LHCb StarterKit 2024: Practice session *

DaVinci Run 3

StarterKit Organizers

November 2024

This is the version with solutions. Solutions are marked in this font.

Goals of the session

After a long week of lessons, having headaches after using the software of our beloved collaboration, and having fun at yesterday's social event, now is time to shine :). In this session, we are going to put into practice all the knowledge acquired during this week with some realistic examples of how to perform analysis at LHCb.

For this first part, we are going to focus on something really important, which is **tupling**. Independently of the physics analysis you do, the first step is always to check your data samples and create the famous ntuples. So, this morning we are going to create ntuples from scratch in Run 3. This time, we are all together doing the exercise, so do not hesitate to ask around, for the next time it will be you on your own.

Now, let's get started!

1 Create your own GitLab repository

First step: let's create a repository where we will store our scripts! As a rule in LHCb, everyone has to preserve the code to run the analysis, including the production of the ROOT files. You must:

- Create a new repository on your personal GitLab.
- As you write the scripts, regularly commit them.
- When the scripts are ready, add a little `README` explaining how to run them, which software versions to use, and other necessary instructions to facilitate reproducibility.

2 Introduction

Today we are going to study the decays $D^+_{(s)} \to \phi\pi^+$ decays[1].

Before we start, let's discuss some physics questions about this decay.

- Which type of the transition is this decay? (Strong, weak, electromagnetic)? ***Weak.***

*Found a mistake in this document? Please contact lhcb-starterkit-organisers@cern.ch

[1]The $D^+_{(s)}$ stands for both D^+ and D^+_s mesons

- Which of the final state particles are stable?

  Pion.

- If there are unstable particles, which decay mode(s) could be used to reconstruct them?

  The ϕ resonance decays mainly to two kaons, either neutral or charged, but in LHCb we can only reconstructed the charged one. It may also decay into lighter unflavoured resonances and to pions, but also into a dilepton pair.

- Are there unstable particles flying measurable distance before the decay?

  The charm meson $D^+_{(s)}$ flies before decaying.

- Why is it interesting to study such a decay mode?

  There are many reasons, one can perform a CP violation analysis. They are also used as control modes to study other decays with similar topology, like $D^+_{(s)} \to \pi^+\ell^+\ell^-$.

Even thought the kaon decay of the ϕ resonance ahs higher BF, today we will consider ϕ decaying to $\mu^+\mu^-$ (we don't search for easy ways!).

2.1 Dataset and trigger lines

When you start an analysis, it's important to know how the data that will be relevant for you was recorded and, in particular, whether it went to the Turbo stream or sprucing. The first step to get this information is to know which working group your analysis belongs to. Once you do know it, you may reach out to the corresponding RTA/DPA liaisons. In this case, because of the presence of the charm meson, the HLT2 lines that select these decays belong to the Charm Working Group. The decay is stored through the Turbo stream.

Next, we need to find a possible HLT2 line to analyze our decay. Note that even if there is not a specific trigger line to your specific case, there are some procedures to work around it.

- Your task now is to find your trigger line. For that, go the HLT2 lines reference `https://lhcbdoc.web.cern.ch/lhcbdoc/moore/master/selection/hlt2_lines_reference.html` and check the `Charm` stream. **Hint**: The line should start with `Hlt2Charm_DpDsp`.

  The line is `Hlt2Charm_DpDspToPipMumMup`.

- Once you have found it, check the `DecayDescriptor` of the `ParticleContainerMerger` to make sure we are in the right place. Does it contain our decay of interest?

  The decay descriptor is `[D+ -> J/psi(1S) pi+]CC` ***. There is no*** $D^+_{(s)} \to \phi(\to \mu^+\mu^-)\pi^+$***, because intermediate particles have symbolic names.***

- Check if there is any decay of the intermediate resonance that is relevant for us.

  The `TwoBodyCombiner: Charm_RareCharm_MuMuPair` ***has a*** $J/\psi \to \mu^+\mu^-$ ***decay descriptor which leads us to our final state.***

- Finally, you can check the Moore code of the line to check which cuts are made in the candidates.

2.2 Simulation

When doing analysis in High Energy Physics, the use of simulations is essential, and this has no exception. It is important to find out if there are Monte Carlo samples of your decay. In the simpler cases, you can simply check the BookKeeping. If you have any doubt, just ask your WG's MC liaisons. In this case, we can confirm that there are official 2024 MC samples for the $D^+_s \to \phi(\to \mu^+\mu^-)\pi^+$ decay.

- To find the available samples in the Bookkeeping, the `EventType` of the simulation is going to be important. Find it in the decfile reference `http://lhcbdoc.web.cern.ch/lhcbdoc/decfiles/`. In this lesson we will only have simulation of the channel $D_s^+ \to \phi\pi^+$. *Hint*: The name should have this structure: `Ds...,mm=FromD.dec`

  The EventType is 23173003, with name Ds_phipi,mm=FromD.dec

3 Finding your data in the Bookkeeping

The first step is always checking where our data and simulations are located on the Grid. For that, we use the Bookkeeping.

- First we are going to find our data samples. Look for 2024 samples with all sub-detectors included. Remember that we are using the `Turbo` stream, and the `Sprucing24c3` version.
- Next, we should go for the Monte Carlo samples. Use the `EventType` you got before and find the samples. **Note**: There are going to be multiple samples inside the decay folder. Choose the one with `W` in the name. In these, the HLT2 reconstruction was processed.
- Get the `CondDBtag` and `DDDBtag` of your simulation in order to add them in the `yaml` file options. Remember that you'll have to run MC in `detdesc` platforms!

  We can know the compatible platforms with a given DaVinci version by typing `lb-run --list-platforms DaVinci/v64r13` ***We are happy to use*** `x86_64_v2-el9-gcc13+detdesc-opt`.
- **Advanced**: From the `.py` with the `.dst` files, get a subsample of LFNs and create a catalog with them.

  The MC DSTs we are interested for today are located in this folder:

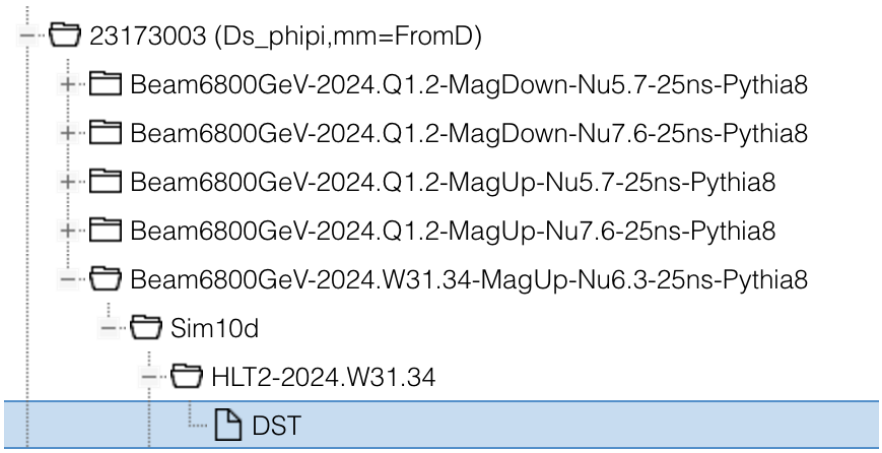

Figure 1: Location of the simulation in the bookkeeping.

4 Minimal DaVinci Job

Now it is time to start playing around with DaVinci. We need to choose a release of DaVinci, which will in general be the most recent one, unless there is a strong reason not to do so. You can check the list of DaVinci releases here: `https://lhcbdoc.web.cern.ch/lhcbdoc/davinci/releases/`, so we will use `DaVinci/v64r13`. First, we are going to write the most basic script we can do, the so-called `minimal`. Ideally,

in a real analysis the DaVinci script should be compatible to run both real data and simulation, and that's what we'll do here.

- Create a DaVinci options file with a `main` function defined.
- Define your input data using the `get_particles` function. Define also the corresponding HLT2 filter (`create_lines_filter`).
- Define the fields dictionary. Remember that the `DecayDescriptor` should match the one you found before in the HLT2 line reference.
- Create a `FunctorCollection` with the Functors mass, η, p_{T} and ϕ in order to retrieve this information from all the particles of the decay.
- Create the `FunTuple` and finish the DaVinci script.
- Develop two `.yaml` files, one for data and one for simulation, considering its differences. Use the data and Monte Carlo samples as well as the CondDB and DDDB tags you found before if you run within a `detdesc` platform, or `geometry` and `conditions` version if you run in `dd4hep`.
- Run DaVinci over data and simulation until you get candidates in your ntuples (Around 50k in data). Use the following command

```
lb-run DaVinci/v64r13 lbexec options_file:main options.yaml
```

For that, import the following objects:

```
from DaVinci import make_config, Options

from DaVinci.algorithms import create_lines_filter

from FunTuple import FunTuple_Particles as Funtuple

from PyConf.reading import get_particles

from FunTuple import FunctorCollection

import Functors as F
```

In case you didn't manage to get the data from the bookkeeping here there is a list of `DSTs` you can use to run your DaVinci jobs:

- data files

```
-root://x509up_u585@eoslhcb.cern.ch//eos/lhcb/grid/prod/lhcb/LHCb/Collision24/CHARM.
  DST/00243718/0000/00243718_00000891_1.charm.dst
- root://x509up_u585@eoslhcb.cern.ch//eos/lhcb/grid/prod/lhcb/LHCb/Collision24/CHARM.
  DST/00243718/0000/00243718_00000947_1.charm.dst
- root://x509up_u585@eoslhcb.cern.ch//eos/lhcb/grid/prod/lhcb/LHCb/Collision24/CHARM.
  DST/00243718/0000/00243718_00000948_1.charm.dst
- root://x509up_u585@eoslhcb.cern.ch//eos/lhcb/grid/prod/lhcb/LHCb/Collision24/CHARM.
  DST/00243718/0000/00243718_00001791_1.charm.dst
- root://x509up_u585@eoslhcb.cern.ch//eos/lhcb/grid/prod/lhcb/LHCb/Collision24/CHARM.
  DST/00243718/0000/00243718_00002088_1.charm.dst
- root://x509up_u585@eoslhcb.cern.ch//eos/lhcb/grid/prod/lhcb/LHCb/Collision24/CHARM.
  DST/00243718/0000/00243718_00002559_1.charm.dst
- root://x509up_u585@eoslhcb.cern.ch//eos/lhcb/grid/prod/lhcb/LHCb/Collision24/CHARM.
  DST/00243718/0000/00243718_00003657_1.charm.dst
- root://x509up_u585@eoslhcb.cern.ch//eos/lhcb/grid/prod/lhcb/LHCb/Collision24/CHARM.
  DST/00243718/0000/00243718_00003838_1.charm.dst
- root://x509up_u585@eoslhcb.cern.ch//eos/lhcb/grid/prod/lhcb/LHCb/Collision24/CHARM.
  DST/00243718/0000/00243718_00003846_1.charm.dst
- root://x509up_u585@eoslhcb.cern.ch//eos/lhcb/grid/prod/lhcb/LHCb/Collision24/CHARM.
  DST/00243718/0000/00243718_00004566_1.charm.dst
```

```
  - root://x509up_u585@eoslhcb.cern.ch//eos/lhcb/grid/prod/lhcb/LHCb/Collision24/CHARM.
    DST/00243718/0000/00243718_00004595_1.charm.dst
  - root://x509up_u585@eoslhcb.cern.ch//eos/lhcb/grid/prod/lhcb/LHCb/Collision24/CHARM.
    DST/00243718/0000/00243718_00004637_1.charm.dst
  - root://x509up_u585@eoslhcb.cern.ch//eos/lhcb/grid/prod/lhcb/LHCb/Collision24/CHARM.
    DST/00243718/0000/00243718_00004904_1.charm.dst

```

- MC files

```
   - root://x509up_u147315@eoslhcb.cern.ch//eos/lhcb/grid/prod/lhcb/MC/2024/DST
    /00251770/0000/00251770_00000001_1.dst
   - root://x509up_u147315@eoslhcb.cern.ch//eos/lhcb/grid/prod/lhcb/MC/2024/DST
    /00251770/0000/00251770_00000008_1.dst
   - root://x509up_u147315@eoslhcb.cern.ch//eos/lhcb/grid/prod/lhcb/MC/2024/DST
    /00251770/0000/00251770_00000009_1.dst
```

- condDBtag = `dddb-20240427` and DDDBtag = `sim10-2024.Q3.4-v1.3-mu100`

```
def main(options: Options):
    Minimal DaVinci Job + fiels
    line = f"Hlt2Charm_DpDspToPipMumMup"

    data = get_particles(f"/Event/HLT2/{line}/Particles")
    prefilt = create_lines_filter(name=f"PreFilter_{line}", lines=[line])

    fields = {"D":   "[D+ -> (J/psi(1S) -> mu- mu+) pi+]CC"}

    basicinfo = FunctorCollection({
        'M':F.MASS,
        'ETA':F.ETA,
        'PHI':F.PHI
    })

    features = {"ALL": global_features}

    dtt = Funtuple(name="D2pimumuTuple",
                   tuple_name="DecayTree",
                   fields=fields,
                   variables=features,
                   inputs=data,
                   event_variables=evt_info)

    alg = {line: [prefilt, dtt]}

    return make_config(options, alg)
```

The YAML file would look like (MC):

```
input_raw_format: 0.5
input_type: ROOT
output_type: ROOT
input_process: Hlt2
input_stream: charm
simulation: True
data_type: "Upgrade"
```

```
lumi: False
input_files:
  - root://x509up_u147315@eoslhcb.cern.ch//eos/lhcb/grid/prod/lhcb/MC/2024/DST/00251770/0000/002517
print_freq: 1000
evt_max: 8000
ntuple_file: Ds2mumupi_MC_MagDown_2024_tuple.root
dddb_tag: dddb-20240427
conddb_tag: sim10-2024.Q3.4-v1.3-mu100
```

And for real data processing:

```
input_raw_format: 0.5
input_type: ROOT
output_type: ROOT
input_process: TurboPass
input_stream: charm
simulation: False
data_type: "Upgrade"
geometry_version: run3/2024.Q1.2-v00.00
conditions_version: master
scheduler_legacy_mode: False
input_files:
  - root://x509up_u585@eoslhcb.cern.ch//eos/lhcb/grid/prod/lhcb/LHCb/Collision24/CHARM.DST/00243718/
print_freq: 1000
evt_max: 50000
ntuple_file: D2pimumu_turbo_sprucing24c3_pp_MagDown_2024_tuple.root
```

5 Adding fields

Besides the information of the head particle of the decay we will usually also need the information of the rest of particles to perform a proper analysis. This means we need to add branches to our ntuples.

- Add a field for each daughter of the $D^+_{(s)}$: the ϕ and the π^+.
- Do it also for the muons of the $\phi \to \mu^+\mu^-$ decay.
- Check that everything worked properly.

You should update the fields dictionary to:

```
fields = {"D":   "[D+ -> (J/psi(1S) -> mu- mu+) pi+]CC",
          "phi": "[D+ -> ^(J/psi(1S) -> mu- mu+) pi+]CC",
          "mum": "[D+ -> (J/psi(1S) -> ^mu- mu+) pi+]CC",
          "mup": "[D+ -> (J/psi(1S) -> mu- ^mu+) pi+]CC",
          "pi":  "[D+ -> (J/psi(1S) -> mu- mu+) ^pi+]CC"}
```

6 Fun with Functors

Once the minimal DaVinci Job is done, the next step is to fill our ntuples with all the branches we need for the analysis. Now, in Run 3 we run with much higher instantaneous luminosity, which makes the amount of data taken much bigger than in Run 1 and 2. For this reason, the choice of functors added to the ntuple must be made in an optimal way (only add information we will most likely use).

6.1 Adding more information to your decay

First, we are going to add more features to our script by adding a custom Functor Collection or using the already-defined collections in the DaVinci project. Remember you can see all the Functor collections available in: `https://lhcb-davinci.docs.cern.ch/autoapi/FunTuple/functorcollections/index.html`.

- For the final state particles (muons and pions), add PID information (`ProbNNpart` and `PID_part`, with `part=P,PI,K,MU`), kinematics (momenta, energy, pseudorapidity).
- For the composite particles (ϕ and $D^+_{(s)}$) add the functors which allow you to get information how the quality of the particle reconstruction was made (DOCA, DIRA, χ^2, IPCHI2, flight distance χ^2...).

You need the following imports:

```
from PyConf.reading import get_pvs

import FunTuple.functorcollections as FC

from GaudiKernel.SystemOfUnits import MeV
```

Hint: In the DaVinci Run 3 lessons you studied the $B^+ \to J/\psi(\to \mu^+\mu^-)K^+$ decays. At track level the decay is the same as the one which we are studying today, so you can take a view in the class to get inspired :)

We can use the `Kinematics` and `ParticleID(extra_info=True)` to retrieve the information. The reconstructed information can be retrieved with this collection of functors:

```
comp_features = FunctorCollection({
    "BPVFDCHI2"      : F.BPVFDCHI2(pvs),
    "BPVFD"          : F.BPVFD(pvs),
    "BPVLTIME"       : F.BPVLTIME(pvs),
    "BPVDIRA"        : F.BPVDIRA(pvs),
    "MAXDOCACHI2"    : F.MAXDOCACHI2,
    "MAXDOCA"        : F.MAXDOCA,
    "MAXSDOCACHI2"   : F.MAXSDOCACHI2,
    "MAXSDOCA"       : F.MAXSDOCA,
    "END_VX"         : F.END_VX,
    "END_VY"         : F.END_VY,
    "END_VZ"         : F.END_VZ,
})
```

6.2 Adding the Event Information

Some information about the decay we're studying does not refer to a given particle of our decay chain, but to the entire event. For example, one can think of the number of charged tracks that have been reconstructed or the Run Number (a run in data taking is a collection of events taken during a small amount of time, no longer than an hour). By default, the FunTuple will add both `EVENTNUMBER` and `RUNNUMBER` to your ntuple.

- Add the Event Information to your ntuples using the functor collection that we saw in the DaVinci lesson.
- **Advanced**: Retrieve also the Reconstruction Summary information with the `RECSUMMARY` functor. You'll need to import the following line:

  ```
  from PyConf.reading import get_rec_summary
  ```

  You can find an example on how to use the functor below:

  ```
  'nTracks' :  F.VALUE_OR(-1)  F.RECSUMMARY_INFO(rec_summary, 'nTracks'
  ```

In this example, we are retrieving the number of charged tracks that have been reconstructed in a given event, `nTracks`. Other variables can be retrieved by changing the argument `nTracks` for any other that is listed here: `https://gitlab.cern.ch/lhcb/LHCb/-/blob/master/Event/RecEvent/include/Event/RecSummary.h?ref_type=heads#L48`.

Also, check what the `RecSummary` functor collection adds to the ntuple.

6.3 Adding MCTruth info

As we said, the DaVinci script has to be made both for data and MC simulations. Until now we are in that scenario, but it is very useful to also add the MC generation-level information to our simulation ntuple (MC TRUE). It's normally used when calibrating simulations in order to know signal efficiencies.

- Use the `MCTruthAndBkgCat` algorithm to retrieve MC TRUE information. For that, check the implementation we made during the lessons. Be sure that the code is made for running both over data and MC samples.
- **Advanced:** Some functor collections have been made in order to get several MCTruth variables of your decay. You'll find a list of them in the functor collections reference given in section 6.1. Their names start with MC (e.g. MCKinematics).

You need the following imports:

```
from DaVinciMCTools import MCTruthAndBkgCat
```

We can use the available functor collections in DaVinci:

```
mctruth = MCTruthAndBkgCat(input_particles=data, name="MCTruthAndBkgCat_functor")

variables = FC.MCKinematics(mctruth_alg=mctruth)
variables += FC.MCHierarchy(mctruth_alg=mctruth)
variables += FC.MCPrimaryVertexInfo(mctruth_alg=mctruth)
variables_head.update({"BKGCAT": mctruth.BkgCat})
variables_head += FC.MCPromptDecay(mctruth_alg=mctruth)
variables_comp += FC.MCVertexInfo(mctruth_alg=mctruth)
```

To add custom collections of functors:

```
variables = FunctorCollection({
    "TRUEP": mctruth(F.P),
    "TRUEPHI": mctruth(F.PHI),
    "TRUEETA": mctruth(F.ETA)
})
```

6.4 Adding Trigger Selection information

One the most important tasks while doing your analysis is having information about the trigger that was used to record your data. So addition information about the HLT1 or HLT2 lines that our data passed in our ntuples will be mandatory

- Add information about the HLT1 trigger[2] to your ntuples. Start from the two usual `Hlt1TrackMVA` and `Hlt1TwoTrackMVA`. Add the TISTOS information to the ntuples (use the `HltTisTos` functor collection).

[2]Remember that this data is turbo, so it is directly the output of a HLT2 line

- **Advanced**: There is list of HLT1 lines regarding muon reconstruction, which can be really helpful in our case. Look at the HLT1 lines at Allen `https://allen-doc.docs.cern.ch/selection/hlt1.html` and add the muon and dimuon lines.

7 DTF and mass constraints

As we saw, the DecayTreeFitter algorithm is a fantastic tool to improve the resolution of our mass peaks once we have knowledge of our decay. In this case, the $\phi \to \mu^+\mu^-$ should have quite good resolution, but we can apply some extra constraints to improve it a bit more.

- Add the DecayTreeFitter algorithm to your DaVinci script and add the features to the head of the decay.
- First try to add a primary vertex constraint and see the effect it has on the mass resolution.
- Now add the ϕ mass constraint to the DecayTreeFitter. **Note**: If you notice it, the DecayDescriptor of the line is `D+ -> (J/psi(1S) -> mu- mu+) pi+`, so the dilepton combination is labeled as J/ψ. So, if we add the `phi(1020)` as a constraint the algorithm won't know what particle are we trying to apply the mass constrain to. We have to tweak the algorithm a little bit to make it work, which consists on constraining the `J/psi(1S)` candidate and performing a mass substitution. For that, check the following example `https://lhcb-davinci.docs.cern.ch/examples/tupling/option_davinci_tupling_DTF_substitutePID.html` and infer how could it be applied to our case.

You'll need to import the DecayTreeFitter class:

```
from DecayTreeFitter import DecayTreeFitter
```

This is how we add dtf fit results:

```
dtf = DecayTreeFitter(name="PV_dilep_Fit",
                      input_particles=data,
                      input_pvs=pvs,
                      substitutions=["D+ -> (J/psi(1S){{phi(1020)}} -> mu- mu+) pi+"],
                      mass_constraints=["phi(1020)"],
                      )
```

We can use the dtf object to obtain information using the fit result:

```
dtfinfo = FunctorCollection({
    'MASS_PhiConstr':dtf(F.MASS),
    'PT_PhiConstr':dtf(F.PT)
    'CHI2DOF_PhiConstr':dtf(F.CHI2DOF)
})
```

Does it make a huge effect to the mass resolution to perform the ϕ mass constraint?

Actually not, as muon momentum reconstruction doesn't degrade the mass resolution very much. The ϕ resonance is considered to have narrow width, but not as much as the J/ψ, and this width has a similar order of magnitude than the mass resolution of the candidate.

8 Prepare your Analysis Production (optional)

Finally, the exercise if finishing. If you manage to get here, your DaVinci script should be run perfectly with all the information you need for performing your analysis. So now, we want to run it over all the data and simulated samples to start our analysis, and for that we have to prepare our Analysis Production (AP)

- Clone the AP repository and create a branch with the name you want.
- Create a folder called `sk-practice-session` and copy your DaVinci options file inside.
- Prepare your `info.yaml`.
- Commit all your changes, and test the job locally.
- Once you're happy push your commits to remote and check that the pipelines are running. **Remember to add the *Starterkit* label!**

```
defaults:
    wg: charm
    application: DaVinci/v64r12
    inform:
        - name.surname@cern.ch

data_sprucing24c3_2024_MagUp:
  input:
    bk_query: /LHCb/Collision24/Beam6800GeV-VeloClosed-MagDown/Real Data/Sprucing24c3/
    94000000/CHARM.DST
  options:
    entrypoint: sk-practice-session.options_file:main
    extra_options:
      input_raw_format: 0.5
      input_type: ROOT
      geometry_version: run3/2024.Q1.2-v00.00
      conditions_version: master
      simulation: False
      input_process: TurboPass
      input_stream: charm
  output: DS2PHIPI.ROOT

MC_sprucing24c3_2024_MagUp:
  input:
    bk_query: /MC/2024/Beam6800GeV-2024.W31.34-MagUp-Nu6.3-25ns-Pythia8/Sim10d/HLT2-2024.W31.34/
    23173003/DST
  options:
    entrypoint: sk-practice-session.options_file:main
    extra_options:
      input_raw_format: 0.5
      input_type: ROOT
      simulation: True
      data_type: "Upgrade"
      dddb_tag: dddb-20240427
      conddb_tag: sim10-2024.Q3.4-v1.3-mu100
      input_process: "Hlt2"

  output: DS2PHIPI.ROOT
```