Table of Contents

How to (re)run the Hlt

Think about your input!

- Moore expects to be running on L0-data
- Moore expects to be running **before** reconstruction
- Moore expects to be running on a full DST or RAW(MDF) file
- Moore expects the raw event to be located at **DAQ/RawEvent**
- Moore does not expect to be running on top of a previous Moore's output

Think carefully about what is there on your input.

- RawEventJuggler: to move around the raw event back how Moore expects it, a prerequisite for running Moore
- L0App: to add the correct L0 to your file, a prerequisite for running Moore

Think about your Trigger conditions!

- Moore for 2012 data which ran in the pit on 2012 data will only make sense to apply to 2012 simulation
- Moore for 2011 data which ran in the pit on 2011 data will only make sense to apply to 2011 simulation
- ... and so on.

Multiple Trigger Results into one file?

- Yes, this is possible, see examples on the RawEventJuggler wiki

Running Moore

Moore is the trigger software, as run in the pit. It is available through the usual SetupProject command.

Default Moore options

This is a collection of options files that perform common tasks. If you have a use case you think is also useful for others, add your **commented** options here with a brief description of what they do.

Options for untriggered data

A skeleton script that should work on Moore v23r2 and later that runs on real raw data looks like this:

```
#!/usr/bin/env gaudirun.py
import Gaudi.Configuration
from Moore.Configuration import Moore

#Configure L0 emulation
Moore().ForceSingleL0Configuration = True
Moore().CheckOdin  = False

# How are we configuring the trigger? TCK or settings?
#Moore().ThresholdSettings = 'SomeThresholdSettings'     # Only use ThresholdSettings when develo
Moore().UseTCK = True                                    # Use a TCK in all other cases.
Moore().InitialTCK = '0x00a10045'                        # Change me to the TCK you want to use
Moore().WriterRequires=['HltDecsionSequence']            # What sequence must pass for the events
```

```
Moore().DataType        = "2012"
Moore().Simulation      = False
Moore().UseDBSnapshot = False
Moore().DDDBtag         = "head-20120126"  #Change me to the DB tag you want
Moore().CondDBtag       = "head-20120607"  #Change me to the DB tag you want
Moore().inputFiles  = [ 'castor:/castor/cern.ch/user/e/evh/114774/114774_0x003D_NB_L0Phys_00.raw'
Moore().Verbose = False
Moore().EvtMax = 100
Moore().outputFile  = 'mymooreout.raw' # If the input file is DST (raw) file, this should be a DS

from Configurables import EventSelector
EventSelector().PrintFreq = 100
```

TCK or Settings?

As a user running Moore on MC, you should **always** use a TCK instead of configuring using settings. This is at present the only way to guarantee that you will get the same trigger decisions as would have been obtained in the pit.

If you are developing new Hlt lines, you should run from ThresholdSettings. Please note that in this case, the resulting DST **cannot** be processed with DaVinci. Any monitoring of quantities that you want to do as part of the development must be done in the same job. You could for example use GaudiPython and create an NTuple.

Running from a TCK

Which TCK?

When a new TCK is made to be used in the pit, a new MC TCK is also made. The difference between the two is that prescales are turned off in the MC TCK. As of late this year, all MC TCKs will start with a "4", so if 0x00XXXXXX was run in the pit, 0x40XXXXXX is the MC TCK to use. You can explore the contents and differences of these TCKs using TCKsh. If the MC version is unavailable contact the HLT experts to get it made for you.

If you want to find out what TCKs were used for a given run, look it up in the rundb: http://lbrundb.cern.ch

To find out what TCKs at-a-glance were used for a given year, take a look at the summary from the Operations Plots page

Version-specific configs

Moore is highly version dependent. You need to run the Moore version your TCK was designed for. Do **not** just take the latest version. You can find out which version of Moore is needed by which TCK using the TCKsh command

```
getTCKInfo(%TCK%)
```

Multiple TCKs: adding a new TCK to a pre-existing dataset

Lets say you have MC that has been produced with one TCK, and you want to get trigger decisions for a different one. To do this you need to move the old TCK out of the default location before re-running Moore. The RawEventJuggler (see the twiki page) is what you need. The following script will do this for you:

```
from Gaudi.Configuration import *
from LHCbKernel.Configuration import *
from Configurables import GaudiSequencer,RawEventJuggler
```

```
from Configurables import LHCbApp
LHCbApp()

tck = "0xORIGINALTCK"

MySeq=GaudiSequencer("MoveTCKtoNewLocation")
MyWriter=InputCopyStream("CopyToFile")

RawEventJuggler().GenericReplacePatterns={"Averyunlikelystringtohaveinaraweventlocation99": "idon
RawEventJuggler().TCK=tck
RawEventJuggler().Input=0.0
RawEventJuggler().Output=3.0
RawEventJuggler().Sequencer=MySeq
RawEventJuggler().WriterOptItemList=MyWriter
RawEventJuggler().KillInputBanksAfter="L0*|Hlt*"
RawEventJuggler().KillExtraNodes=True #remove DAQ/RawEvent completely


ApplicationMgr().TopAlg = [MySeq]
LHCbApp().EvtMax=-1
from GaudiConf import IOHelper
IOHelper().inputFiles([ "myInputDataSet.dst" ])
IOHelper().outStream("myOutputDataSet.dst",MyWriter)
```

To run it, set up the latest version of LHCb:

```
SetupProject LHCb
```

Copy the script from here and modify it to point to your input dataset. Change the tck=0xORIGINALTCK to the one contained in the file that you want moved out of the way. You may also need to change the RawEventJuggler().Input=0.0 line to a different input event format depending on the way in which the input file was produced. See here for information on what the formats and their corresponding numbers look like. Now run the script:

```
gaudirun.py movetck.py
```

You can check this has worked using Dst_Explorer:

```
SetupProject bender
Dst_Explorer myOutputDataSet.dst
ls('/Event/Trigger')
```

If your move worked, you should see the following:

```
>>> ls('/Event/Trigger')
 /Event/Trigger                                                  DataObject
 /Event/Trigger/0xORIGINALTCK
 /Event/Trigger/RawEvent                                         LHCb::RawEvent
```

You can now run Moore using the script shown earlier, which will write your new TCK to the modified output dst. You can do this several times, moving and writing new TCKs each time, until you have several TCKs saved to Event/Trigger/TCK1, Event/Trigger/TCK2, etc. If you don't move the last TCK you write to the file, you don't need to do anything special to access it- All LHCb applications will see its trigger decisions by default.

Multiple TCKs: recovering an old TCK from a file with several

If you want to access the old TCK after adding new ones as described above, SetupProject your application with the additional option:

```
SetupProject <Application> --use RawEventFormat
```

Where is DaVinci or whatever. The RawEventJuggler (see the twiki page) is what you need. Add the following lines to your application at the start of your options. Everything after this will see the TCK you requested, assuming it was written to the file:

```
RawEventJuggler().Input=3.0
RawEventJuggler().Output=2.0
RawEventJuggler().DataOnDemand=True
RawEventJuggler().TCK=0xTCKYOUWANT
```

Easy as that!

Using Moore v23r4 with new (as of February 2015) style trigger lines

1. lb-dev Moore v23r4
2. Get head versions of following packages:

- Hlt/HltTracking
- Hlt/Hlt1Lines
- Hlt/Hlt2Lines
- Hlt/HltConf
- Hlt/HltSettings
- Tf/PatAlgorithms
- Tr/TrackTools
- Tr/TrackInterfaces
- Tr/PatPV

1. make configure && make
2. Skeleton Moore.py:

```
#!/usr/bin/env gaudirun.py
import Gaudi.Configuration
from Moore.Configuration import Moore

Moore().ThresholdSettings = 'Physics_September2012'
Moore().UseTCK = False
Moore().EvtMax = 100
Moore().outputFile  = 'mymooreout.raw'

# In this example, we run on 2012 raw data. Not MC.
from PRConfig.TestFileDB import test_file_db
input = test_file_db['2012_raw_default']
input.run(configurable=Moore())
```

3. ./run gauditun Moore.py

Running from

When running from ThresholdSettings to develop and Hlt line, use

```
$> TCKsh
>>> listConfigurations()
MOORE_v23r2
    Hlt_PassThrough
      0x809d0000 : d55392174b99e00234c02e4c31050e51 : 2014 PassThrough
    Physics_September2012
```

```
0x0a000044 : 0524406614869d6d9813cba5ccaca6f0 : 2012 Physics, September default, less charm
0x0a010046 : 49bfcfac7e695ba1d94b8bd4e1226ff0 : As 0x00a90046 for november 2014 commissioni
```

In the example output above, the name of the ThresholdSettings is listed between the Moore version and the TCKs, in this case e.g. **Physics_September2012**. Set this in the script listed above:

```
Moore().ThresholdSettings = 'SomeThresholdSettings'      # Only use ThresholdSettings when develo
Moore().UseTCK = False                                   # Don't use a TCK in this case.
```

Adding an Additional Hlt2 line.

There are two ways to add an additional Hlt2 line to Hlt/Hlt2Lines.

1. In an existing Hlt2XXXLines.py file
2. In a new Hlt2XXXLines.py file

When creating a new HLT2XXXLines.py file, have a look at the existing Hlt2XXXXLines.py files and copy the skeleton. The class that build the lines and inherits from HltLinesConfigurableUser **must** be called Hlt2XXXXLinesConf, where XXXX is the same as in the file name. Don't forget to add an ID for your line to the HltANNSvc by picking a number larger than 65000 that isn't taken yet.

Instantiate you Hlt2 line inside the

```
__apply_configuration__
```

method of your configurable of choice and then add the following to the script:

```
from Configurables import HltConf
HltConf().AdditionalHlt2Lines = ['TheNameOfYourLine']  # The name of your line is given to the co
```

This topic: LHCb > RunningMooreOnMC
Topic revision: r14 - 2015-02-13 - NikitaKazeev