LHCb Note 2005-015, TRIG & COMP
LPHE Note 2005-011
August 17, 2005

HLT Exclusive Selections Design and Implementation

P. Koppenburg
CERN

L. Fernández
LPHE, EPFL

Abstract

The present note describes the requirements, design choices and implementation of the HLT exclusive selection prototype.

This prototype is the first complete attempt to use the online tracking to select signal and reject background down to 200 Hz. The details of the "physics" outcome can be read in a subsequent note [1]. Here we address a few technical problems that have been observed and set some requirements for the future.

The Appendix is a user guide that explains how to use the released code, using the algorithms described in note 2005-016 [2].

Contents

1 Introduction **1**
- 1.1 Prototype . . . 1

2 Requirements **2**

3 Design **3**

4 Implementation **3**
- 4.1 Reconstruction and Generic HLT . . . 4
- 4.2 Exclusive Streams . . . 4
- 4.3 Decision . . . 6
- 4.4 Usage of RICH . . . 6
- 4.5 HLT Packages . . . 7

5 Outlook and Further Developments **7**
- 5.1 Lessons . . . 7
- 5.2 Requirements to Reconstruction . . . 8
- 5.3 Going Inclusive . . . 8
- 5.4 What about a 1 MHz Readout? . . . 8

6 Conclusion **8**

A How to Access the Trigger Decision **9**
- A.1 Run the Trigger . . . 9
- A.2 Warning: Make sure you run in On- or Offline Mode! . . . 9
- A.3 Retrieve the Information . . . 10
- A.4 The `HltScore` Class . . . 10
- A.5 Some Monitoring and Checking . . . 11
- A.6 RICH . . . 11

B How to Add a Selection **12**
- B.1 General Structure of the Selection . . . 12
- B.2 Pre-selection of HLT Particles . . . 13
- B.3 Intermediate States . . . 13
- B.4 Final State . . . 14
- B.5 Add your Selection to `HltScore` . . . 15
- B.6 Offline Selections . . . 15
- B.7 MC Truth . . . 16
- B.8 Efficiency Correlations . . . 17
- B.9 Plots . . . 17

C Full HLT sequence **18**

1 Introduction

The LHCb High-Level-Trigger (HLT) will run at 40 kHz after L1 on the same PC farm as the latter and will use in average 400 CPUs. The average computing time per event should thus not exceed 10 ms. It will have an output rate of 2 kHz divided in four main streams:

Exclusive B (~ 200 Hz): The core physics stream with exclusively reconstructed decays including sidebands and control channels.

D^* (~ 300 Hz): PID-blind D^* events with $D^0 \to hh$ and no D^0 mass cut. These events allow to measure the PID efficiency and mis-ID rate and can also be used for *CP* measurements in D decays.

Dimuon (~ 600 Hz): Time unbiased dimuons with a mass above 2.5 GeV. These events are used to measure the uncertainty on lifetime measurements.

Inclusive B (~ 900 Hz): Events with one high p_T and high-IP muon, used for systematic studies of the trigger efficiency and for data mining. Because of the muon this sample is highly tagging-enriched.

The first complete HLT prototype described here implements this functionality and has been successfully run in the Real Time Trigger Challenge (RTTC) in the MOORE application.

The note describes in detail the implementation of the exclusive B stream, optimized to select 10 "core physics" channels described below. It also covers the D^* stream that is based on the same implementation. The inclusive and dimuon streams are part of the generic HLT and described elsewhere [3].

We first remind about the requirements for the HLT exclusive selections and describe the implementation choices. Then we give some ideas about how the HLT is likely to evolve until it runs in the real experiment in 2007.

In the Appendix we explain how to retrieve the trigger decision and how to add a selection. This section has to be read using our other note about generic selection algorithms [2].

The present note does not show any efficiency, background retention numbers or optimized cuts, nor any timing results. This will be addressed in a subsequent note [1].

1.1 Prototype

The first prototype of the HLT exclusive selections stream has been implemented based on 10 core physics channels presented by Roger Forty at the June 29 2004 Tuesday Meeting.[1]

This selection corresponds to an educated guess of the main measurements one would like to make during year one of data taking:

1. $B_s \to D_s h$ for the measurement of Δm_s ($h = \pi$) and γ ($h = K$).
2. $B^0 \to J/\psi K^0_S$ for ϕ_d;
3. $B \to hh$ for γ (and α);
4. $B^0 \to DK^*$ for γ;
5. $B_s \to J/\psi\phi$ for ϕ_s;

[1] http://agenda.cern.ch/askArchive.php?base=agenda&categ=a04957&id=a04957s1t1/transparencies

6. $B_s \to \phi\phi$ for ϕ_s in b $\to$ s penguins;
7. $B^0 \to D^*\pi$ for B^0 oscillations;
8. $B^0 \to K^*\mu\mu$ to look for a non-SM forward-backward asymmetry;
9. $B \to \mu\mu$ to look for rate enhancements due to New Physics;
10. $B_s \to \gamma\phi$ to look for enhancements of *CP* violation due to New Physics.

Presently the Physics Task Forces are preparing a more precise strategy for the first years of data taking and will provide new HLT selections for other channels. For example one needs to study the efficiency of the HLT for channels including electrons ($J/\psi \to ee$ or $B \to K^{(*)}ee$), π^0 ($B \to \rho\pi$) or K^0_S ($B \to \phi K^0_S$).[2] One also would like to consider the advantages of inclusive streams. This is discussed in the outlook (Sec. 5.3).

2 Requirements

The design requirements for the exclusive HLT are the following:

Efficiency: The HLT having access to the same data as the offline selections it is assumed that its efficiency for offline selected events can be very close to 1.

Output rate: The total output rate for the exclusive selections is 200 Hz, including sidebands and control channels, to which one adds approximatively 600 Hz of dimuons, 300 Hz of D^* and 900 Hz of single muons for a total of 2 kHz.

Speed: The HLT runs after L1 at 40 kHz on 1/4 of the 1600 CPUs in the online farm.[3] This leads to an average CPU time of 10 ms per L1-accepted event.

It is assumed that this corresponds roughly to 60 ms on a 1 GHz Pentium III processor. The experience with the present 2.8 GHz Pentium-IV processors shows that this assumption might be quite optimistic.

To these requirements set by the trigger, we express here some more or less implicit "computing" requirements:

Stability: The HLT should not crash during a typical run of 10 hours, i.e. $\sim 10^6$ events per CPU and run. This sets strong constraints to possible memory leaks and requires very well tested code.

Configuration: The HLT should support as many as $O(100)$ selections to run in the same process. We would like to be able to add and remove selections quickly (typically between two runs). We also would like to be able to change a cut quickly.

User-friendliness: The typical selections will not be developed and tested by a limited number of HLT experts. A typical user should be able to quickly develop and test a new HLT selection.

Standard: The selection criteria ("cuts") applied in the HLT should be standard and have a maximal (i.e. 100%) correlation with the ones applied in the stripping and the final selections. Dedicated HLT code that would yield slightly different results from the offline results should be avoided as much as possible.

[2] In $B^0 \to J/\psi K^0_S$ the K^0_S is not needed to trigger.
[3] The 1 MHz readout scheme would not change dramatically this figure. See Section 5.4.

Avoid duplications: No operation should be performed if not necessary or if already done. Whenever possible similar operations should be performed only once. A typical example is the pre-selection of tracks and the building of intermediate particles used in several pre-selections, as the D^0.

3 Design

These requirements lead to the following design choices:

1. To ensure a maximal efficiency on offline selected events, the selections of the exclusive HLT stream have to mimic the offline selections but with looser requirements (if possible). There are thus *n* output streams that are "or-ed" in order to make the 200 Hz exclusive HLT decision.

2. To avoid duplication of computing all shared particles are built only once per event: The D^0, ϕ, K^* and J/ψ are shared among all possible exclusive selections. A refinement is then performed in the dedicated exclusive streams.

3. To limit the combinatorics the requirements on the final states are applied as early as possible. Impact parameter and p_T cuts are applied at the particle creation level. This design choice is sometimes in contradiction with choice 1.

4. The stability of the system is ensured by minimizing the amount of code used. All basic operations like

   - reconstructing the decay $R \to AB[C\ldots]$ applying some typical cuts, or
   - selecting a sub-sample of particles from a given sample

   are performed by several instances of two algorithms (`MakeResonances` and `FilterDesktop`, respectively). They are described in note [2].

5. The above-mentioned algorithms use the same tools to apply the selection cuts. This ensures that all cuts are defined in a standard, reproducible and documented way. For instance the impact parameter significance requirement $\mathrm{IP}/\sigma_{\mathrm{IP}} > 3$ *always* means that the ratio of the *unsigned* impact parameter to its calculated error has to be larger than 3 with respect to *all* reconstructed primary vertices.

6. These algorithms are configured by options[4] and share the same option syntax, which facilitates the reshuffling of options from one algorithm to the other. This allows an easy extension of the HLT to other selections.

7. Also, since everything is defined by options, the full HLT chain can be configured from the configuration database. No code needs to be recompiled in order to change the behaviour of the HLT. The reading of the database is not implemented yet.

4 Implementation

This section describes in more detail the code organization, the algorithms and the data flow.

[4] Whether the options are coded using GAUDI option files, as they are now, or PYTHON is irrelevant for this discussion. Pere Mato has recently prepared a proposal for a PYTHON-steered GAUDI.

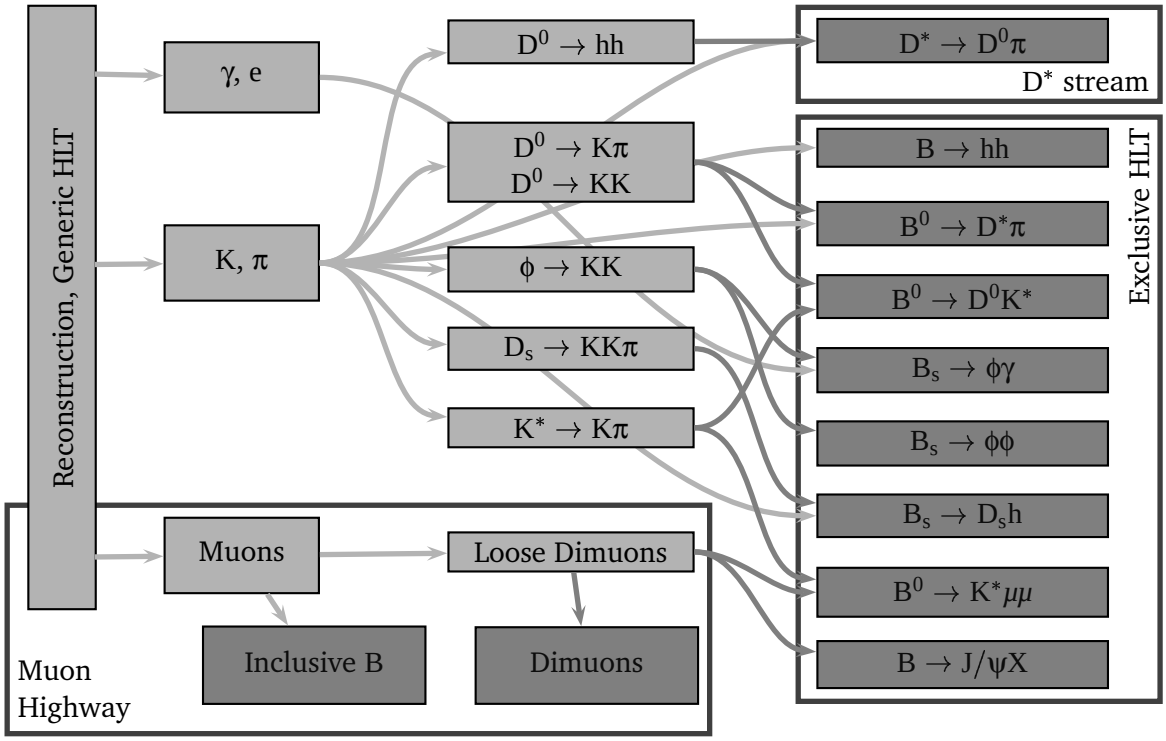

Figure 1: Simplified data flow in the HLT. Each gray box is an algorithm (or a set of algorithms).

4.1 Reconstruction and Generic HLT

A detailed description of the pattern recognition [4] and the generic HLT [3] is beyond the scope of this note. In the following we briefly summarize the procedure.

First the Velo RZ- and Velo 3D-tracking are performed and the primary vertices (PV) are built.[5] Then one enters the HLT generic sequence[6] where first the `VeloTT` algorithm is run and the muon pattern recognition is made. The forward tracking is done for selected tracks and muon candidates. The errors from the tracking are not used, but recomputed from a parameterization [5]. Events with good muon candidates and heavy dimuons lead to a HLT yes for the inclusive B and dimuon streams. For the others, the generic HLT decision is made by re-doing the L1-generic line with an improved precision (so-called L1-confirmation), in particular with better momentum determination.

4.2 Exclusive Streams

After the HLT-generic yes, one enters the exclusive part of the HLT. Presently this happens even if the event is accepted by the inclusive B and dimuon streams. Generally all possible decision streams are run in the HLT regardless of the results of the others which allows to study correlations.

The data flow of the HLT prototype is sketched in Figure 1. The full sequence is listed in Appendix C.

1. First the `TrgParticleMaker` is executed to make pion and kaon candidates. By default every track is a made as a pion and as a kaon candidate, but one can use the

[5] Controlled by `$TRGSYSROOT/options/Hlt.opts`.
[6] Controlled by `$HLTGENERICROOT/options/HltGeneric.opts`.

RICH to tighten the kaon selection. It is not foreseen to select pions based on some PID requirements as pions are used later on as generic hadrons with no clear PID assignment. The particles are stored in `/Event/Phys/Trg` (from now on `/Event/Phys` is implicit). Although Velo-TT tracks are created by the `TrgParticleMaker`, only long tracks are used in the HLT selections with the exception of slow pions from D^*.

2. The particles are distributed according to their PID to `HLTPions`, `HLTKaons` and `HLTPhotons`.[7] Electrons and π^0 are lost at this level, but easy to recover (see for instance Appendix B).

3. A first refinement of the particles is performed applying loose cuts. Four categories are created:

   - `HLTDetachedPions`: π with $\mathrm{IP}/\sigma_{\mathrm{IP}} > 2$;[8]
   - `HLTPreselPions`: Detached π with 300 MeV p_T ;
   - `HLTPreselKaons`: Kaons with 300 MeV p_T and $\mathrm{IP}/\sigma_{\mathrm{IP}} > 2$;
   - `HLTPreselPhotons`: Photons with 300 MeV p_T .[9]

   Detached pions will be for instance used as slow pions to form D^*. For the other charged particles, only those made from long tracks are kept.

4. The kaons and pions from `HLTPreselXXX` are then combined to make the following composite particles:[10]

   - $D^0 \to K\pi$ and $D^0 \to KK$ (`HLTSharedD02KH`) with $\chi^2 < 50$, $p_T > 1.5\,\mathrm{GeV}$ and in a $\pm 50\,\mathrm{MeV}$ mass window ;
   - $D^0 \to \pi\pi$ (`HLTLooseD02PiPi`) with $\chi^2 < 50$, $p_T > 1.5\,\mathrm{GeV}$; and in a $-50\,\mathrm{MeV} + 1500\,\mathrm{MeV}$ mass window ;
   - $K^* \to K\pi$ (`HLTSharedKstar`) with $\chi^2 < 200$ and in a $\pm 30\,\mathrm{MeV}$ mass window ;
   - $\phi \to KK$ (`HLTSharedPhi`) with $\chi^2 < 200$ and in a $\pm 30\,\mathrm{MeV}$ mass window ;
   - $D_s \to KK\pi$ (`HLTSharedDs`) with all cuts from the $B_s \to D_s K$ HLT selection; in particular a $\pm 45\,\mathrm{MeV}$ mass window is applied.

   All cuts listed in this section are to be revised by the Physics Task Forces and will be updated in note [1].

   Additionally, the generic HLT saves all dimuon candidates before any mass cut in `HltGeneric/JPsi`.

5. Finally the shared particles with some refinement are combined to make B candidates for all the 10 core physics channels. The detailed cuts used are explained in Ref. [1].

   - $B_s \to \gamma\phi$ is built using ϕ from `HLTSharedPhi` and photons from `HLTPreselPhotons`.[11]

[7] This step is controlled by `$HLTSELECTIONROOT/options/TrgTracks.opts`.

[8] Controlled by `$HLTSELECTIONROOT/options/HLTFilterParticles.opts`, as well as the two next categories.

[9] Controlled by `$HLTSELECTIONROOT/options/HLTNeutrals.opts`.

[10] Controlled by `$HLTSELECTIONROOT/options/HLTSharedD0.opts`, `$HLTSELECTIONROOT/options/HLTSharedKstar.opts`, `$HLTSELECTIONROOT/options/HLTSharedPhi.opts` and `$HLTSELECTIONROOT/options/HLTSharedDs.opts`.

[11] Controlled by `$HLTSELECTIONSROOT/options/HLTselBs2PhiGamma.opts`.

- $B \to hh$ is built as $B \to \pi\pi$ using π from `HLTPreselPions`. The mass window is wide enough to accommodate all $B \to hh$ modes.[12]
- $B_s \to D_s h$ is built as $B_s \to D_s\pi$ using D_s from `HLTSharedDs` and π from `HLTPreselPions`. The mass window is wide enough to accommodate $B_s \to D_s K$.[13]
- $B_s \to \phi\phi$ is built using ϕ from `HLTSharedPhi`.[14]
- $B^0 \to D^0 K^*$ is built using D^0 from `HLTSharedD02KH` and K^* from `HLTSharedKstar`.[15]
- $B^0 \to D^*\pi$ is built using a D^0 from `HLTSharedD02KH`, a slow pion from `HLTDetachedPions` and a fast pion from `HLTPreselPions`.[16]
- $B^0 \to K^*\mu\mu$ is built using dimuons from `HltGeneric/JPsi` and K^* from `HLTSharedKstar`.[17]
- $B^0 \to J/\psi X$ is built using dimuons from `HltGeneric/JPsi`.[18]
- $B \to \mu\mu$ is built using dimuons from `HltGeneric/JPsi`.[19]

6. The D^* stream is built using $D^0 \to \pi\pi$ (no D mass cut) from `HLTLooseD02PiPi` and π from `HLTDetachedPions`.[20]

Throughout the code to calculate IP and form vertices the tracks are extrapolated assuming a straight trajectory and the errors are computed from a p_T-dependent parameterization. This is the only place in the HLT exclusive code where the behaviour is different than in an offline analysis, but for really straight lines one can show that the two methods are mathematically equivalent [5, 6].

4.3 Decision

After the HLT sequence the algorithm `HltDecision` fills the `HltScore` class. This class summarizes the HLT decision. It is described in Appendix A.4.

4.4 Usage of RICH

In the default implementation the RICH is not used. All tracks are made both as π and K candidates. If one switches on the RICH[21] only tracks compatible with the K hypothesis are made as kaons. This does not affect pions in order to guarantee that the flavour-blind selections still work for the h=K case.

Using the RICH allows to reduce the background and to reduce the combinatorics (which saves time), but takes about 10 ms per event on a 1 GHz Pentium III. Preliminary studies have shown that time saved because of the lower combinatorics does not compensate the CPU time cost.

[12] Controlled by `$HLTSELECTIONSROOT/options/HLTselB2HH.opts`.
[13] Controlled by `$HLTSELECTIONSROOT/options/HLTselBs2DsH.opts`.
[14] Controlled by `$HLTSELECTIONSROOT/options/HLTselBs2PhiPhi.opts`.
[15] Controlled by `$HLTSELECTIONSROOT/options/HLTselBd2D0Kstar.opts`.
[16] Controlled by `$HLTSELECTIONSROOT/options/HLTselBd2DstarPi.opts`.
[17] Controlled by `$HLTSELECTIONSROOT/options/HLTselBd2MuMuKstar.opts`.
[18] Controlled by `$HLTSELECTIONSROOT/options/HLTselB2JpsiX.opts`.
[19] Controlled by `$HLTSELECTIONSROOT/options/HLTselB2MuMu.opts`.
[20] Controlled by `$HLTSELECTIONSROOT/options/HLTselDstar.opts`.
[21] Controlled by `$HLTSELECTIONSROOT/options/UseRich.opts`.

4.5 HLT Packages

Here we list the packages relevant for the HLT. They are in the PHYS project and available from the DAVINCI and MOORE applications.

Trg/TrgSys and used packages**:** All HLT reconstruction.

Hlt/HltSys: "Sys" package that defines a coherent set of HLT packages. This package is only available in the PHYS project in the DC04-incompatible releases of the software.

Hlt/HltGeneric: The code of the generic HLT and the muon highway.

Hlt/HltSelections: All options needed to run the HLT.

Hlt/HltSelChecker: Checking algorithms and options.

Hlt/HltMonitor: Algorithms that monitor the HltScore class and compare it to the raw buffer. This package is only available in the PHYS project in the DC04-incompatible releases of the software. It is mainly used for the RTTC.

The HltScore summary class is in the LHCB project in Event/TrgEvent.

5 Outlook and Further Developments

This prototype is the first complete attempt to use the online tracking to select signal and reject background down to 200 Hz. The details of the "physics" outcome can be read in a subsequent note [1]. Here we address a few technical problems that have been observed and set some requirements for the future.

5.1 Lessons

The main observation is that although it is possible to achieve a 200 Hz minimum-bias rate, we have not been able to get close enough to 100% efficiency for signal. The average efficiency is about 95% per track. There are several reasons:

- The online forward tracking and the offline tracking are too different. The correlation of the track finding efficiencies is too low. Ideally one would like to find online all tracks relevant for an offline selection.
- One reason for this is he VeloTT algorithm. To gain speed one requires that tracks found in the Velo are first extrapolated to TT before the full forward tracking is performed. Unfortunately this extrapolation has inefficiencies, partially due to the design of the TT stations. An improved VeloTT algorithm is now available.[22]
- The parameters of the tracks are not as good as for offline tracks causing abnormally high vertex χ^2 in some rare cases. This has yet to be understood.
- Many minimum-bias events passing the HLT have a different number of online and offline reconstructed primary vertices.
- Some official selections apply very loose IP and p_T requirements, which we think is not reasonable.

There are two fronts to overcome these problems: more similar online and offline pattern recognitions and more inclusive HLT selections. They are discussed in the next sections.

[22] It will be released in DAVINCI v12r14 in September 2005.

5.2 Requirements to Reconstruction

The on- and offline pattern recognitions should be as correlated as possible. As the analysis code which is the same for on- and offline selections, the pattern recognition should also share as much code as possible. This is presently under study under the new `Pat/` head.

Also the inefficiencies in TT should affect as little as possible the pattern recognition.

Finally, the efficiency for finding all primary vertices should be as high as possible, eventually at the price of some signal inefficiencies due to B vertices mis-identified as a PV.

5.3 Going Inclusive

We have to be prepared for imperfect efficiencies and need thus to think about a backup solution.[23] It is desirable to have several chances to select one event, one being the exclusive selection of the B and the other could be based on a more inclusive selection. This is already the case for all J/ψ and D^* channels.

One can thus think about an inclusive selection for ϕ, D_s, D and dileptons with a high-p_T and a high IP, which would be trivial to implement using the present prototype.

There are also studies about allowing one track to be only reconstructed as a Velo track, with a "guessed" momentum. Unfortunately such algorithms are quite CPU-time consuming.

More generally, "on-demand" reconstruction tools would be desirable. For instance such a tool would allow to extrapolate a Velo track of interest to TT and T.

5.4 What about a 1 MHz Readout?

The present HLT has been designed to run after L1. Originally it was even foreseen to have a quite separate generic layer before all exclusive selections are run. It appeared that everything related to muons had to be handled by `HltGeneric` [3]. This allowed to recover efficiencies by reconstructing the muon candidates without using TT.

With a 1 MHz running scenario we expect similar developments. The present L1 would be run at the beginning, but could profit from more information from the muon detector and maybe the T stations. All duplications between L1 and HLT (Velo tracking, primary vertex finding...) could be removed and the generic part of HLT would slowly merge with L1.

6 Conclusion

A first fully functional High-Level-Trigger prototype is ready and released both in the DC04- and RTTC-compatible software suites and has been run successfully in the RTTC.

It allows to easily add new exclusive and inclusive selections which is one of the high priority task of the Physics Task Forces.

The development and testing of this new prototype triggered developments in the analysis code that will profit also to the stripping and the offline selections. On the other hand, it emphasized some weaknesses of the reconstruction that will hopefully be cured with the next iteration in the development of the pattern recognition code.

[23] That's the legacy from HERA-B: an `or` of hard requirements is better than an `and` of loose requirements.

Appendix

A How to Access the Trigger Decision

This Appendix explains how to run the HLT, how to retrieve the decision and how to access the summary. How to extend the HLT is explained in Appendix B.

Everything which is in the Appendix is copied from `http://lhcb-trig.web.cern.ch/lhcb-trig/HLT/Default.htm`. The text below may be more detailed, but the up-to-date information is the one on the web.

A.1 Run the Trigger

The minimal options to run the HLT before your analysis are:

```
#include "$HLTGENERICROOT/options/HltGeneric.opts"
#include "$HLTSELECTIONSROOT/options/HLTSelections.opts"
ApplicationMgr.TopAlg += { "ResetOnOffline/SetOffline" };
SetOffline.ResetToOffline = true ;
```

The HLT is run by including the options for the generic HLT (`$HLTGENERICROOT/options/HltGeneric.opts`) and the exclusive selections `$HLTSELECTIONSROOT/options/HLTSelections.opts`). The latter also fills the decision class. The inclusion order of these two files is actually irrelevant.

The HLT does not run when the generic part is not executed. But one can ignore the decision of the latter with

```
HltGeneric.Decision = false ; // to ignore HltGeneric decision
```

A.2 Warning: Make sure you run in On- or Offline Mode!

The HLT resets the default location of the primary vertex to the HLT PV.[24] It also redefines the vertexing and extrapolation tools. If you plan to run an offline selection after the HLT, you have to reset to offline mode. The knowledge of the "context" is in the `OnOfflineTool`. It can be switched back to offline by adding the following line to your options after the HLT execution:

```
ApplicationMgr.TopAlg += { "ResetOnOffline/SetOffline" } ; // Offline.
SetOffline.ResetToOffline = true ;
```

If you don't do this you will get a poorer vertexing accuracy, and will get the primary vertex computed by the HLT.

To get the HLT behaviour:

```
ApplicationMgr.TopAlg += { "ResetOnOffline/SetOnline" } ; // Online.
SetOnline.ResetToOnline = true ;
```

This is already done in the HLT options.

This "hack" is considered as temporary. It is needed because an instance of the `MakeResonances` algorithm has no way to know if it has been called in an online or offline context. In the future this information could be obtained from the newly added "context" flag of `GaudiAlgorithm`.

[24] This has changed with DaVinci v12r12. If you are using an older version (why?) have a look at `http://lhcb-trig.web.cern.ch/lhcb-trig/HLT/PVLocator.htm`.

A.3 Retrieve the Information

There is one global HLT decision written out in the `TrgDecision` class. The method is called `hlt()` in DaVinci v13 and more and `HLT()` DaVinci v12. This is a little unfortunate, but we had to change the inconsistently called method `HLT()` to the standard lowercase someday. Typically one would get it with:

```
#include "Event/TrgDecision.h"

bool hlt = false ;
if ( exist<TrgDecision>(TrgDecisionLocation::Default){
  TrgDecision* trg = get<TrgDecision> (TrgDecisionLocation::Default);
  hlt = trg->hlt() ; // or HLT() in DaVinci v12
}
```

A.4 The `HltScore` Class

If you want more detail, you need the `HltScore` class. It contains the decisions of all the four streams of the HLT and the decision of the generic HLT.

```
#include "Event/HltScore.h"

if ( exist<HltScore>(HltScoreLocation::Default){
  HltScore* hlt = get<HltScore> (HltScoreLocation::Default);
  debug() << "Hlt decision is    " << hlt->decision() << "\n"
          << "Inclusive b->mu  : " << hlt->decisionInclusiveB()  << "\n"
          << "Inclusive Dimuon : " << hlt->decisionDimuon )  << "\n"
          << "Inclusive D*     : " << hlt->decisionDstar() << "\n"
          << "Exclusive B      : " << hlt->decisionExclusive() << "\n"
          << "Generic HLT      : " << hlt->decisionGen() << endmsg ;
}
```

The global decision of the HLT, which is also the one filled in the `TrgDecision` class, is defined as

```
decision = (( decisionGen())       &&   // generic
            ( decisionInclusiveB() ||   // inclusive B->mu
              decisionDimuon()     ||   // dimuon
              decisionDstar()      ||   // D*
              decisionExclusive() ));   // exclusive B
```

There are also some statistics available in `HltScore`, like the number of primaries (`nbPV()`), tracks (`nbTrack()`) or B (`nbB()`) found. Have a look at the `Doxygen` reference for a complete and up-to-date documentation.

Finally, there is a container of candidates with their mass, χ^2 and the ID of the algorithm that found them. The relation between the algorithm name and its ID is set by the `IAlgorithm2ID` tool that is interfaced by the `algorithmID()` method of `DVAlgorithm`. See the example code below:

```
const std::vector<HltAlgorithmScore>* algoScores = hlt->candidates() ;
for ( std::vector<HltAlgorithmScore>::const_iterator ias =
      algoScores->begin() ; ias != algoScores->end() ; ++ias ){
  std::string name = algorithmID()->nameFromId(ias->algorithmID());
  debug() << name << " (" << ias->algorithmID() << ")"
          << ", mass = " << ias->mass()
          << ", chi2 = " << ias->chi2() << endmsg ;
}
```

The fields `mass()` and `chi2()` are only filled when relevant. If not the value -1 is returned.

A.5 Some Monitoring and Checking

The main example option file for the HLT and all available tests and options is `$HLTSELCHECKERROOT/options/DVHLTCorr.opts`. It also suggests how to run TDR selections and to compare the efficiencies. One can for instance produce plots (include `$HLTSELCHECKERROOT/options/Plots.opts`) for all generic algorithms [2] (not in DaVinci v12r12 because of a bug), measure the timing (include `$HLTSELCHECKERROOT/options/MeasureTimes.opts`) and test the `HltScore` class (include `$HLTSELCHECKERROOT/options/TestHltScore.opts`). You can also fill in n-tuples for the offline and online selections in order to study the different cuts; this can be done with the options in `$HLTSELCHECKERROOT/options/DVHLTNTuplesCheck.opts`.

When you include include `$HLTSELCHECKERROOT/options/Correlations.opts` (requires access to MC truth) you get efficiency correlation tables for all core channels. But to just get the correlations between the various HLT streams add:

```
HltDecision.MakeCorrelations = true ; // Correlation table
```

This will produce a correlation table like the following

	Algorithm	Eff.	1	2	3	4	5	6	7	8	9	10	11	13	14
1	HltGeneric	21.16%	#####	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
2	InclusiveB	2.04%	9.6%	#####	8.2%	4.1%	10.8%	0.0%	2.7%	0.0%	7.1%	0.0%	0.0%	0.0%	10.5%
3	Dimuon	1.25%	5.9%	5.0%	#####	0.6%	22.1%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	100.0%	100.0%
4	HLTFilterDstar	0.44%	2.1%	0.8%	0.2%	#####	2.5%	5.2%	21.6%	25.0%	14.2%	0.0%	16.6%	0.0%	0.0%
5	ExclusiveB	5.60%	26.4%	29.6%	99.0%	32.1%	#####	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
6	HLTFilterB2HH	0.05%	0.2%	0.0%	0.0%	0.6%	1.0%	#####	5.4%	4.1%	3.5%	0.0%	0.0%	0.0%	0.0%
7	HLTFilterBd2D0Kstar	0.11%	0.5%	0.1%	0.0%	5.4%	2.0%	10.5%	#####	33.3%	32.1%	0.0%	0.0%	0.0%	0.0%
8	HLTFilterBd2DstarPi	0.07%	0.3%	0.0%	0.0%	4.1%	1.3%	5.2%	21.6%	#####	28.5%	0.0%	0.0%	0.0%	0.0%
9	HLTFilterBs2DsH	0.08%	0.4%	0.2%	0.0%	2.7%	1.5%	5.2%	24.3%	33.3%	#####	0.0%	0.0%	0.0%	0.0%
10	HLTFilterBs2PhiPhi	0.00%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	#####	0.0%	0.0%	0.0%
11	HLTFilterBs2PhiGamma	0.01%	0.0%	0.0%	0.0%	0.6%	0.3%	0.0%	0.0%	0.0%	0.0%	0.0%	#####	0.0%	0.0%
13	HLTFilterBd2MuMuKstar	0.00%	0.0%	0.0%	0.4%	0.0%	0.1%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	#####	10.5%
14	HLTFilterB2JpsiX	0.05%	0.2%	0.2%	4.6%	0.0%	1.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	100.0%	#####

The absolute efficiency of each algorithm is given in the "Eff." column. In the rest of the table, each entry in line indexed by L and column indexed by C gives the efficiency for algorithm L given that algorithm C found something. For example the efficiency of `HLTFilterBd2DstarPi` ($L = 8$) is 4.1% when a D^* was found by `HLTFilterDstar` ($C = 4$).

A.6 RICH

To use the RICH in the HLT simply include:

```
#include "$HLTSELECTIONSROOT/options/UseRich.opts" // to use RICH in HLT
```

This will only affect the making of the kaons. The kaon-versus-π-ID delta-log-likelihood cut can be tuned with:

```
TrgParticleMaker.KaonDLLCut = 15 ; // DLL cut applied to make kaons in HLT
```

B How to Add a Selection

In this Appendix we explain how to add a selection to the HLT. We take the example of the $B \to \ell\ell K$ selection that can be found in the recent versions of the `PhysSel/Bu2LLK` package.

The whole selection is based on standard algorithms and tools and hence there is not a single line of C++ to write.

B.1 General Structure of the Selection

In our example we will write an option file called `DVHLTBu2LLK.opts` that looks like:

```
//--------------------------------------------------------------
#pragma print off
#include "$DAVINCIROOT/options/DaVinciCommon.opts"
#pragma print on
//--------------------------------------------------------------
// Re-run L0 Trigger with DC04 tuning
#include "$DAVINCIROOT/options/L0.opts"
//--------------------------------------------------------------
// Run L1 Trigger
#include "$TRGSYSROOT/options/L1.opts"
//--------------------------------------------------------------
// Run HLT
#include "$HLTGENERICROOT/options/HltGeneric.opts"
#include "$HLTSELECTIONSROOT/options/HLTSelections.opts"
//--------------------------------------------------------------
// Select Bu->LLK: Main HLT selection
#include "$BU2LLKROOT/options/HLTselBu2LLK.opts"
//--------------------------------------------------------------
// Offline: Run DC04
//#include "$BU2LLKROOT/options/HLTOfflineBu2LLK.opts"
//--------------------------------------------------------------
// Truth
//#include "$BU2LLKROOT/options/HLTTrueBu2LLK.opts"
// Correlations
//#include "$BU2LLKROOT/options/HLTCorrBu2LLK.opts"
//--------------------------------------------------------------
// Plots
//#include "$BU2LLKROOT/options/HLTBu2LLKPlots.opts"
```

Only the ”Main HLT selection” `HLTselBu2LLK.opts` is mandatory, the options files below are used for monitoring and checking purposes.

B.2 Pre-selection of HLT Particles

Here we start to write the HLT selection file `HLTselBu2LLK.opts` for the channel B $\to \ell\ell$K. Rather than making the full combinatorics of B $\to \mu\mu$K and B $\to$ eeK, we will start making the dileptons first as they are much more rare. The strategy in the HLT is to re-use as much as possible already existing particles. In this case we already have the dimuon from the B $\to \mu\mu$K* decay (actually from the generic HLT), we are thus missing the ee dilepton.

Before you start make sure you have a copy of note [2].

First let's make sure we have the electrons available and copy the electrons from `Phys/Trg` to `Phys/HLTElectrons`. This is of course only necessary for not already provided particles. Pions, kaons and photons are already there and muons are provided as dimuons. If you use one of these, you can skip this section.

```
// copy all electrons (TrgTracks.opts)
//-------------------------------------------------------------
CopyParticles.Members += { "PIDFilter/HLTElectrons" };
HLTElectrons.PhysDesktop.InputLocations = { "Phys/Trg" } ;
HLTElectrons.ParticleNames = { "e+", "e-" };
```

Then in a second step let's apply some loose cuts:

```
// Apply some pre-selection cuts on electrons (HLTFilterParticles.opts)
//-------------------------------------------------------------------------
HLTFilterParticles.Members += {"FilterDesktop/HLTPreselElectrons"};
HLTPreselElectrons.PhysDesktop.InputLocations = {"Phys/HLTElectrons"};
HLTPreselElectrons.Filter.Selections = {"e+ : TrackTypeFilterCriterion,
                                           KinFilterCriterion,
                                           PVIPFilterCriterion"} ;
HLTPreselElectrons.Filter.KinFilterCriterion.MinPt = 300.*MeV;
HLTPreselElectrons.Filter.KinFilterCriterion.MinMomentum = 0.;
HLTPreselElectrons.Filter.PVIPFilterCriterion.MinIPsignif = 2.;
HLTPreselElectrons.Filter.TrackTypeFilterCriterion.RequireLong = true ;
HLTPreselElectrons.Filter.ExclusiveSelection = true ;
```

Here we apply a 300 MeV p_T cut and a 2σ IP significance. We also require the track to be long.

Of course these two steps could have been done in one, and in the real experiment they probably will be (or even taken care of by the `TrgParticleMaker`). But now it's handy to be able to refine the electrons adding some MC truth requirements (association to signal) in between the two algorithms, as will be seen later.

B.3 Intermediate States

The next step is to make dielectrons from electrons. People who want to use K*, ϕ, D^0, D$_s$ or dimuons can skip this section.

```
// Make dielectron
//--------------------------------------------------------------------
HLTMakeResonances.Members += {"MakeResonances/HLTSharedDiElectron"} ;
HLTSharedDiElectron.PhysDesktop.InputLocations =
                        {"Phys/HLTPreselElectrons"};
HLTSharedDiElectron.DecayDescriptor = "J/psi(1S) -> e+ e-" ;
HLTSharedDiElectron.LowerWindow = 3000.* MeV ;
HLTSharedDiElectron.UpperWindow = 2500.* MeV ;
HLTSharedDiElectron.MotherFilter.Selections =
                 { "J/psi(1S) : VtxFilterCriterion" };
HLTSharedDiElectron.MotherFilter.VtxFilterCriterion.MaxChi2 = 200;
```

There is no PID for "dielectron", so we call it "J/psi(1S)", but set an asymmetric mass window between $m_{J/\psi} - 3.0\,\text{GeV}$ and $m_{J/\psi} + 2.5\,\text{GeV}$, which is large enough. We also require a maximal χ^2 of 200.

B.4 Final State

Now we are done for the shared particles, we can start with the final states.

For the final selection one algorithm is enough to make $B \to \mu\mu K$ and $B \to eeK$. We actually build $B \to J/\psi K$ using both our dielectron and the dimuon from the generic HLT.

```
// HLT selection
//--------------------------------------------------------------------
HLTExclusive.Members += {"GaudiSequencer/HLTselBu2LLK"};

// Combine K and dilepton
HLTselBu2LLK.Members += {"MakeResonances/HLTFilterBu2LLK"};
HLTFilterBu2LLK.PhysDesktop.InputLocations =
          { "Phys/HltGeneric/JPsi", "Phys/HLTSharedDiElectron",
    "Phys/HLTPreselKaons"};
HLTFilterBu2LLK.DecayDescriptor = "[ B+ -> J/psi(1S) K+ ]cc";

HLTFilterBu2LLK.DaughterFilter.Selections =
                 { " K+ : KinFilterCriterion/KKin",
                   "mu+ : PVIPFilterCriterion/MuPVIP" };
HLTFilterBu2LLK.DaughterFilter.FilterByDescendents = true ;
HLTFilterBu2LLK.DaughterFilter.KKin.MinPt =  1000 * MeV ; // MeV
HLTFilterBu2LLK.DaughterFilter.MuPVIP.MinIPsignif = 0 ; // MeV
HLTFilterBu2LLK.Window = 500 * MeV; //

HLTFilterBu2LLK.MotherFilter.Selections =
           { " B+ : PVIPFilterCriterion, VtxFilterCriterion,
            FlightDistanceFilterCriterion" };

HLTFilterBu2LLK.MotherFilter.VtxFilterCriterion.MaxChi2 = 50 ;
HLTFilterBu2LLK.MotherFilter.PVIPFilterCriterion.MaxIPsignif = 5 ;
HLTFilterBu2LLK.MotherFilter.FlightDistanceFilterCriterion.MinFSPV =  5 ;

// This is just for a nice printout
HLTselBu2LLK.Members += {"PrintHeader/PassedHLTBu2LLK"};
PassedHLTBu2LLK.OutputLevel = 2 ;
```

Here we require a minimal p_T of 1 GeV for the kaon and a 2σ PV separation for the muons (as it has not yet been applied) from the "J/ψ" (note the option `FilterDescendents` to go down to the daughters). We apply a ± 500 MeV mass window on the J/ψK system and want it to have a χ^2 below 50, to be pointing at one PV within 5σ and to be separated from it by 5σ.

That's it for the selection!

B.5 Add your Selection to `HltScore`

The last step is to declare your selection to the HLT decision. For each candidate in the `HltScore` class, the algorithm that selected it is identified by a unique ID handled by the `Algorithm2ID` tool.

```
// HLT Decision
//=============================================================
HltDecision.ExclusiveAlgorithms += { "HLTFilterBu2LLK" };
// overwrite Algorithm2ID list
ToolSvc.Algorithm2ID.AlgorithmNames = { "HLTFilterDstar",
                      @HltDecision.ExclusiveAlgorithms  };
```

A the beginning of the job you will get the printout:

```
ToolSvc.Algorithm2ID  INFO The following algorithms get an ID:
ToolSvc.Algorithm2ID  INFO 0 : HLTFilterDstar
ToolSvc.Algorithm2ID  INFO 1 : HLTFilterB2HH
ToolSvc.Algorithm2ID  INFO 2 : HLTFilterBd2D0Kstar
ToolSvc.Algorithm2ID  INFO 3 : HLTFilterBd2DstarPi
ToolSvc.Algorithm2ID  INFO 4 : HLTFilterBs2DsH
ToolSvc.Algorithm2ID  INFO 5 : HLTFilterBs2PhiPhi
ToolSvc.Algorithm2ID  INFO 6 : HLTFilterBs2PhiGamma
ToolSvc.Algorithm2ID  INFO 7 : HLTFilterB2MuMu
ToolSvc.Algorithm2ID  INFO 8 : HLTFilterBd2MuMuKstar
ToolSvc.Algorithm2ID  INFO 9 : HLTFilterB2JpsiX
ToolSvc.Algorithm2ID  INFO 10 : HLTFilterBu2LLK
```

Hence we get ID = 10 for `HLTFilterBu2LLK`. But as mentioned above you can decode this ID to a string using `Algorithm2ID`.

This ends the mandatory part of the HLT options.

B.6 Offline Selections

To measure the correlation of the HLT selection with the offline selection, we need to run the latter. First switch the context from online to offline (in order to use the offline vertex fitter and geometrical calculator) and then paste whatever is needed to run your offline selection. In this case it's the DC04 selection for $B \to \ell\ell K$.

```
HLTSelections.Members += { "ResetOnOffline/SetOffline" };
SetOffline.ResetToOffline = true ;
#include "$DAVINCIROOT/options/DaVinciElectrons.opts"
#include "$DAVINCIROOT/options/DaVinciReco.opts"
#include "$DAVINCIROOT/options/DaVinciNeutrals.opts"
#include "$BU2LLKROOT/options/DoPreselBu2LLK.opts"
#include "$BU2LLKROOT/options/DoDC04selBu2LLK.opts"
DC04selBu2LLK.Window =  50 * MeV; // Small mass window
```

B.7 MC Truth

To check the efficiency on MC truth, one can do the following (file `HLTTrueBu2LLK.opts`):

```
//#include "$HLTSELCHECKERROOT/options/Truth.opts" // for all channels
// comment this line if you uncomment the line above:
TrueSignals.Members += {"TrgL1Filter/TrueTrgL1Filter",
         "TrgTrackToContainer","TrgTrack2MCParticle"};

TrueSignals.Members += {"FilterDesktop/SelectBu2LLK"};
SelectBu2LLK.PhysDesktop.InputLocations =
                { "Phys/HLTElectrons", "Phys/HLTKaons"} ;
SelectBu2LLK.Filter.Selections =  { "K+ : TrueMCFilterCriterion/Truth",
                                    "e+ : TrueMCFilterCriterion/Truth" };
SelectBu2LLK.Filter.Truth.MCDecayFinder.Decay =
            "{[B+-> ^K+ ^e+ ^e- {,gamma} {,gamma}]cc,
      [B+ -> ^K+ ^mu+ ^mu- {,gamma}{,gamma}]cc}" ;
SelectBu2LLK.Filter.Truth.Particle2MCLinksAsct.Location =
                          "Phys/Relations/Particle2MCLinksSelectBu2LLK" ;
SelectBu2LLK.Filter.Truth.Particle2MCLinksAsct.AlgorithmType =
                                                    "Particle2MCLinks" ;
SelectBu2LLK.Filter.Truth.Particle2MCLinksAsct.AlgorithmName =
                                         "Particle2MCLinksSelectBu2LLK" ;
SelectBu2LLK.HistoProduce = false ;

Particle2MCLinksSelectBu2LLK.InputData =
        { "Phys/HLTElectrons/Particles", "Phys/HLTKaons/Particles"};

// check that all tracks are there
TrueSignals.Members += {"MakeResonances/AllBu2LLKTracks"};
AllBu2LLKTracks.PhysDesktop.InputLocations = {"Phys/SelectBu2LLK"};
AllBu2LLKTracks.DecayDescriptor = "[B+ -> e+ e- K+]cc" ;
AllBu2LLKTracks.DaughterFilter.Selections = {
                            "e+ : TrackTypeFilterCriterion/TType",
                            "K+ : TrackTypeFilterCriterion/TType"};
AllBu2LLKTracks.DaughterFilter.TType.RequireLong = true ;

TrueSignals.MeasureTime = true ;
```

If you uncomment the first line you will get the checking of MC truth for all channels (which is quite slow) and you need to uncomment the second line to avoid duplications. The two algorithms `TrgTrackToContainer`, `TrgTrack2MCParticle` perform the association of `TrgTracks` to `MCParticles`. Then you can associate the particles to your signal using the `TrueMCFilterCriterion` tool. It only needs to know the decay string descriptor which you can find in your `EvtGen` decay file in `$LHCBRELEASES/DBASE/Gen/DecFiles/-.../dkfiles/` (with a hat ^ flag added for the required particles). Note that presently (DaVinci v12r12) the association of (online) muons does not work.

The algorithm `SelectBu2LLK` selects all particles made from your signal and `AllBu2LLKTracks` tries to build a B using only these particles. If it succeeds it means you have all particles you need. Hence the efficiency of this algorithm measures the total reconstruction efficiency for your decay. We will re-use this information later on.

You could also use the output of the algorithms to overwrite the `InputLocation` of `HLTPreselElectrons` and `HLTPreselKaons` which allows to run only on signal tracks.

B.8 Efficiency Correlations

For each selection in the HLT an efficiency correlation matrix is defined in `$HLTSELCHECKERROOT/options/Correlations.opts`. Let's add one more, but first let's define an algorithm that selects events where both the offline selection fires and all signal tracks are reconstructed. This is the normalization point for the selection efficiency. In an ideal world the HLT selection should be 100% efficient on these events. That's where we re-use our algorithm `AllBu2LLKTracks`.

```
#include "$HLTSELCHECKERROOT/options/Correlations.opts"
// check for DC04 and MC truth
HLTCorrelations.Members += { "CheckSelResult/Bu2LLKAndDC04" };
Bu2LLKAndDC04.ANDmode = true ;
// require offline and all tracks:
Bu2LLKAndDC04.Algorithms = { "DC04selBu2LLK" , "AllBu2LLKTracks" } ;
Bu2LLKAndDC04.AvoidSelResult = false ;
```

Then we can use the `SelResultCorrelations` algorithm to produce a correlation table:

```
// correlation of HLT and DC04
HLTCorrelations.Members += { "SelResultCorrelations/CorrBu2LLK" };
CorrBu2LLK.Algorithms = { "CheckHLTGeneric",
                          "CheckHLTDimuon",
                          "HLTElectrons",
                          "HLTPreselElectrons",
                          "HLTSharedDiElectron",
                          "HLTFilterBu2LLK",
                          "AllBu2LLKTracks",
                          "DC04selBu2LLK",
                          "Bu2LLKAndDC04" } ;
```

At the end of the job it will print a correlation table of the efficiencies of all declared algorithms.

B.9 Plots

`MakeResonances` (not in DaVinci v12r12 because of a bug) and `FilterDesktop` allow to make plots using the `RecursivePlotTool`. You can plot a few standard variables this way:

```
// Shared DiElectron
HLTSharedDiElectron.HistoProduce = true ;
HLTSharedDiElectron.DaughterPlots.Variables = { "P", "Pt", "IPs" } ;
HLTSharedDiElectron.MotherPlots.Variables =
                 { "WM", "Chi2", "Vz", "P", "Pt", "IPs", "FS" } ;
HLTSharedDiElectron.DaughterPlotsPath  = "DiElectronIn" ;
HLTSharedDiElectron.MotherPlotsPath = "DiElectronOut";

// Bu->LLK
HLTFilterBu2LLK.HistoProduce = true ;
HLTFilterBu2LLK.DaughterPlots.Variables =
                 { "WM", "Chi2", "Vz", "P", "Pt", "IPs" } ;
HLTFilterBu2LLK.MotherPlots.Variables =
                 { "M", "WM", "Chi2", "Vz", "P", "Pt", "IPs", "FS" } ;
HLTFilterBu2LLK.DaughterPlotsPath  = "Bu2LLKIn" ;
HLTFilterBu2LLK.MotherPlotsPath = "Bu2LLKOut";
```

The name of the plotting variables is probably going to change someday (for DC06). We would like to make it coherent with the name of the LoKi functors.

C Full HLT sequence

The following listing shows the full HLT sequence as printed out by the timing tool of the GaudiSequencer.

```
Hlt                          // HLT sequence
  TrgL1Filter                // Check that L1 passed
  HLTProcessing              // HLT sequence (now without L1 checking)
    HLTVeloTracking          // HLT velo tracking
      DecodeRawVelo
      VeloRTracking
      VeloSpaceTracking
      TrgPrimVertexFinder
    HLTFullTracking          // HLT full tracking. This is in principle not needed
      DecodeTTFromRaw        //                    and should be done only by the generic part
      TrgVeloTT
      FwdHitFromRawBuffer
      TrgForwardAlg
      TrgInsertTrackErrParam
    HLTGenericSeq            // HLT Generic sequence
      TrgMuonRec
        Load from Buffer
        Make pad from strip
        Reconstruct muons
        Associate muons
      HltGenericSelection
      GenMakeTTHit
      GenVeloTT
      TrgMuonRecover
      GenMakeTHit
      GenForward
      TrgMuonRefine
      GenErrParam
      GenParticleMaker
      HltGenericDecision     // If generic no, stop here
    HLTParticles             // Making of particles
      TrgMuonID
```

```
      TrgRICH4HLTSeq
      TrgCaloReco
      TrgParticleMaker
    HLTSelections               // Start of HLT selections
      SetOnline                 // Set to online mode
      CheckPV                   // Check that PV is there
      CopyParticles             // Particles are copied (cloned) to some specific locations.
        HLTPions                // Pions will be in /Event/Phys/HLTPions
        HLTKaons                // Kaons will be in /Event/Phys/HLTKaons
        HLTPhotons              // Photons will be in /Event/Phys/HLTPhotons
      TrueSignals               // Checking of MC truth. Empty by default
        [...]                   // Here will run the association to MC truth if requested
      HLTFilterParticles        // Filter particles
        HLTDetachedPions        // Select high IP pions
        HLTPreselPions          // Select high Pt pions among high IP pion s
        HLTPreselKaons          // Select high IP and Pt kaons
        HLTPreselPhotons        // Select High PT photons
     HLTMakeResonances          // Make shared intermediate states
        HLTSharedKstar          // Make K* -> K Pi using HLTPreselPions and HLTPreselKaons
        HLTSharedPhi            // Make Phi -> KK using HLTPreselKaons
        HLTSharedD02KH          // Make D0 -> KK and D0 -> K Pi using HLTPreselPions and HLTPreselKaons
        HLTLooseD02PiPi         // Make D0 -> PiPi using HLTPreselPions with no mass cut
        HLTSharedDs             // Make Ds -> K K Pi using HLTPreselPions and HLTPreselKaons
      HLTselDstar               // D* stream selection
        HLTFilterDstar          // Make D* from HLTLooseD02PiPi and HLTDetachedPions
      HLTExclusive              // HLT Exclusive B stream.
        HLTselB2HH              // B -> H H selection
          HLTFilterB2HH         // Make B -> Pi Pi using HLTPreselPions
        HLTselBs2PhiPhi         // B ->  selection
          HLTFilterBs2PhiPhi    // Make Bs -> PhiPhi using HLTSharedPhi
        HLTselBd2D0Kstar        // B -> D0 K* selection
          FilterD0ForHLTBd2D0K  // Refine D0 from HLTSharedD02KH
          FilterKstarForHLTBd2  // Refine K* from HLTSharedKstar
          HLTFilterBd2D0Kstar   // Make B -> D0 K*
        HLTselBd2DstarPi        // B -> D* Pi selection
          FilterD0ForHLTBd2Dst  // Refine D0 from HLTSharedD02KH
          HLTDstarForBd2DstarP  // Make D* -> D0 Pi using HLTPreselPions and FilterD0ForHLTBd2Dst
          HLTFilterBd2DstarPi   // Make B -> D* Pi
        HLTselBs2DsH            // Bs -> Ds H selection
          HLTFilterBs2DsH       // Make B -> Ds Pi using HLTPreselPions and HLTSharedDs
        HLTselBs2PhiGamma       // Bs -> Phi Gamma selection
          HLTFilterBs2PhiGamma  // Make Bs -> Phi Gamma using HLTPreselPhotons and HLTSharedPhi
        HLTselB2MuMu            // B -> Mu Mu selection
          HLTB2MuMu             // Rename J/psi -> Mu Mu from HltGeneric into B -> Mu Mu
          HLTFilterB2MuMu       // Refine B -> Mu Mu
        HLTselBd2MuMuKstar      // B -> Mu Mu K* selection
          HLTFilterBd2MuMuKsta  // Make B -> Mu Mu K* using HltGeneric and HLTSharedKstar
        HLTselB2JpsiX           // B -> J/psi X selection
          HLTFilterB2JpsiX      // Refine J/psi from HltGeneric
HltDecision                     // Fill HltScore
```

`HltDecision` is always run as `TopAlg` after the sequence.

References

[1] L. Fernández and P. Koppenburg. Exclusive HLT Performance. LHCb-2005-047, LPHE-2005-015, 2005.

[2] P. Koppenburg and L. Fernández. DaVinci for Busy People. LHCb-2005-016, LPHE-2005-012, 2005.

[3] JOSE-ANGEL HERNANDO. Generic HLT. Unpublished, 2005.

[4] OLIVIER CALLOT. Online Pattern Recognition. LHCb-2004-094, 2004.

[5] HUGO RUIZ. Parameterization of track uncertainties for the HLT. LHCb-2005-012, 2005.

[6] HUGO RUIZ. Fast tools for vertexing and geometry calculations for the HLT. LHCb-2005-013, 2005.