Table of Contents

Analyzing the output of the MuCTPI simulation

Creating analizable ROOT NTuples

I assume, that you have now the POOL file with the output of the MuCTPI simulation ready. The easiest way to analyze it is to "translate" it to a standard ROOT file. I've been using the following two jobOption files to do this: ntupleMakerConfig.py and jobOptions.ntupleMaker.py.

jobOptions.ntupleMaker.py should generally be set up right to run the job, one should only modify ntupleMakerConfig.py. There the input and output file names can be defined, together with the detector decription version. Still remember to use the same DD, as earlier!

Having copied the two job option files to your run directory, and set up CMT, the NTuple creation job can be run with the command: `#>athena.py ntupleMakerConfig.py jobOptions.ntupleMaker.py` (**Note: To use these jobOptions, you'll also need at least version TrigT1Muctpi-00-00-24 of the MuCTPI simulation.**)

jobOption for releases from 11.0.1 onwards

Because of some modification in the release, from release 11.0.1 the following jobOption has to be used if you want to be able to use the CBNT_MuctpiRoI algorithm: jobOptions.ntupleMaker2.py.

It adds a new service, EventInfoMgt, as needed by the geometry services.

Also there is a commented line in the jobO: `#ReadMuCTPI_RDO.RDOLocID = "LUT_MUCTPI_RDO"` When running the MuCTPI simulation on digitized data, the output of the simulation is usually not saved with the ID "MUCTPI_RDO", but with a different name, to ease the comparison to the previous run. The RDOLocID parameter of the ReadMuCTPI_RDO algorithm defines where to look for the MuCTPI_RDO object.

Analyzing the NTuples

I've written two classes usable from a ROOT macro, which give easy access to the data words of the output of the MuCTPI simulation. Detailed information on the format of these data words can be found in the MIROD manual. The two classes are **MuonDataWordAnalyzer** and **MuonMultiplicityAnalyzer**. Their source code can be found in this page's attachments.

Of course there can be very different things to analyze. To give an example, here is a simple macro, that prints out various informations of the muon candidates: double_muon_info.C

(I assume the reader has some experience with running ROOT macros, so I don't go into detail on how to use these.)

Modifying the look up tables of the MuCTPI simulation

The sector logics of the barrel and endcap trigger chambers are connected to 16 so called MIOCT modules. These modules are responsible for resolving overlapping muon candidates (probably coming from the same muon). More secription on how this is done can be found in the MIOCT manual.

As can be seen on the 3. page of the document, one MIOCT is responsible for resolving the overlaps between 4 Barrel and 6 Endcap sectors. It can also be seen from the figure, that there should be no overlaps between the "upper" and "lower" sectors. Note: The layout of the Forward sectors has changed since the writing of this document.

The LUT files should define, when two muon candidates should be considered as coming from the same physical muon. The structure of the LUT files is much the same as the tables found at the end of the MIOCT document. For each condition in which a muon candidate should be suppressed, there is a row in the LUT. (Lines beginning with a # are comments. Also, there should be no empty lines.) There is actually some description on the meaning of the various columns in the example file (**~/data/MioctLUTupper.data**).

There can be 2 muon candidates coming from each of the sectors. For each candidate it's RoI and overlap flag should be defined. An RoI and overlap flag of 0 means that the given candidate is not considered. (**Note: The code may change in this respect later on! This scheme was developed to handle barrel phi overlaps with overlap flags set. If we want to handle overlaps between the barrel and the endcap where the overlap flag in the barrel is not set, this might have to be changed!**)

The last 16 numbers in the rows define which candidate should be discarded in case of an overlap. 1 means the candidate should be suppressed, and 0 means the candidate should be kept. The order is documented in the sample file.

About the layout of the sectors

The MIOCT modules get the input from the following barrel and endcap sectors (I really hope this is accurate...):

MIOCT number	Name in LUT		BA01	BA02	EC01	EC02	EC03
1	Sector Number	upper	0	1	2	3	4
		lower	30	31	47	0	1
2	Sector Number	upper	4	5	8	9	10
		lower	2	3	5	6	7
3	Sector Number	upper	8	9	14	15	16
		lower	6	7	11	12	13
4	Sector Number	upper	12	13	20	21	22
		lower	10	11	17	18	19
5	Sector Number	upper	16	17	26	27	28
		lower	14	15	23	24	25
6	Sector Number	upper	20	21	32	33	34
		lower	18	19	29	30	31
7	Sector Number	upper	24	25	38	39	40
		lower	22	23	35	36	37
8	Sector Number	upper	28	29	44	45	46
		lower	26	27	41	42	43

The other 8 MIOCTs have the same sector numbers, but for the sectors of the other rapidity region.

The current idea is, that the different MIOCTs handle the overlaps in the same way, ie. using the same look up tables. If the optimization requires, this could be changed later on.

About my first look at phi overlaps

I created the sample LUT, **MioctLUTupper.data** by shooting single muons in the eta=0.0 - 1.0, phi=0.18 - 0.22 region with a pt of 30 GeV. This is the overlap region of the barrel sectors 0 and 1. Hence the "upper" in the name of the file, if you look at the previous table.

If you look at the file, there are a lot of conditions defined, in which the first or second candidate of BA01 (sector 0 in this case) overlaps with the first or second candidate of BA02 (sector 1 in this case).

Barrel - Endcap overlaps

The new LUT files, 6GeVSBEMioctLUTupper.data and 6GeVSBEMioctLUTlower.data implement a very simple scheme to handle fake double-counts between barrel and endcap sectors. In this scheme if a muon candidate is found in RoIs 17-26 in a barrel and in RoIs 0-31 in an overlapping endcap sectors, then the endcap candidate is not taken into account in the multiplicity sum.

It is not needed to have the eta overlap flag set for the candidates. In fact, the MuCTPI simulation assumes that the flag is not set, so it doesn't resolve the overlap if one of the candidates is flagged. This will have to be changed later on, but this was the fast way to implement the changes necessary.

As noted earlier, the format of the files has indeed changed somewhat, but by now I realized I'll be the only one ever modifying these look up tables, so I rather not go into details here.

-- AttilaKrasznahorkay - 15 Dec 2005

This topic: Main > AnalyzingDigOutput
Topic revision: r4 - 2005-12-15 - AttilaKrasznahorkaySecondary