User's Guide

[November 2012]

This document is distributed under a Creative Common license cc-by-nc-nd 3.0

This document describes how to configure **R**OOT **H**istogram **B**uilder (**RHB**) run options and how to declare basic parameters and histograms.

These operations are directly handmade by editing some RHB configuration files.

This guide is divided in three parts:

- the first one gives informations to set RHB environment configuration in **.RHBrc** resource file
- then you will find examples to declare RHB basic parameters and histograms in a configuration file (**facqConf**)
- and at least a further description parameter input description (**pid**) file is given when using RHB with a FASTER DAQ.

LPC Caen
6 Boulevard Maréchal Juin
14050 – CAEN Cedex

Tel: +33 (0)2 31 45 25 00

Table of Contents

I – RHB environment configuration

The **.RHBrc** resources file is used to configure RHB options at startup. These options are data read type and source, file names for parameters and histograms declarations, file name for visualization, etc...

I – 1. General options

I – 1. a. Parameters and histograms declaration file

```
ConfigFile: MyDeclarationsFile.facqConf
```

When such a file is specified, declared parameters and histograms are loaded when RHB is launched.

I – 1. b. Automatic histograms saving at end of run

```
SaveHistos: true
```

Set this option to false to not automatically save histograms at end of run.

I – 1. c. Basic histograms visualization

If a C visualization file name is given, it will be loaded with the "-v" or "--visu" option

```
DisplayFile: MyDisplayMacroFile.C
```

Note that an another way to visualize histograms is to use RHV (ROOT Histogram Visualization) option, as described in next I – 4. chapter.

I – 2. Data format reading

I – 2. a. FASTER DAQ reading

- From **file**:

```
DataReader: FFileFasterDataReader
FFasterDataReader.ParamDescFile: RawParametersDescriptionFile.pid
Device: FFileDevice
FFileDevice.FileName: /path/of/file/FileName.fast
FFileDevice.BufferSize: 8192
```

- From **network**:

```
DataReader: FIPFasterDataReader
FFasterDataReader.ParamDescFile: RawParametersDescriptionFile.pid
Device: FIPDevice
FIPDevice.Protocol: UDP
FIPDevice.Server: $FASTER_SRC
FIPDevice.Port: $RHB_UDP_LISTEN_PORT
```

FASTER_SRC and *RHB_UDP_LISTEN_PORT* environment variables should have been previously defined in the .bashrc user file for example.

I – 2. b. GANDALF DAQ reading (GUINEVERE)

- From **file**:

  In the same manner than with FASTER,

```
DataReader: FGuinevereDataReader
# On x86 PC (Little Endian): dat files need to swap bytes in data written by Big
# Endian CPUs
# BUT only for dat files, NO SWAP for gdlf files (built by Ganoffline)
FBlockDataReader.SwapData: true
Device: FFileDevice
FFileDevice.FileName: /path/of/file/FileName.dat
FFileDevice.BufferSize: 8192
```

- From **online written file**:

```
DataReader: FGuinevereDataReader
# On x86 PC (Little Endian): dat files need to swap bytes in data written by Big
# Endian CPUs
# BUT only for dat files, NO SWAP for gdlf files (built by Ganoffline)
FBlockDataReader.SwapData: true
Device: FWrittenFile
# Timeout to try reading some new data written in file online
# By default: 10s
FFileDevice.Timeout: 35.0
FFileDevice.FileName: /path/of/exported_file/FileName.dat
FFileDevice.BufferSize: 8192
```

I – 2. c. ROOT TTree reading

```
DataReader: FTTreeReader
FTTreeReader.TreeName: DataTree
FTTreeReader.FileName: /path/of/first/file/FileName.root
FTTreeReader.ParamsFromBranches: true
```

I – 2. d. ROOT TChain reading

```
DataReader: FTChainReader
FTTreeReader.TreeName: DataTree
FTTreeReader.FileName: /path/of/first/file/file1.root
FTTreeReader.FileName: /path/of/second/file/file2.root
FTTreeReader.FileName: /path/of/third/file/file3.root
FTTreeReader.FileName: /path/of/all/files/in/directory/*.root
FTTreeReader.FileName: /path/of/selected/files/in/directory/My??Wild*Card*.root
FTTreeReader.ParamsFromBranches: true
```

I – 3. GUI options

I – 3. a. Enable periodic canvas refresh

```
FAcqGUI.CanSetRefreshTime: true
```

I – 3. b. Canvas refresh time setting

Periodic time refresh value is expressed in seconds.

```
FAcqGUI.CanvasRefreshTime: 2.0
```

I – 3. c. Simple histograms display (ROOT TBrowser)

```
FAcqGUI.OpenVisuEnv: true
```

If the DisplayFile item has been specified as explained in I – 1. c. chapter, the C macro will be executed and displayed beside the ROOT TBrowser window.

I – 4. RHV options

I – 4. a. Enable pages editor

```
RHV.WithEditor: true
```

When allowed, user can manage its own display by defining its pages, booklets and layouts via a dedicated "**Editor**" tab.

Set this option to false not to display RHV pages editor when one configuration is stable for example.

I – 4. b. Enable oscillograms display

```
RHV.WithOscillograms: true
```

When allowed, all known oscillograms are displayed in an additional "**OSC**" tab.

I – 4. c. RHV configuration file

```
RHV.ConfigFile: MyRHVConfigFile.rhvConf
```

Define inner RHB file used to load and save RHV environment.

I – 5. User class definitions

I – 5. a. User Event Processor

```
UserEventProcessor: MyEventProcessor
```

User can define its own Event Processor to treat data. This must be a **subclass** of FEventProcessorThread in terms of C++ language. Furthermore, the source files for the class (MyEventProcessor.C and MyEventProcessor.h here), must be in the directory from which RHB is launched.

I – 5. b. User Data Reader

```
DataReader: MyDataReader
```

This line replaces one of those shown in I – 2. chapter.

User can defined its own data format. This must be a **subclass** of FDataReader in terms of C++ language meaning all pure virtual methods have to be coded. As with User Event Processor, the source files for the class (MyDataReader.C and MyDataReader.h here), must be in the directory from which RHB is launched too.

II – RHB parameters and histograms declarations

All parameters and histograms declarations are saved in a RHB configuration file ended with "**.facqConf**" extension. At launch, if the *ConfigFile* option is specified in the .RHBrc file as described in I – 1. chapter, all these definitions will be loaded in RHB.

Some samples are given here, giving how-to examples to define particular RHB parameters and histograms types, depending on your needs.

As some of them cannot be simply declared with RHB GUI, you will need to edit and write yours in your facqConf file following these examples.

II – 1. Parameters declarations

II – 1. a. Calculated parameter (FParamCalc)

Here's how to declare a calculated parameter to perform conversions or to control parameters:

```
//=========================================================================
//
// Calculated parameter declaration example
//
//=========================================================================
// This configuration file shows how to declare a "calculated" parameter.
// This kind of parameter allows to define a parameter which is a mathematical formula of already
// existing parameters. Variable coefficients may be added if needed. In that case, the coefficient
// have to be indicated as digits into brackets, and their initial values (values of coefficients when
// RHB is launched) have to be specified. See the ROOT TFormula documentation to know what
// are the possible formulas.
// WARNING: this parameter is active and its value is computed only if all the parameters in the
// formula are active simultaneously. If one of the parameters is  inactive (value not set), the
// "calculated" parameter is inactive.
// The following examples assume that the raw parameters are "Param1", "Param2" and "Param3".
//
//-------------------------------------------------
FParamCalc;Energy1;[0]+Param1*[1]
[0];10
[1];2.5
//-------------------------------------------------
FParamCalc;ParCal1;(Param1*[0])+([1]*Param2)
[0];1
[1];2.5
//-------------------------------------------------
FParamCalc;DiffParam12;Param1-Param2
//-------------------------------------------------
FParamCalc;ComplexExpression;([0]*sin(Param2*[1]+[2]))+[3]
[0];40
[1];0.14
[2];-0.456
[3];80
//-------------------------------------------------
// A control parameter can also be defined by using calculated parameters
FParamCalc;CtrlParam;[0]
[0];1
```

II – 1. b. Condition (FCondition)

Here's how to declare a condition:

```
//============================================================
//
// Condition declaration example
//
//============================================================
// This configuration file shows how to declare a "condition" parameter.
// This kind of parameter allows to define a parameter which is a boolean formula of already
// existing parameters. Variable coefficients may be added if needed. In that case, the coefficient
// have to be indicated as digits into brackets, and their initial values (values of coefficients when
//  RHB is launched) have to be specified. See the ROOT TFormula documentation to know
// what are the possible formulas. If a parameter name only is given, the return value is "true"
// when  the parameter is activated, i.e. when the parameter is present in the event.
// The following examples assume that the raw parameters are "Param1", "Param2" and "Param3".
//
//------------------------------------------------
FCondition;CondParam1;(Param1>[0])&&(Param1<[1])
[0];48.5
[1];55
//------------------------------------------------
FCondition;CondParam1ANDParam2;Param1 && Param2
//------------------------------------------------
FCondition;CondParam1Bis;CondParam1||(abs(Param1-[0])<[1])
[0];70
[1];2
//------------------------------------------------
FCondition;CondParam3;Param3<[0] || Param3 > [1]
[0];6
[1];27.5
//------------------------------------------------
FCondition;MixedCondition;CondParam1 && CondParam3
//------------------------------------------------
```

II – 1. c. Graphical cut (FGCondition)

Here's how to declare a graphical cut:

```
//=================================================================
//
// Graphical cut declaration example
//
//=================================================================
// This configuration file shows how to declare a "graphical condition" parameter.
// This kind of parameter allows to define a parameter which is true if the values of
// already existing parameters are in a contour. Calculated parameters can be used in
// graphical conditions. Graphical cut conditions can be used in conditions.
// The following examples assume that the raw parameters are "Param1", "Param2"
// and "Param3".
//-------------------------------------------------
// A Graphical Condition parameter named "CondGraph" based of parameters "Param1"
// and  "Param2" is defined. A graphical cut (TCutG) named "SuperCut" is generated.
// The coordinates of the 11 points are specified after the parameters names.
//
FGCondition;CondGraph;SuperCut
Param2;Param1
11
0;51.2786;73.3398
1;71.7369;69.1773
2;39.7709;51.6353
3;79.835;50.1487
4;66.1961;25.7683
5;51.6105;43.4466
6;31.3646;34.8154
7;39.647;46.3237
8;25.3828;61.0288
9;45.6287;60.7092
10;51.2786;73.3398
//-------------------------------------------------
FCondition;CondParam1;(abs(Param1-[0])<[1])
[0];48
[1];2
//-------------------------------------------------
FCondition;MixedCondition;CondParam1 && CondGraph
//-------------------------------------------------
```

II – 1. d. Time parameter (FParamTime)

Here's how to declare a time parameter:

```
//================================================================
//
// Time parameter declaration example
//
//================================================================
// This configuration file shows how to declare a "time" parameter.
// This parameter contains the actual machine time (nanosec precision) in seconds.
// Its value is evaluated once per event. It can be used afterwards in other parameters definitions.
//-----------------------------------------------
FParamTime;MyTime;My Time Parameter
//-----------------------------------------------
FParamCalc;MyTimeMicroSec;MyTime*1.e6
//-----------------------------------------------
```

II – 1. e. Counter (FCounter)

Here's how to declare a counter parameter that could be applied to a parameter or a condition:

```
//================================================================
//
// Counter parameter declaration example
//
//================================================================
// This configuration file shows how to declare a "counter" on a parameter.
// The counter parameter is incremented each time the related parameter is active,  i.e. is
// present/read in an event. If the related parameter is a condition or a graphical condition,
//  the counter is incremented if the condition is true.
// The following examples assume that the raw parameters are "Param1", "Param2"
// and "Param3".
//-----------------------------------------------
FCounter;CntParam1;Counter on Param1
Param1
//-----------------------------------------------
FCounter;CntParam2;Counter on Param2
Param2
//-----------------------------------------------
// Counter on a condition
//
FCondition;CondParam1;(Param1>[0])&&(Param1<[1])
[0];48.5
[1];55
//
FCounter;CntCondParam1; Counter on the condition named "CondParam1"
CondParam1
//-----------------------------------------------
```

II – 1. f. Differential counter (FDiffCounter)

Here's how to declare a differential counter parameter allowing count rates computing:

```
//================================================================
//
// Differential counter parameter declaration example
//
//================================================================
// This configuration file shows how to declare a "differential counter" parameter.
// This kind of parameter allows to define counting rates. It simply computes the ratio
// of the differences of two parameters. The user has to define the two parameters and
// the frequency of the parameter evaluation.
// WARNING: the parameter indicated as the divider must increase with time.
// The following examples assume that the raw parameters are "Param1", "Param2"
// and "Param3".
//
//------------------------------------------------
// Define the counting rate (counts/sec) of Param1
//
// First define the counter for Param1
FCounter;CntParam1;Counter on Param1
Param1
// Declare the time parameter since it will be used
FParamTime;MyTime;My time parameter
// Finally declare the counting rate. Its value will be evaluated every 2.5s.
FDiffCounter;RateParam1;Counting rate of Param1
CntParam1;MyTime;2.5
//
//------------------------------------------------
// Define the ratio of Param1 values verifying a condition. The counter "CntParam1"
// previously defined will be used here.
//
// Define the condition
FCondition;CondParam1;(Param1>[0])&&(Param1<[1])
[0];48.5
[1];55
// Define the related counter
FCounter;CntCondParam1;Counter on CondParam1
CondParam1
// Define the ratio using a differential counter. Its value will be computed every
// 1000 Param1 counts.
FDiffCounter;RatioCondParam1;Ratio of the condition "CondParam1"
CntCondParam1;CntParam1;1000
//------------------------------------------------
```

II – 1. g. Persistent parameter (FParamLastValue)

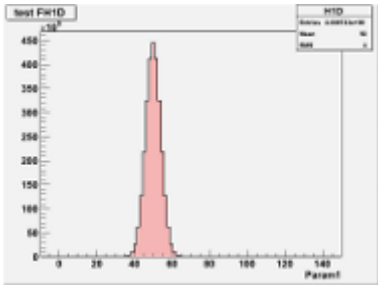
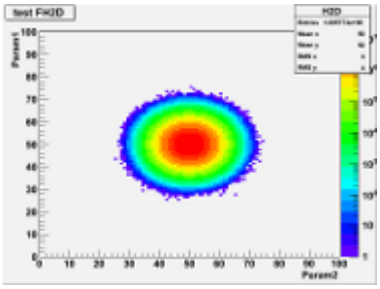
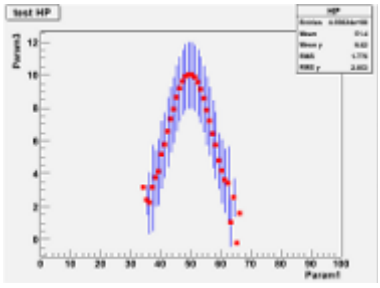
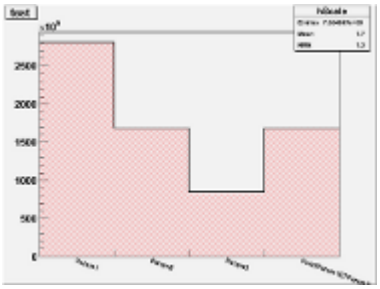
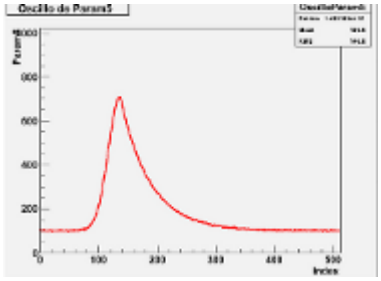
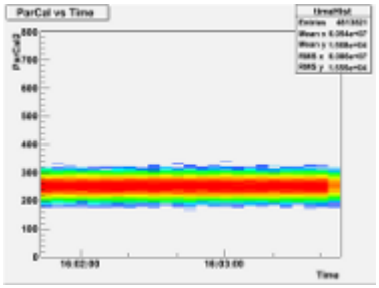
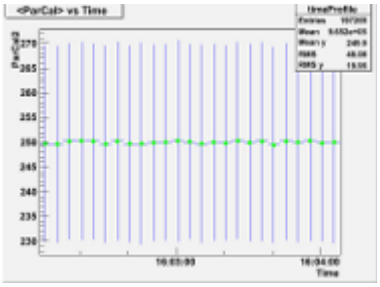
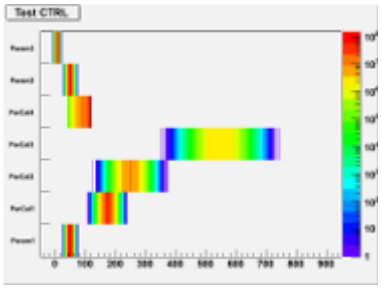
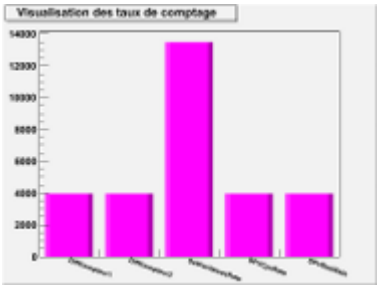
Here's how to declare a last value parameter allowing to define time of flight or differences compared to a reference for example:

```
//=============================================================
//
// Last value parameter declaration example
//
//=============================================================
// This configuration file shows how to declare a "last value" parameter.
// This type of parameter is always active. It is useful when one wants to keep the value
// of a raw parameter whose value is set with a low rate compared to the event rate. It
// also allows to store the value of a parameter each time a condition is fulfilled. This
// type of parameter can be used to build user defined time of flights for example.
// The following examples assume that the raw parameters are "Param1", "Param2"
// and "Param3" this last parameter being active at a very low rate.
//
//-----------------------------------------------
// Store the Param3 value the last time Param3 was active
//
FParamLastValue;LastParam3;Param3
//
//-----------------------------------------------
// Store the time at which Param3 is active
//
// First declare the Time Parameter (in seconds)
//
FParamTime;MyTime;My Time Parameter
// Define the condition on the activation of Param3
FCondition;HitParam3;Param3
// Then the time at which Param3 is active (in seconds)
FParamLastValue;TimeP3;MyTime;HitParam3
//
// One can define now a time difference (in milli seconds)
//
FParamCalc;TimeDiff;(MyTime-TimeP3)*1.e3
//-----------------------------------------------
```

II – 2. Histograms declarations

II – 2. a. List of RHB histograms types

Here's an overview of histograms types described in next chapters:

<table>
<tr>
<td>FH1D/FH1F
 
See II – 2. b. 1D histogram chapter</td>
<td>FH2D/FH2F
 
See II – 2. c. 2D histogram chapter</td>
</tr>
<tr>
<td>FProfile
 
See II – 2. d. Profile histogram chapter</td>
<td>FHScale
 
See II – 2. e. Scale histogram chapter</td>
</tr>
<tr>
<td>FOscilloH1F
 
See II – 2. f. Oscillogram chapter</td>
<td>FTimeHist
 
See II – 2. g. Time histogram 2D chapter</td>
</tr>
<tr>
<td>FTimeProfile
 
See II – 2. h. Time profile histogram chapter</td>
<td>FHCtrl2D
 
See II – 2. i. Control histogram 2D chapter</td>
<td>FHRawScale
 
See II – 2. j. Scale count histogram chapter</td>
</tr>
</table>

II – 2. b. 1D histogram (FH1D/FH1F)

Here's how to declare declare a 1D histogram and its filling condition:

```
//==========================================================================
//
// 1D histogram declaration example
//
//==========================================================================
// This configuration file shows how to declare a 1D histogram.
// The histogram is filled:
// 1. if no condition is specified, when the parameter is active.
// 2. if a condition is given, when the parameter is active and the condition is fulfilled
// The axis limits can be static or can be recomputed in order to cover the whole range
// of the parameter's values.
// The following examples assume that the raw parameters are "Param1", "Param2" and
// "Param3" this last parameter being active at a very low rate.
//------------------------------------------------
// Declare a 1D histogram of Param1 with no condition and with a static binning The
// histogram will contain 100 bins, the values ranging from 0 to 100.
//
FH1D;hParam1;Histogram of Param1 values
Condition :none
Param1;100;0;100
//------------------------------------------------
// Declare a 1D histogram of Param1 when a condition is fulfilled and with a
// static binning The histogram will contain 100 bins, the values ranging from
// 0 to 100.
//
// Define the condition
FCondition;CondParam2;abs(Param2-[0]) < [1]
[0];20
[1];1.5
// declare the histogram
FH1D;hParam1Cond;Histogram of Param1 values when CondParam2 is true
Condition :CondParam2
Param1;100;0;100
//------------------------------------------------
// Declare a 1D histogram of Param3 values with no condition and with a dynamic
// binning. The histogram will contain 250 bins, the initial values range is set
// from 0 to 25.
//
FH1D;hParam3;Histogram of Param3 values
Condition :none
Param3;250;0;25.;1
//------------------------------------------------
// Declare a 1D histogram of Param2 values with no condition and with a dynamic
// binning. The histogram will contain 200 bins, the values range being unknown.
FH1D;hParam2;Histogram of Param2 values
Condition :none
Param2;200;0;0;1
//------------------------------------------------
```

II – 2. c. 2D histogram (FH2D/FH2F)

Here's how to declare declare a 2D histogram and its filling condition:

```
//============================================================
//
// 2D histogram declaration example
//
//============================================================
// This configuration file shows how to declare a 2D histogram.
// The histogram is filled:
// 1. if no condition is specified, when the parameters are active.
// 2. if a condition is given, when the parameters are active and the condition is fulfilled.
// The axis limits can be static or can be recomputed in order to cover the whole range of
// the parameter's values. If one axis at least is declared with a dynamic binning, both axis
// will be set with a dynamic binning.
// The following examples assume that the raw parameters are "Param1", "Param2" and
// "Param3" this last parameter being active at a very low rate.
//
//------------------------------------------------
// Declare a 2D histogram of Param2 versus Param1 (Param1 is the abscissae and Param2
// the ordinate) with no condition and with a static binning The histogram will contain 100
// bins on the X axis, the values ranging from 0 to 100 and 100 bins on the Y axis, the values
// ranging from 0 to 100.
//
FH2D;hP2vsP1;Histogram of Param2 vs Param1
Condition :none
Param1;100;0;100
Param2;100;0;100
//
//------------------------------------------------
// Declare a 2D histogram of Param2 versus Param1 when a condition is fulfilled and
// with a static binnig. The histogram will contain 100 bins on the X axis, the
// values ranging from 0 to 100 and 100 bins on the Y axis, the values ranging
// from 0 to 100.
//
// Define the condition
FCondition;CondP1P2;(abs(Param2-[0]) < [1]) && (abs(Param2-Param1)<[2])
[0];20
[1];1.5
[2];25.5
// declare the histogram
FH2D;hP2vsP1Cond;Histogram of Param2 vs Param1 when CondParam2 is true
Condition :CondP1P2
Param1;100;0;100
Param2;100;0;100
//
```

```
//-------------------------------------------------
// Declare a 2D histogram of Param3 versus Param1 with no condition and with a
// dynamic  binning. The histogram will contain 250 bins on the X axis, the initial
// values range is set from 0 to 250 and 250 bins on the Y axis, the initial values
// range is set from 0 to 25.
//
FH2D;hP3P1;Histogram of Param3 vs Param1
Condition :none
Param1;250;0;250.;1
Param3;250;0;25.;1
//
//-------------------------------------------------
// Declare a 2D histogram of Param3 versus Param2 with no condition and with a dynamic
// binning. The histogram will contain 200 bins on both axis, the values ranges being
// unknown on the Y axis and set from 50 to 150 on the X axis.
//
FH2D;hP3P2;Histogram of Param3 vs Param2
Condition :none
Param2;200;50;150;1
Param3;200;0;0;1
//
//-------------------------------------------------
// Declare a 2D histogram of Param3 versus Param2 with no condition and with a dynamic
// binning. The histogram will contain 200 bins on both axis, the values ranges being
// unknown on the Y axis and set from 50 to 150 on the X axis. This declaration is
// equivalent to the previous one, even if the X axis binning is declared static.
//
FH2D;hP3P2bis;Histogram of Param3 vs Param2
Condition :none
Param2;200;50;150
Param3;200;0;0;1
//
//-------------------------------------------------
```

II – 2. d. Profile histogram (FProfile)

Here's how to declare declare a profile histogram and its filling condition:

```
//==========================================================
//
// Profile histogram declaration example
//
//==========================================================
// This configuration file shows how to declare a profile histogram
// The histogram is filled:
// 1. if no condition is specified, when the parameters are active.
// 2. if a condition is given, when the parameters are active and the condition is fulfilled.
// The X axis limits can be static or can be recomputed in order to cover the whole range
// of the X parameter's values.
// The following examples assume that the raw parameters are "Param1", "Param2" and
// "Param3" this last parameter being active at a very low rate.
//
```

```
//--------------------------------------------------
// Declare a profile histogram of <Param2> versus Param1 (<Param2> means mean value of
// Param2) with no condition and with a static binning The histogram will contain 100 bins on
// the X axis, the values ranging from 0 to 100.
//
FProfile;hMeanP2vsP1;Histogram of <Param2> vs Param1
Condition :none
Param1;100;0;100
Param2
//
//--------------------------------------------------
// Declare a profile histogram of <Param2> versus Param1 when a condition is fulfilled
// and with a static binning. The histogram will contain 100 bins on the X axis,
// the values ranging from 0 to 100.
//
// Define the condition
FCondition;CondP1P2;(abs(Param2-[0]) < [1]) && (abs(Param2-Param1)<[2])
[0];20
[1];1.5
[2];25.5
// declare the histogram
FProfile;hMeanP2vsP1Cond;Histogram of <Param2> vs Param1 when CondParam2 is true
Condition :CondP1P2
Param1;100;0;100
Param2
//
//--------------------------------------------------
// Declare a profile histogram of <Param3> versus Param1 with no condition and with a
// dynamic  binning. The histogram will contain 250 bins on the X axis, the initial
// values range is set from 0 to 250.
//
FProfile;hMeanP3P1;Histogram of <Param3> vs Param1
Condition :none
Param1;250;0;250.;1
Param3
//
//--------------------------------------------------
// Declare a profile histogram of <Param3> versus Param2 with no condition and with a
// dynamic binning. The histogram will contain 200 bins on X axis, the values ranges
// being unknown.
//
FProfile;hMeanP3P2;Histogram of <Param3> vs Param2
Condition :none
Param2;200;0;0;1
Param3
//
//--------------------------------------------------
```

II – 2. e. Scale histogram (FHScale)

Here's how to declare declare a scale histogram allowing to visualize several counters:

```
//=============================================================
//
// Scale histogram declaration example
//
//=============================================================
// This configuration file shows how to declare a scale histogram It show the number of
// times a parameter is activated or a condition is true.
// The following examples assume that the raw parameters are "Param1", "Param2" and
// "Param3" this last parameter being active at a very low rate.
//
//-----------------------------------------------
// Declare a scale histogram of Param1, Param2 and Param3.
//
FHScale;hScale;Test of Scale histogram.
Condition :none
Param1
Param2
Param3
--> EndOfList <--
//
//-----------------------------------------------
// Declare a scale histogram of Param1, CondParam1, Param2, CondParam1AND2, Param3
// and MixedCondition
//
// declare conditions
FCondition;CondParam1;(Param1>[0])&&(Param1<[1])
[0];48.5
[1];55
//
FCondition;CondParam1AND2;Param1 && Param2
//
FCondition;CondParam3;Param3<[0] || Param3 > [1]
[0];6
[1];27.5
//
FCondition;MixedCondition;CondParam1 && CondParam3
//
FHScale;hScale2;Second Test of Scale histogram.
Condition :none
Param1
CondParam1
Param2
CondParam1AND2
Param3
MixedCondition
--> EndOfList <--
//
//-----------------------------------------------
```

II – 2. f. Oscillogram (FOscilloH1F)

Here's how to declare declare an oscillogram to visualize a signal:

```
//===========================================================
//
// Oscillogram declaration example
//
//===========================================================
// This configuration file shows how to declare an oscillogram. An oscillogram allows
// to display the shape of an electronic signal. The related parameter has to be a vector
// parameter, i.e. a parameter with several values. The number of bins of the oscillogram
// has to be greater that the number of values in the parameter. By default, the value's index
// in the vector is attributed to the X axis. If needed, an affine transformation can be done
// on the index in order to convert in a user friendly unit (time signal for example). An
// homothetic transform on the Y axis can also be applied if needed.
// WARNING: if the FASTER acquisition system is used, the conversion factors are applied
// automatically and do not need to be defined in the declaration.
//
// The following examples assume that the raw parameters are "Param5" which contains
// 512 values.
//
//-------------------------------------------------
// Declare an oscillogram of Param5 with no condition and with 512 bins. On the Y
// axis, the values range from 0 to 1024.
//
FOscilloH1F;OscilloParam5;Oscillo of Param5
Condition :none
Param5;512;0;1024
//
//-------------------------------------------------
// Declare an oscillogram of Param5 with no condition and with 512 bins. On the Y
// axis, the values range from 0 to 1024. The last 5 signal will be shown.
//
FOscilloH1F;OscilloParam5Persist;Oscillogram of Param5 with the last 5 signals
Condition :none
Param5;512;0;1024;5
//
//-------------------------------------------------
// Declare an oscillogram of Param5 with no condition and with 512 bins. On the Y
// axis, the values range from 0 to 1024. The last 5 signal will be shown. The index
// to time function is t=-100+0.423*index. The digit to volt conversion factor is
// volt=0.0514*y
//
FOscilloH1F;OscilloParam5UserFriendly;User friendly oscillogram of Param5
Condition :none
Param5;512;0;2048;10;0.423;0.0514;-100.
//
//-------------------------------------------------
```

II – 2. g. Time histogram 2D (FTimeHist)

Here's how to declare declare a time histogram 2D allowing to visualize time evolution of a parameter:

```
//==================================================================
//
// Time histogram 2D declaration example
//
//==================================================================
// This configuration file show how to declare a time 2D histogram .  The X axis is the computer
// time axis. The limits are computed according to the  number of bins and to the time width of
// each bin. When the upper limit is reached, whole time axis (and the corresponding bin contents)
// is shifted by one bin,  i.e. the axis time limits are incremented by one bin width. This allow to
// check the variations of a parameter distribution with time in a moving time window of fixed
// total width. The Y axis limits can be static or can be recomputed in order to cover the whole
// range of the parameter's values.
// The following examples assume that the raw parameters are "Param1", "Param2" and "Param3"
// this last parameter being active at a very low rate.
//-------------------------------------------------
// Declare a time 2D histogram of Param1 with no condition and with a static binning
// for the last 60 seconds. The time X axis is divided in 30 bins with a bin width set
// to 2 seconds.
// The Y axis will contain 400 bins on the X axis, the values ranging from 0 to 100.
//
FTimeHist;P1vsTime;Param1 vs Time
Condition :none
30;2
Param1;400;0;100
//-------------------------------------------------
// Declare a time 2D histogram of Param1 with no condition and with a dynamic binning
// for the last 50 seconds. The time X axis is divided in 20 bins with a bin width set
// to 2.5 seconds.
// The Y axis will contain 200 bins, the initial values range set
// from 0 to 100.
//
FTimeHist;P2vsTime;Param2 vs Time
Condition :none
20;2.5
Param1;200;0;100;1
//-------------------------------------------------
// Another way to declare a time histogram is to declare a 2D histogram with a
// FParamTime on one of its axis. To cover the whole time range, the histogram has to
// be declared with a dynamic binning. Here is an example to see the time evolution of
// Param3.
// declare the time parameter
FParamTime;MyTime;My Time Parameter
//
FH2D;hP3vsTime;Param3 versus time
Condition :none
MyTime;100;0;0;1
Param3;200;0;200;1
//-------------------------------------------------
```

II – 2. h. Time profile histogram (FTimeProfile)

Here's how to declare declare a time profile histogram allowing to visualize time evolution of a parameter's average value :

```
//================================================================
//
// Time profile histogram declaration example
//
//================================================================
// This configuration file shows how to declare a time profile histogram.
// The X axis is the computer time axis. The limits are computed according to the number
// of bins and to the time width of each bin. When the upper limit is reached, whole time
// axis (and the corresponding bin contents) is shifted by one bin, i.e. the axis time limits
// are incremented by one bin width. This allow to check the variations of the mean value
// of a parameter with time in a moving time window of fixed total width.
//
// The following examples assume that the raw parameters are "Param1", "Param2" and
// "Param3" this last parameter being active at a very low rate.
//
//------------------------------------------------
// Declare a time profile histogram of Param1 with no condition for the last 60
// seconds. The time X axis is divided in 30 bins with a bin width set to 2 seconds.
//
FTimeProfile;MeanP1vsTime;<Param1> vs Time
Condition :none
30;2
Param1
//
//------------------------------------------------
// Another way to declare a time profile histogram is to declare a profile histogram
// with a FParamTime on one of its axis. To cover the whole time range, the histogram
// has to be declared with a dynamic binning. Here is an example to see the time
// evolution of the mean value of Param3.
//
// declare the time parameter
FParamTime;MyTime;My Time Parameter
//
FProfile;MeanP3vsTime;<Param3> versus time
Condition :none
MyTime;100;0;0;1
Param3
//
//------------------------------------------------
```

II – 2. i. Control histogram 2D (FHCtrl2D)

Here's how to declare declare a control histogram 2D allowing to visualize simultaneously the distributions of several parameters:

```
//=============================================================
//
// Control histogram 2D declaration example
//
//=============================================================
// This configuration file shows how to declare a 2D control histogram.  The axis limits of  the
// values axis can be static or can be recomputed in order to cover the whole range of the
// parameter's values. The following examples assume that the raw parameters are "Param1",
// "Param2" and "Param3" this last parameter being active at a very low rate.
//-------------------------------------------------
// Declare a 2D control histogram of Param1, Param2 and Param3. The parameters names will
// be on the X (horizontal) axis. The values axis (here the Y axis, i.e. the vertical axis) will contain
// 2000 bins, the values ranging  from 0 to 200.
FHCtrl2D;hCtrl;Test of a 2D Control histogram
Condition :none
Vertical;2000;0;200
Param1
Param2
Param3
--> EndOfList <--
//-------------------------------------------------
// Declare a 2D control histogram of calculated parameters. The values axis (here the horizontal X
// axis) will  contain 100 bins, the initial values limits being set ranging from 0 to 100. The value
// axis limits will be re-computed if needed (dynamic binning).
// Definition of the calculated parameters
FParamCalc;Energy1;[0]+Param1*[1]
[0];10
[1];2.5
FParamCalc;ParCal1;(Param1*[0])+([1]*Param2)
[0];1
[1];2.5
FParamCalc;DiffParam12;Param1-Param2
FParamCalc;ComplexExpression;([0]*sin(Param2*[1]+[2]))+[3]
[0];40
[1];0.14
[2];-0.456
[3];80
// declare the histogram
FHCtrl2D;hCtrlCalcParam;Test CTRL on calculated parameters
Condition :none
Horizontal;100;0;100;1
Energy1
ParCal1
DiffParam12
ComplexExpression
--> EndOfList <--
//-------------------------------------------------
```

II – 2. j. Scale count histogram (FHRawScale)

Here's how to declare declare a scale count histogram allowing to visualize simultaneously several counting rates:

```
//=============================================================
//
// Scale count histogram declaration example
//
//=============================================================
// This configuration file shows how to declare a "raw scale" histogram. For each parameter,
// it shows its value the last time it was activated. It was originally developed to show counting
// rates.
// The following examples assume that the raw parameters are "Param1", "Param2" and "Param3"
// this last parameter being active at a very low rate.
//
//-------------------------------------------------
// Declare a raw scale histogram showing the counting rates of Param1, Param2
// and Param3.
//
// Declare counting rates using FCounter and FDiffCounter and a FParamTime
FCounter;CntParam1;Counter on Param1
Param1
//
FCounter;CntParam2;Counter on Param2
Param2
//
FCounter;CntParam3;Counter on Param3
Param3
//
FParamTime;MyTime;My Time Parameter
//
FDiffCounter;RateParam1;Counting rate of Param1
CntParam1;MyTime;0.2
//
FDiffCounter;RateParam1;Counting rate of Param2
CntParam2;MyTime;0.5
//
FDiffCounter;RateParam3;Counting rate of Param3
CntParam3;MyTime;2.
//
FHRawScale;CountingRates;Counting rates of parameters Param1, Param2 and Param3
Condition :none
RateParam1
RateParam2
RateParam3
--> EndOfList <--
//
//-------------------------------------------------
```

II – 3. Use case example: Common reference for a run duration

Here's how to declare declare a common reference for a whole run duration:

```
//=================================================================
//
// Common reference run reference declaration example
//
//=================================================================
// This configuration file shows how to define a reference (persistent) parameter
// for the whole duration of a run or an analysis. In this example, the starting
// time of the process will be defined.
//
// The following examples assume that the raw parameters are "Param1", "Param2"
// and "Param3".
//-------------------------------------------------
// Declare a condition which will be true when the run starts. In this example,
// it will be assumed that the run starts when the counter on Param1 is equal to 1.
//
//
// Declaration of the counter on Param1
//
FCounter;CntParam1;Counter on Param1
Param1
// Declaration of the condition
FCondition;StartRun;CntParam1==1
// Declaration of the time parameter
FParamTime;MyTime;My Time Parameter
// Declaration of the staring time
FParamLastValue;TStartRun;MyTime;StartRun
//
// This last parameter will remain active all along the run duration. It can be
// used as a reference time. For example, the run time can be defined this way:
//
FParamCalc;RunTime;MyTime-TStartRun
//
//-------------------------------------------------
```

III – Special FASTER DAQ pid file

III – 1. PID file description

A dedicated **Parameters Input Description** file (pid extension) is attempted by RHB to read data from FASTER DAQ as, up to now, raw parameters can not be directly read and chosen by interacting with FASTER.

This pid file can be edited and relies on a very simple structure like that:

```
# a comment
faster_label:faster_type:parameter_name
```

One line is made of 3 fields, all separated by the special "**:**" symbol.
Comments can be added by beginning a line with the "#" character.

Possible FASTER types are:

- TREF : Time Reference without any measure
- TREF_TDC : Time Reference with its time precision
- RF : Radio Frequency
- RF_COUNT : RF counters (TRIG/SENT)
- OSC : one Oscillogram (704 samples)
- QDC1 : one QDC
- QDC2 : two QDCs
- QDC3 : three QDCs
- QDC4 : four QDCs
- QCOUNT : QDC counters (CALC/SENT)
- ADC1 : one ADC
- ACOUNT : ADC counters (CALC/SENT/TRIG)
- ELECTRO : Electrometer
- SCALER : Scaler
- SCALER_COUNT : Scaler counter (CALC/SENT)

In this way, if a known faster type is recognized by RHB, it will automatically build all raw parameters for this type, all based on the given parameter's name.

User must however take care to give a right label numbers in full compliance with FASTER system settings.

III – 2. PID file example

Here is a simple pid file example:

```
########################################
#                                      #
# Parameters Input Description File example  #
#                                      #
########################################
#
# Experiment: FASTER DAQ TEST with 2 CARAS boards.
#
# On the first CARAS board, first channel “X1” has 2 QDC measures. Its counters are used for
# control and its oscillogram can be used for setting experiment for example.
# The second signal “VREF” on this board is a reference voltage coming from another electronics
# equipment. Furthermore its counter values and its oscillogram, only 1 QDC measure is required.
#
################
# CARAS board 1 #
################
#
# X1 channel
# --------------
#
# Define 2 QDC measurements
1:QDC2:X1
# Define counters
11:QCOUNT:X1
# Define oscillogram
21:OSC:X1
#
# VREF channel
# ------------------
#
# Define 1 QDC measurement
2:QDC1:VREF
# Define counters
12:QCOUNT:VREF
# Define oscillogram
22:OSC:VREF
#
# On the second CARAS board, a radio frequency “BEAM” coming from beam control is used to
# compute a Time Of Flight. As before, its counters and its oscillogram are proposed for analysis.
# No signal is connected to the second input of this board.
#
################
# CARAS board 2 #
################
#
# BEAM channel
# -------------------
#
# Define radio frequency measurement
3:RF:BEAM
```

```
#
# Define counters
13:RF_COUNT:BEAM
# Define oscillogram
23:OSC:BEAM
#
# SPARE channel
# -------------------
# To complete if needed !
#
```

As a result, at RHB startup, raw parameters created from this file should appear in the Terminal:

```
…
FasterDevice opened.
Parameter Input Description File Test.pid opened.
X1_t (1) : 671088896/512
X1_dt (2) : 671088897/128
X1_QDC1 (3) : 671088898/128
X1_QDC1_satured (4) : 671088899/1
X1_QDC2 (5) : 671088900/128
X1_QDC2_satured (6) : 671088901/1
X1_COUNT_t (7) : 838863616/512
X1_CALC (8) : 838863617/16
X1_SENT (9) : 838863618/16
X1_OSC_t (10) : 352326912/512
X1_OSC (11) : 352326913/16
VREF_t (12) : 671089152/512
VREF_dt (13) : 671089153/128
VREF_QDC1 (14) : 671089154/128
VREF_QDC1_satured (15) : 671089155/1
VREF_COUNT_t (16) : 838863872/512
VREF_CALC (17) : 838863873/16
VREF_SENT (18) : 838863874/16
VREF_OSC_t (19) : 352327168/512
VREF_OSC (20) : 352327169/16
BEAM_t (21) : 318767872/512
BEAM_per (22) : 318767873/128
BEAM_satured (23) : 318767874/1
BEAM_dt (24) : 318767875/128
BEAM_pll (25) : 318767876/128
BEAM_t (26) : 335547648/512
BEAM_TRIG (27) : 335547649/16
BEAM_SENT (28) : 335547650/16
BEAM_OSC_t (29) : 352327424/512
BEAM_OSC (30) : 352327425/16
...
```

Use then these parameters to declare your calculated parameters and histograms as described in chapter II.