

## Group decision

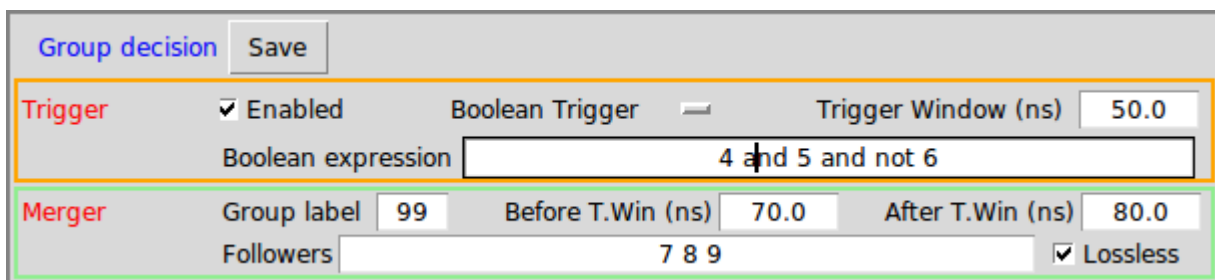
This decision merges data in coincidence to create events from different channels. The merged data are included in a GROUP data (type 10) which represents an event.

This module is divided in two panels : Trigger & Merger.

- Trigger panel defines conditions that do trigger a new Group,
- Merger panel defines how the group is built.

### Trigger panel

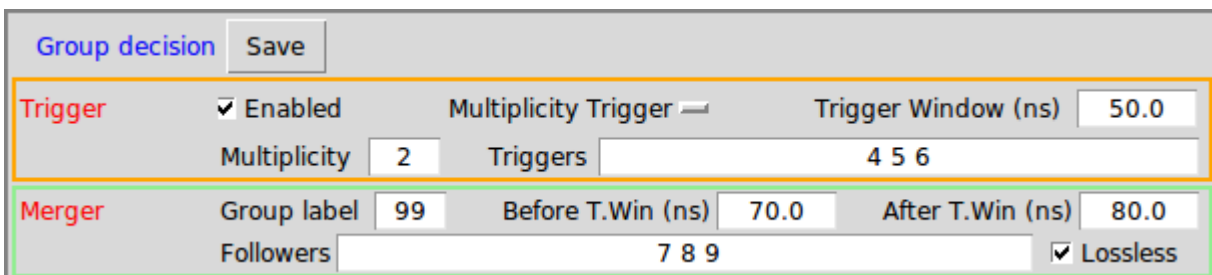
Data coincidences are expressed in the trigger panel with combinations of labels (data id) in a time window. Those labels named **Triggers** are combined in a **Trigger Expression** which can take two forms : **Boolean** or **Multiplicity**. A **Trigger Window** is any time window started by one of the **Triggers**.



The screenshot shows the 'Group decision' interface. The 'Trigger' panel is highlighted with an orange border and contains the following settings: 'Enabled' is checked, 'Boolean Trigger' is selected, 'Trigger Window (ns)' is 50.0, and the 'Boolean expression' is '4 and 5 and not 6'. The 'Merger' panel is highlighted with a green border and contains: 'Group label' 99, 'Before T.Win (ns)' 70.0, 'After T.Win (ns)' 80.0, 'Followers' 7 8 9, and 'Lossless' checked.

fig 1: Group decision with « Boolean expression » trigger

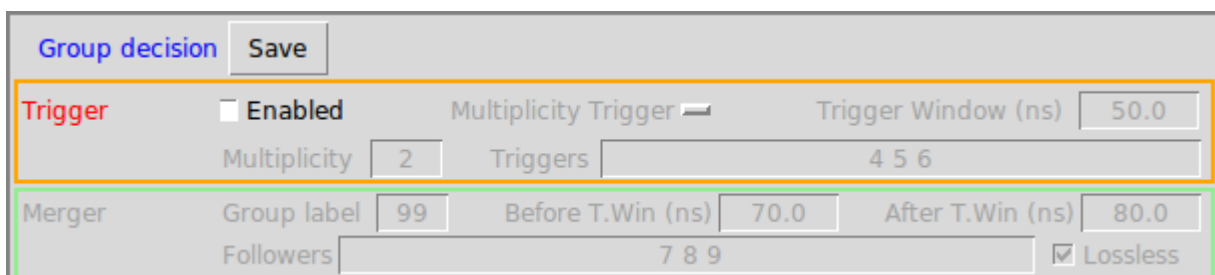
With **Boolean Trigger** selected (fig.1), trigger occurs when the presence of data in the **Trigger Window** evaluate the **Boolean expression** to True. The boolean expression is composed with labels, parenthesis and boolean operators [and - or - not]. Each label is evaluated to True when a corresponding data is present in the window. In the given example, grouping is triggered when at least two data labelled #4 and #5 are seen in a window of 50ns without any #6.



The screenshot shows the 'Group decision' interface. The 'Trigger' panel is highlighted with an orange border and contains: 'Enabled' checked, 'Multiplicity Trigger' selected, 'Trigger Window (ns)' 50.0, 'Multiplicity' 2, and 'Triggers' 4 5 6. The 'Merger' panel is highlighted with a green border and contains: 'Group label' 99, 'Before T.Win (ns)' 70.0, 'After T.Win (ns)' 80.0, 'Followers' 7 8 9, and 'Lossless' checked.

fig 2: Group decision with « Multiplicity » trigger

With **Multiplicity Trigger** selected (fig.2), trigger occurs when at least **Multiplicity** of different data from the list **Triggers** is seen in the **Trigger Window**. In the given example, grouping is triggered when 2 labels from [#4, #5, #6] are present in a 50ns interval.



The screenshot shows the 'Group decision' interface. The 'Trigger' panel is highlighted with an orange border and contains: 'Enabled' unchecked, 'Multiplicity Trigger' selected, 'Trigger Window (ns)' 50.0, 'Multiplicity' 2, and 'Triggers' 4 5 6. The 'Merger' panel is highlighted with a green border and contains: 'Group label' 99, 'Before T.Win (ns)' 70.0, 'After T.Win (ns)' 80.0, 'Followers' 7 8 9, and 'Lossless' checked.

fig 3: Inhibited Group decision

The check button **Enabled** activates or inhibits the decision. When inhibited (fig.3), the decision is transparent to the data flow and doesn't affect it.

## Merger panel

For each trigger, the Merger creates a new Group data composed with all **Groupable Data** present in the **Grouping Window**. A **Groupable Data** is a data which label is in **Triggers** or **Followers**, and the **Grouping Window** is the conjunction of **Before T.Win**, **Trigger Window** and **After T.Win**.

In the examples given, a resulting group data will have the following characteristics :

- type = 10,
- label = 99,
- clock = beginning of the « Trigger Window »,
- load = list of all data from (4, 5, 6, 7, 8, 9) seen in [clock-70ns ; clock+50+80ns]

Parameter **lossless** is used to keep all the data in the flow. If this parameter isn't selected, **each groupable data outside a group will be removed from the flow**.

Lossless or not, the decision only affects the **groupable data**, and keeps the others unchanged.

## Reading stored data

An acquisition is stored in two files : the setup parameters are stored in an ascii file (.setup) and the data in a little endian binary file (.fast).

Data are sorted in function of time, this format is optimized for time stamped data flow and online treatments.

Any data is composed of five fields :

- *type\_alias* type of the data,
- *label* data id,
- *clock* time associated with that data,
- *load\_size* size in octet of the *load* part,
- *load* specific part of the data, depending of its *type\_alias*.

Stored data can be replayed with **RHB** (REF : how to RHB a file) or processed by a C program using **fasterac** library.

The package **fasterac** is available for Ubuntu LTS at the following repository :

« *deb http://faster.in2p3.fr/distribution/ubuntu/ lucid main* »

For other GNU systems, it can be installed from the archive *fasterac-X.Y.tar.gz* with :

```
> wget http://faster.in2p3.fr/fasterac-X.Y.tar.gz
> tar xvzf fasterac-X.Y.tar.gz
> cd fasterac-X.Y
> ./configure ; make ; make install
```

(this example is given with the « *lucid* » LTS and the *X.Y* version of *fasterac*)

This package contains the program **disfast**, the library **fasterac** and **example codes**.

- **disfast** is used to display and to check the content of « .fast » file in a console (*man disfast*),
- **fasterac** is a C library for handling Faster files and data (*man fasterac*),
- **example codes** (*/usr/share/fasterac/examples*) show how to read a data file, to handle data types, to make a basic data treatment, or to create a « *root tree* » from a Faster file.

The simple example below shows the different aspects of handling data from a file.

It consists in displaying the label, the time and the ratio of the first two charges of every QDC data (having at least two charges).

```
1  #include <stdio.h>
2
3  #include "fasterac/fasterac.h"           // Include the specifications of
4  #include "fasterac/qdc_caras.h"        //   - the fasterac library,
5                                         //   - the QDC data formats.
6  int main (int argc, char** argv) {
```

```

7
8     faster_file_reader_p  reader;           // Declare a data reader (from fasterac.h),
9     faster_data_p        data;           // a faster data,
10    unsigned short       label;          // the label of a data,
11    double                clock;         // the clock in seconds of a data.
12    double                ratio;         //
13    qdc_t_x4              qdc;           // Declare a QDC_TDC with 4 charges (from qdc_caras.h),
14                                // this type fits all qdc types to get two charges.
15
16    reader = faster_file_reader_open (argv[1]); // Open the file given in argument (ie "myfile.fast").
17    while ((data = faster_file_reader_next (reader)) != NULL) // Get the next data till the end.
18    {
19        type = faster_data_type_alias (data); // Get the type of the current data.
20        if (type == QDC_X2_TYPE_ALIAS || //
21            type == QDC_X3_TYPE_ALIAS || //
22            type == QDC_X4_TYPE_ALIAS || //
23            type == QDC_TDC_X2_TYPE_ALIAS || // Those constants are defined in qdc_caras.h,
24            type == QDC_TDC_X3_TYPE_ALIAS || // there is all QDC containing at least two charges.
25            type == QDC_TDC_X4_TYPE_ALIAS || //
26            type == QDC_TOF_X2_TYPE_ALIAS || //
27            type == QDC_TOF_X3_TYPE_ALIAS || //
28            type == QDC_TOF_X4_TYPE_ALIAS) // If the current data is one of those QDC then ...
29        {
30            label = faster_data_label (data); // get the label of that QDC data,
31            clock = faster_data_clock_sec (data) ; // get its time stamp in second,
32            faster_data_load (data, &qdc); // get ist specific part (the QDC part).
33            ratio = (double) qdc.q2 / qdc.q1; // Calculate the charge ratio q2/q1 of that QDC,
34            printf ("type=%d label=%d clock=%0.9f Q2/Q1=%f\n", type, label, clock, ratio); // and display all the infos.
35        }
36    } // Here is the end of file,
37    faster_file_reader_close (reader); // close it.
38
39    return 0;
40 }

```

Line 3 includes the specifications of the *fasterac* library (*/usr/include/fasterac/fasterac.h*). The library (*man fasterac*) proposes types and functions to read data files (*faster\_file\_\**) and to handle faster data (*faster\_data\_\**). Almost every lines of that example are referring the library (*lines 3, 8 to 11, 16, 17, 19, 30 to 32, and 37*). All data are manipulated without having to know the specific part of each. The program works on any faster file containing any type of data, and it only focuses on QDCs.

QDC types are involved in the lines 4, 13, 20 to 28, and 33. Line 4 includes the specifications of QDC data formats (*/usr/include/fasterac/qdc\_caras.h*). The header file specifies constants defining QDC types, data formats and functions.

This example shows how to manipulate any data in a generic way with *fasterac*, and how to access the specific parts of data when needed.

To know how to handle those specific parts, refer to the considered header file (ie *qdc\_caras.h*) and the demo program given in example codes :

***/usr/share/fasterac/examples/data\_reader/reader\_demo.c***

This demo treats any type of data (QDCs, ADCs, Oscillo, Group, etc ...), and access every particular field without referring to the inner representation of data. Each data type is referenced in that code by a « switch case » showing how to access each field of the type.

Here is the case of a « one charge QDC » which contains two fields *q1* and *q1\_saturated* :

```

... //
#include « fasterac/qdc_caras.h » // QDC specifications
... //
qdc_x1 q; // a one charge qdc declaration
... //
switch (faster_data_type_alias (data)) { // switch on the type of the current data
... //

```

```
case QDC_X1_TYPE_ALIAS : // case of a one charge QDC
    faster_data_load (data, &q); // get the specific part of the data
    printf (" QDC_X1 : q1=%d", q.q1); // direct access to the q1 field
    if (q.q1_saturated) printf (" saturated : q1"); // direct access to the q1_saturated field
    printf ("\n"); //
    break; // no need to know the low level format
... //
} //
... //
```