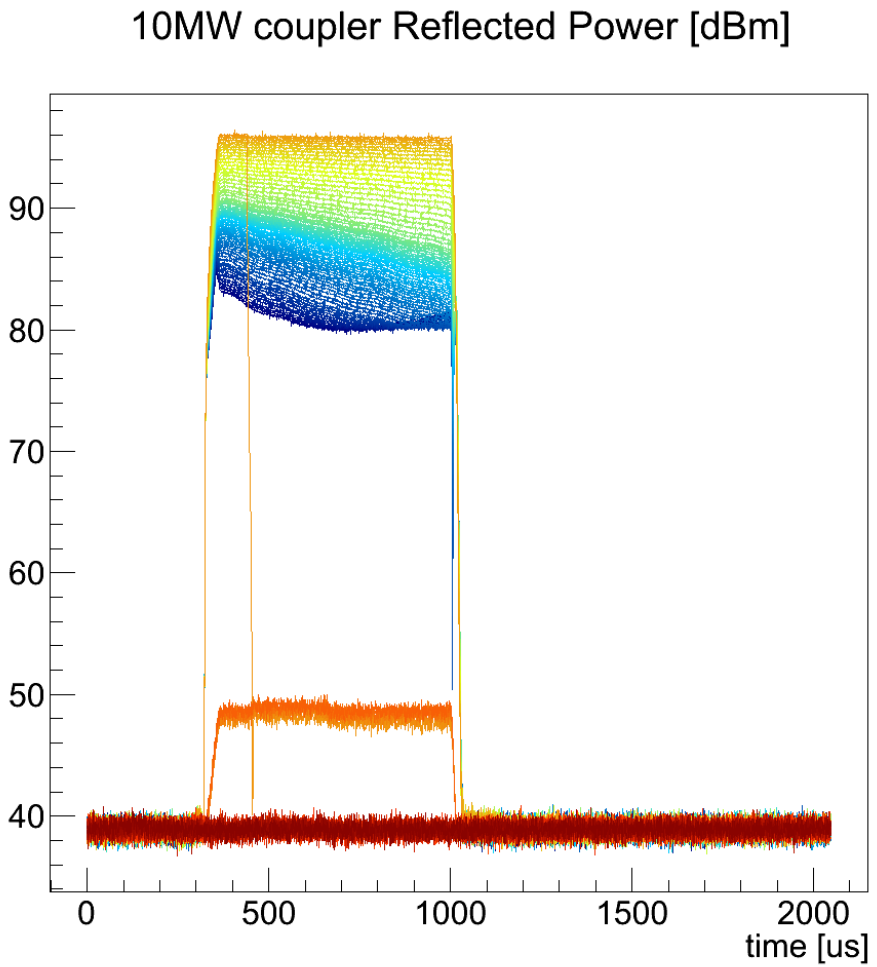
3 Analysis

4 GUI

5 Conclusion

Example 2 - Reflected Power

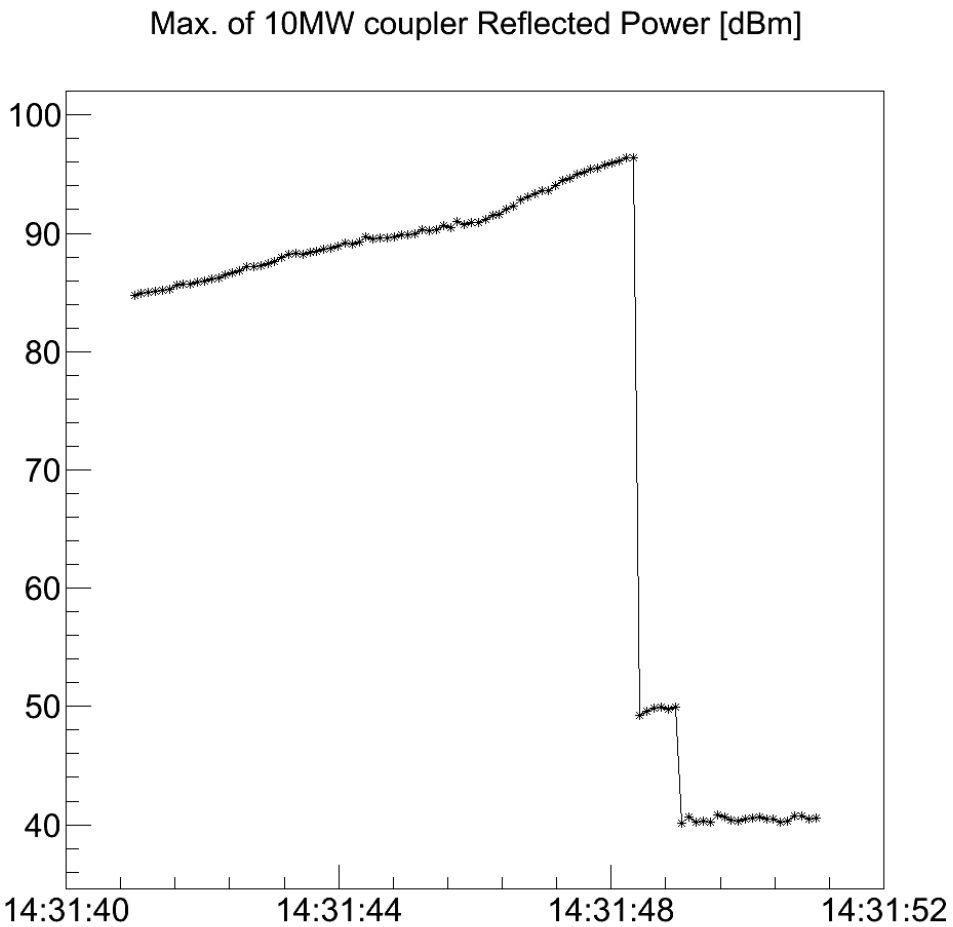
```
root [3] test.tenmwcoupl_refl.Plot();  //blue to red: first to last spectra.
```



1 Get the Data
2 C++ Classes
3 Analysis
4 GUI
5 Conclusion

Example 3 - Max(Reflected Power)

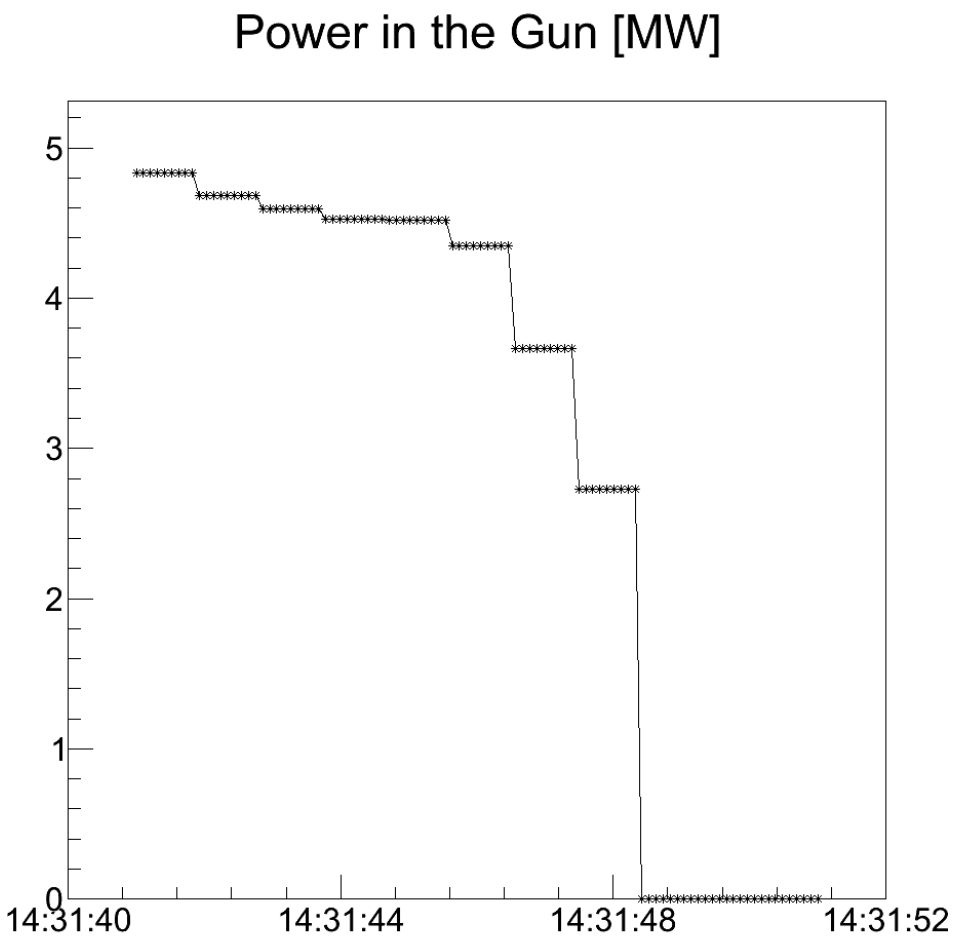
```
root [4] test.tenmwcoupl_refl.make_scalar_data_max().Plot();
```



1 Get the Data
2 C++ Classes
3 Analysis
4 GUI
5 Conclusion

Example 4 - Pgun

```
root [5] test.Pgun.Plot();
```



1 Get the Data
2 C++ Classes
3 Analysis
4 GUI
5 Conclusion

Documentation

Extensive PDF/HTML documentation available.

time_t interlock_data::interlock_timestamp (void) const

Function to get the timestamp of the first entry with an IL.

Returns:
the timestamp of the first entry with an IL, if no IL was triggered, returns the timestamp of the first spike in reflected power.

See also:
entry_IL()

bool interlock_data::klystron_IL (void) const

Function to tell if there was a klystron IL.

return true if PreAmp_Enable was triggered() and Sum_gun_IL was not triggered

Returns:
if true if there was a klystron IL.

See also:
ILflag_data::triggered()

Int_t interlock_data::last_channel_flattop (const char * coupler = "10MW") const

Function to give the last channel of the flattop RF pulse.

Parameters:
coupler which coupler to get the info from ("10WM", "5MWwg1" or "5MWwg2")

Returns:
the last channel of the flattop RF pulse, -1 if not found

Int_t interlock_data::last_entry_fw_power (const char * coupler = "10MW") const

Function to give the last entry with forward power above the threshold.

Parameters:
coupler which coupler to get the info from ("10WM", "5MWwg1" or "5MWwg2")

Returns:
the last entry with forward power above the threshold, -1 if not found

bool interlock_data::lost_gun (void) const

Function to tell if the IL was caused by a temperature drift (lost gun).

This function looks at the ratio of the reflected power over the forward power. If it increases above the threshold before the first event, it returns true.

Returns:
if true the IL was caused by a temperature drift, if false it was not.

Int_t interlock_data::nchannel_with_event (const char * coupler = "10MW") const

Function to give the number of channels durring events (spike in reflected power).

1 Get the Data

2 C++ Classes

3 Analysis

4 GUI

5 Conclusion

Difficulties

1 Get the Data
2 C++ Classes
3 Analysis
4 GUI
5 Conclusion

- Properties changed during the run (f.e. FPGA $\rightarrow \mu$TCA)
- Time rounded to second in DAQ.
- Problem getting Gun temperature and Main solenoid (not synchronous ?)
- Some properties are oversampled in DAQ.

Table of Contents

Get the Data

Classes' Description

Analysis in Detail

GUI for summary of all ILs

Conclusion

1 Get the Data

2 C++ Classes

3 Analysis

4 GUI

5 Conclusion

bool interlock_data::lost_gun()

It tells if the IL was caused by a temperature drift.

1 Get the Data
2 C++ Classes
3 Analysis
4 GUI
5 Conclusion

bool interlock_data::lost_gun()

It tells if the IL was caused by a temperature drift.
This function looks at the maximum of the ratio of the reflected power over the forward power durring the flattop.
If it increases above the threshold before the first event, it returns true.

1 Get the Data
2 C++ Classes
3 Analysis
4 GUI
5 Conclusion

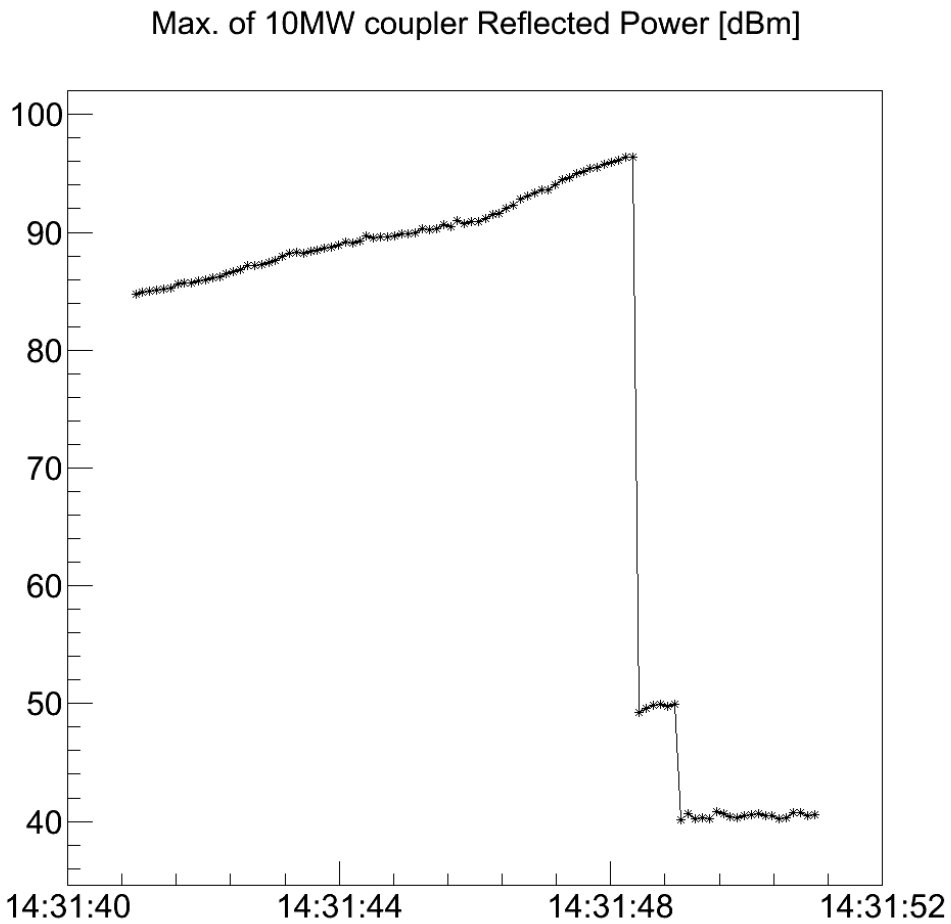
bool interlock_data::lost_gun()

It tells if the IL was caused by a temperature drift.
This function looks at the maximum of the ratio of the reflected power over the forward power durring the flattop.
If it increases above the threshold before the first event, it returns true.

1 Get the Data
2 C++ Classes
3 Analysis
4 GUI
5 Conclusion

lost_gun() 0/4

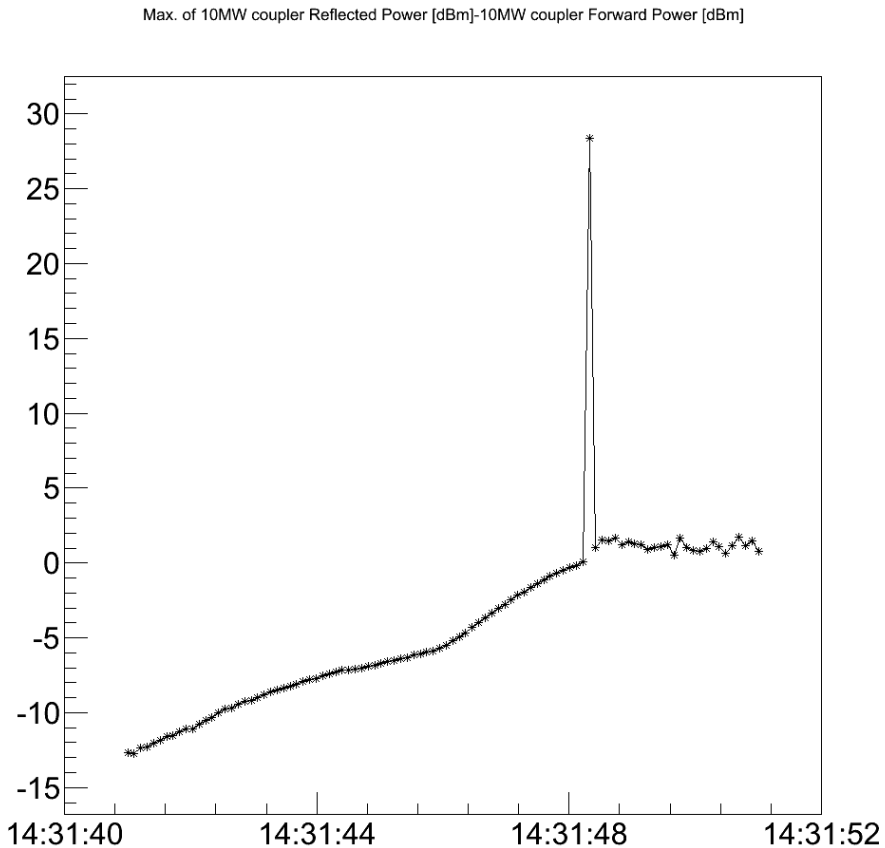
```
root [6] test.tenmwcoupl_refl.make_scalar_data_max().Plot();
```



1 Get the Data
2 C++ Classes
3 Analysis
4 GUI
5 Conclusion

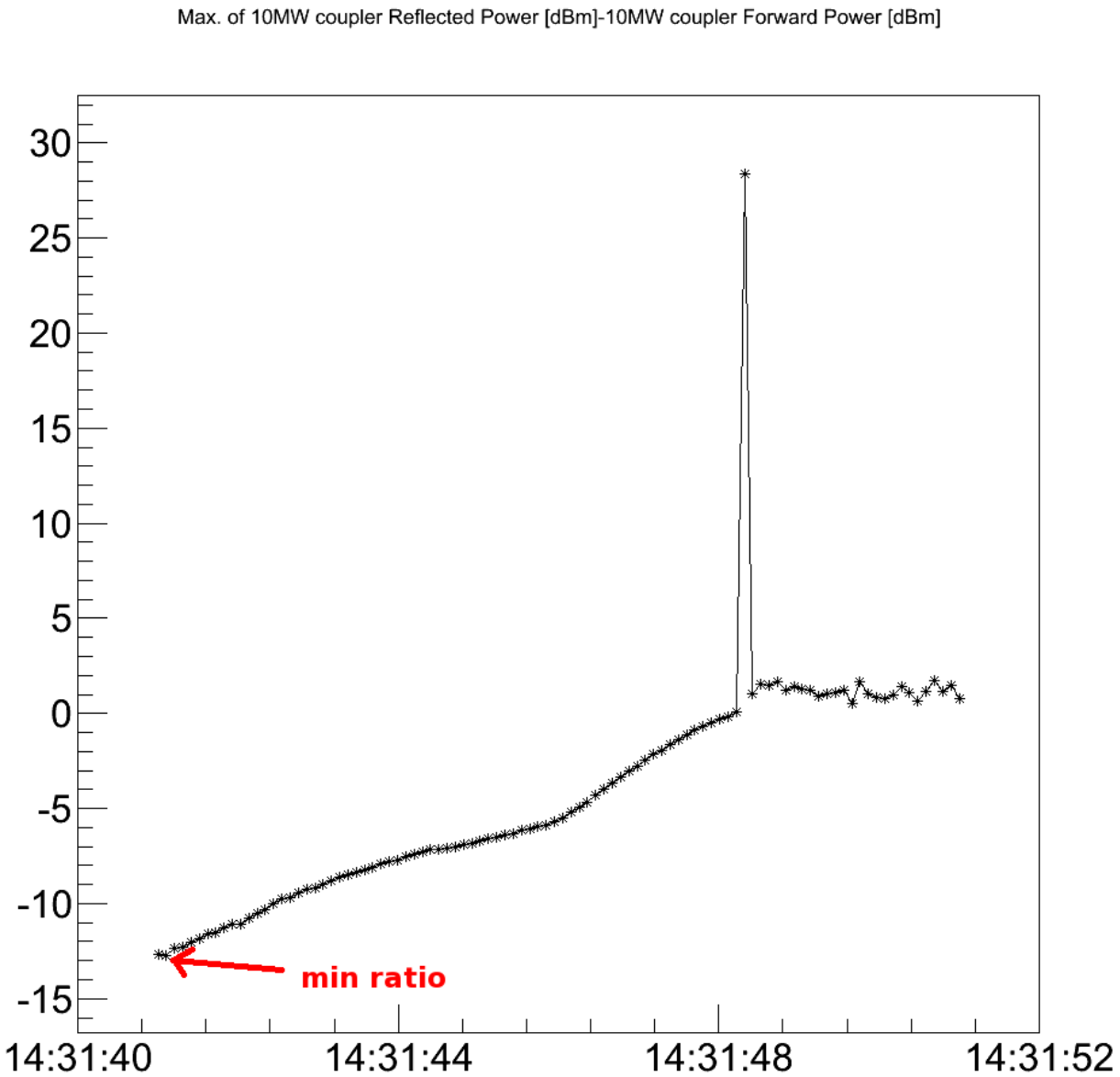
lost_gun() 1/4

```
root [7] Int_t min_channel=test.first_channel_flattop()+10;
root [8] Int_t max_channel=test.last_channel_flattop()-10;
root [9] spectra_data ratio_refl =test.tenmwcoupl_refl-test.tenmwcoupl_fw;
root [10] ratio_refl.make_scalar_data_max(min_channel,max_channel).Plot();
```



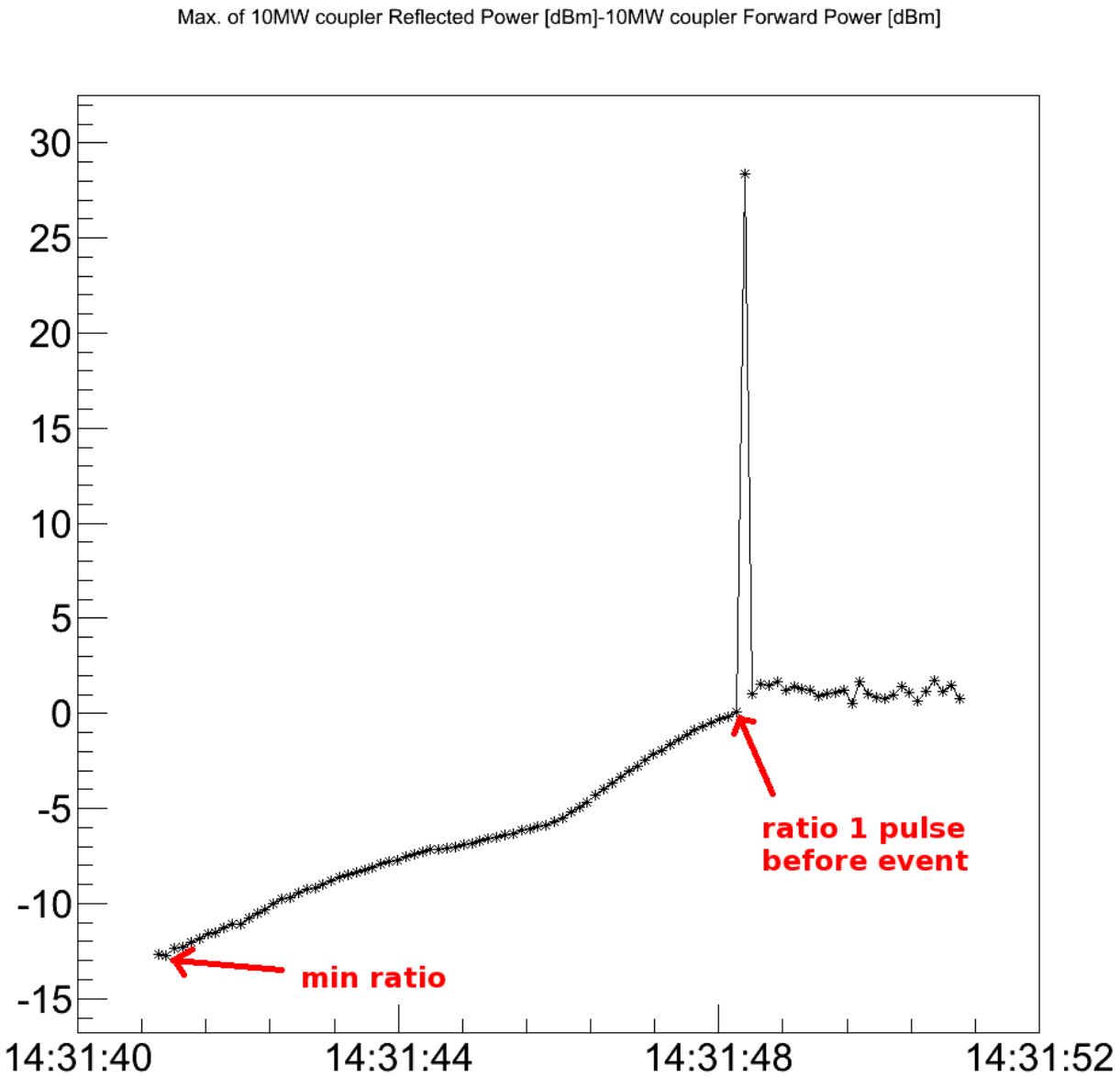
1 Get the Data
2 C++ Classes
3 Analysis
4 GUI
5 Conclusion

lost_gun() 2/4



1 Get the Data
2 C++ Classes
3 Analysis
4 GUI
5 Conclusion

lost_gun() 3/4



1 Get the Data

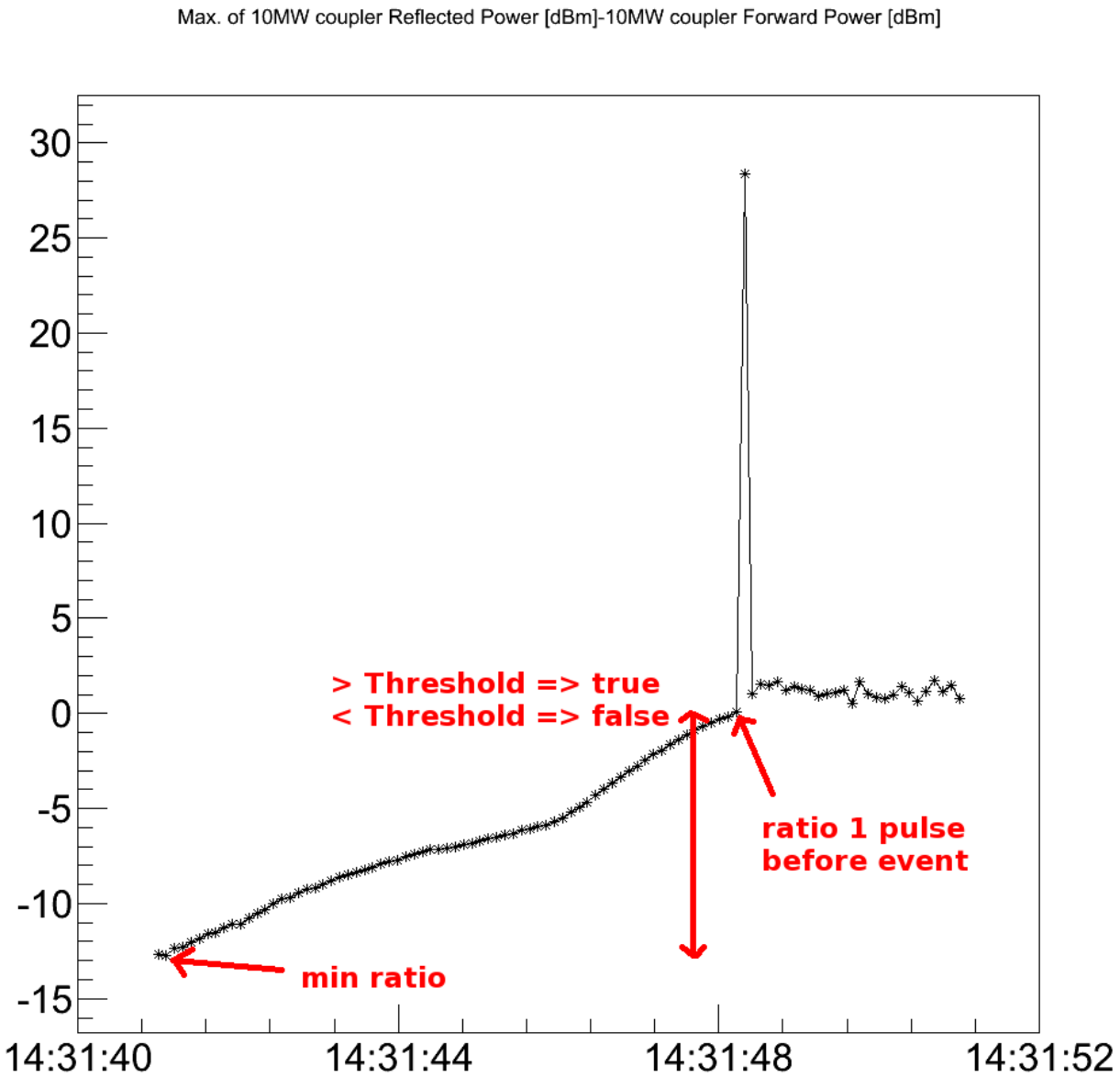
2 C++ Classes

3 Analysis

4 GUI

5 Conclusion

lost_gun() 4/4



1 Get the Data

2 C++ Classes

3 Analysis

4 GUI

5 Conclusion

bool interlock_data::gun_event()

Function to tell if the first event came from the gun. This function looks at the entry of the first event in the 10MW and 5MW couplers:

- if the gun was lost, return false
- if no event was detected in the 10MW coupler it returns false
- if there was an event in detected in the 10MW coupler but not in the 5MW coupler, it ouputs a warning and returns true
- if it happened before in the 5MW couplers it returns false
- if it happend before in the 10MW coupler it output a warning and returns true.
- if it happened at the same time in the 10MW and 5MW couplers it returns true

1 Get the Data
2 C++ Classes
3 Analysis
4 GUI
5 Conclusion

bool interlock_data::window_event()

Function to tell if the first event came from the windows.
This function looks at the entry of the first event in the 10MW and 5MW couplers:

- if the gun was lost, it returns false
- if no event detected at any of the couplers, it returns false
- otherwise it returns the oposite of gun_event()

1 Get the Data
2 C++ Classes
3 Analysis
4 GUI
5 Conclusion

bool interloc_data::klystron_IL()

Function to tell if there was a klystron IL.
Returns true if PreAmp Enable IL was triggered and
Sum gun IL was not triggered

1 Get the Data
2 C++ Classes
3 Analysis
4 GUI
5 Conclusion

Table of Contents

Get the Data

Classes' Description

Analysis in Detail

GUI for summary of all ILs

Conclusion

1 Get the Data

2 C++ Classes

3 Analysis

4 GUI

5 Conclusion

ILTable

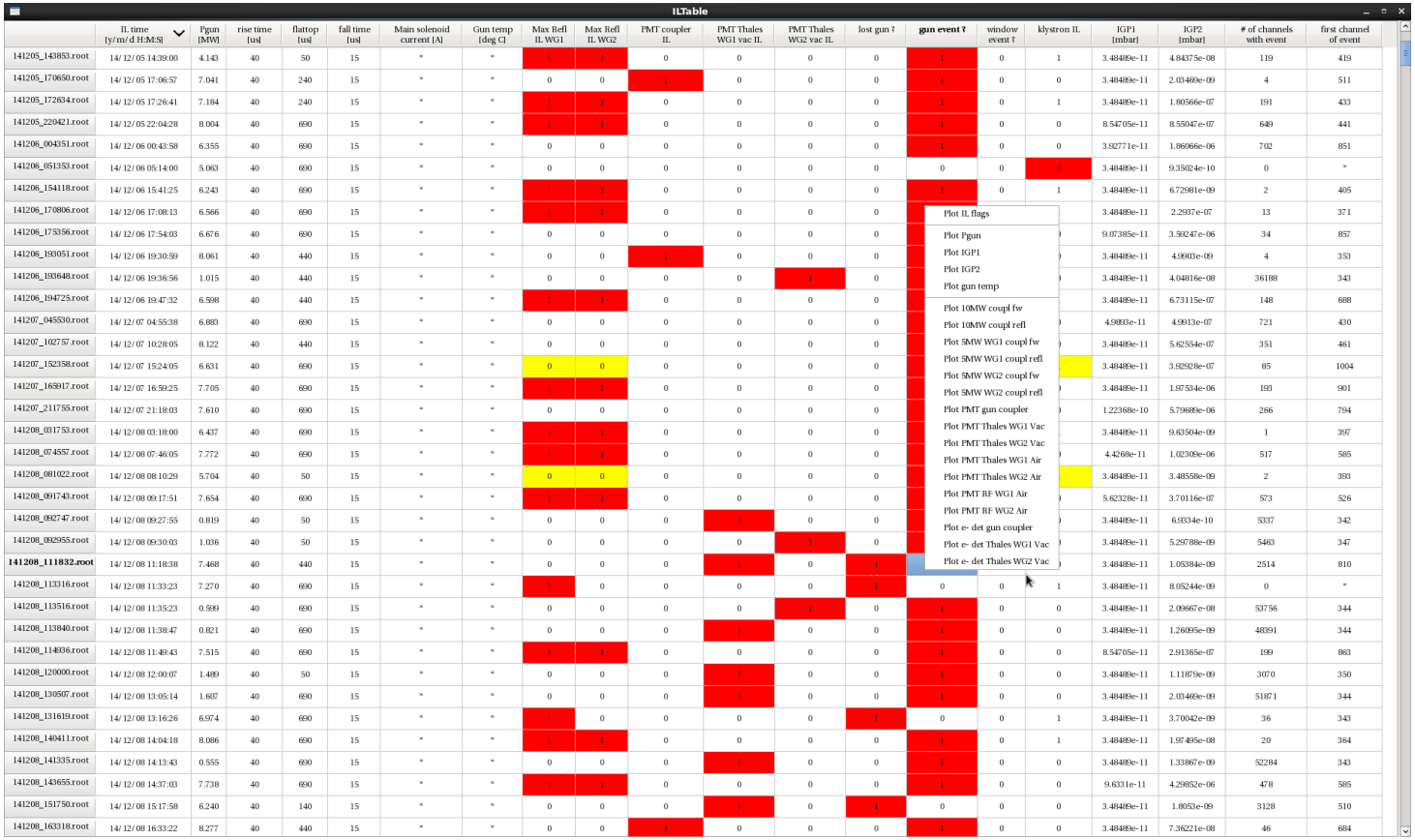
Graphical User Interface (GUI) to present the resutls of the analysis of all ILs since 14/12/2014

Functionalities:

- On first launch, automatically annalyse all available ILs ($<$ 1h).
- Then load the saved summary (almost instantly).
- ROOT filename as header of the line, for futher analysis with ROOT.
- Sorting per columns
- Double click on a cell show the associated plot to the column for that IL.
- Right click on a cell to open a menu showing all available plots.

1 Get the Data
2 C++ Classes
3 Analysis
4 GUI
5 Conclusion

ILTable demo



red: cause of IL

yellow: posible cause of IL

green: RF2 stop by user

1 Get the Data

2 C++ Classes

3 Analysis

4 GUI

5 Conclusion

Table of Contents

Get the Data

Classes' Description

Analysis in Detail

GUI for summary of all ILs

Conclusion

1 Get the Data

2 C++ Classes

3 Analysis

4 GUI

5 Conclusion

Conclusion

1 Get the Data
2 C++ Classes
3 Analysis
4 GUI
5 Conclusion

- IL database implemented.
- Almost fully automatic gathering of all ILs since 14/12/2014.
- Automatic analysis of ILs (in ROOT).
- GUI develloped to see the result of the analysis.
- Available in SVN (`https://svnsrv.desy.de/desy/PITZ/doocs_measure_scripts/ROOT/ILs_database`).
- Full documentation of all classes (`./doc/html/index.html`).

Prospects

1 Get the Data
2 C++ Classes
3 Analysis
4 GUI
5 Conclusion

- migrating from personal location to `/doocs/measure` (ongoing).
- Direct gathering of the data (to get rid of *daqbr* related issues) ???
- Update for the new gun (new properties, ...).
- Server to detect new IL and trigger automatic gathering of data and analysis.
- Interface for the shift crew to validate/correct the analysis.