Available on CMS information server **CMS IN 2009/XXX**

CMS Internal Note

28 August 2009

CSCValidation

A. Kubik, S. Stoynev, M. Schmitt

Northwestern University

N. Terentyev

Carnegie-Mellon University

I. Bloch

Fermi National Accelerator Lab.

P. Jindal

Purdue University Calumet

Abstract

CSCValidation is a CMSSW class of type EDAnalyzer used for monitoring detector performance and data quality in the Cathode Strip Chambers (CSCs) of the muon endcap subsystem. It began to fill a gap in monitoring at the offline level in the 2006 MTCC exercise, and has evolved to include monitoring of advanced quantities such as efficiencies and stand-alone muons. During the CRAFT running of fall 2008, CSCValidation was used as "offline prompt-analysis" for the CSCs, with the goal of marking runs good or bad for cosmic physics analysis. It is also used extensively in validating new CMS Software releases with respect to the CSC system. Finally, the code is written and intended as an example for newcomers to CMS analysis. This note will describe the various modules of the CSCValidation package, as well as its applications. A brief set of instructions for running the code is also given.

Preliminary version

1 Introduction

CSCValidation is a CMSSW class of type EDAnalyzer used for monitoring detector performance and data quality in the Cathode Strip Chambers (CSCs) of the muon endcap subsystem. It began to fill a gap in monitoring at the offline level in the 2006 MTCC exercise, and has evolved to include monitoring of advanced quantities such as efficiencies and stand-alone muons. During the CRAFT running of fall 2008, CSCValidation was used as "offline prompt-analysis" for the CSCs, with the goal of marking runs good or bad for cosmic physics analysis (example plots in this note are from CRAFT '08 and '09 cosmic ray runs). It is also used extensively in validating new CMS Software releases with respect to the CSC system. Finally, the code is written and intended as an example for newcomers to CMS analysis.

2 Modules

The code of CSCValidation is broken up into a modular format to help control flow, make the code easier to follow, and allow multiple authors to easily contribute more code and functionality. Each module is a member function of the CSCValidation class and has a distinct functionality. Every module can be enabled/disabled through the use of a CMSSW configuration file boolean switch, set at runtime (instructions for running the code will be given later). The following is a brief description of all current modules and their functionality. Emphasis is given to describing the quantities monitored through CSCValidation, and not the details of the hardware/reconstruction/calibration.

2.1 Occupancies

CSCValidation creates plots of the occupancy versus chamber of four of the basic reconstruction objects in the CSCs (anode wire Digis, cathode strip digis, rechits, and segments). These occupancy plots are a top level tool for evaluating a given dataset, and give a general overview of which chambers are functioning and supplying data, as well as which may be inefficient or noisy. An example occupancy plot for rechits is given in Fig. 1. Only one entry per event is allowed, to keep showers or noisy chambers from drowning out the rest of the picture.

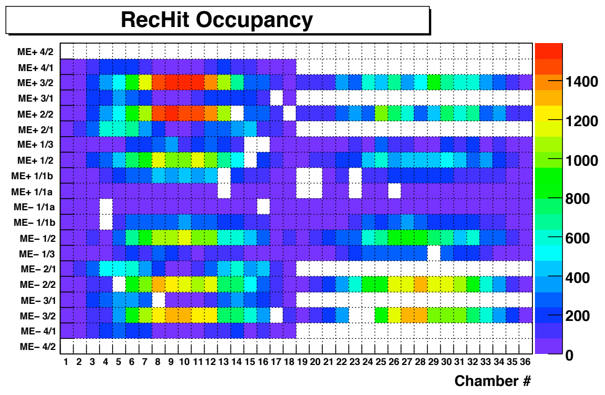


Figure 1: Example of RecHit Occupancy versus Chamber.

2.2 Filter

In some cases, it becomes desirable to filter events which are analyzed to select for quality or other features. CSCValidation currently includess two types of filters, both which can be turned on or off at runtime. The first simply asks that the CSC system has provided a L1A trigger. The second filters events based on event quality. It uses reconstructed tracks (StandAlone muons) with various quality cuts to determine if the event is worth keeping. All quality cuts are configurable in the CSCValidation python configuration. As described previously, the filter is applied before all modules except the occupancy module, which remains unfiltered to enable identification of hot and dead channels.

2.3 Digis

In order to reconstruct muons in CMS, we must first be able to extract useful information from the raw data which is written from the detector. RAW data is processed and formatted as data objects called "digis", created by a

CMSSW module referred to as the raw data unpacker [1]. Digis contain the most basic level of information from the detector, organized into collections which can be utilized by more advanced reconstruction algorithms.

There are multiple types of CSC digis corresponding to the different types information from the CSC detector. Three types of digis are currently directly analyzed by CSCValidation: anode wire digis, cathode strip digis, and comparitor digis. The code retrieves the digi collections, loops over them, and fills histograms with various quantities from them, described below.

2.3.1 Anode Wire Digis

The wire digis carry information from the anode wires of the CSCs. There is one wire digi created for each cluster of consecutive wiregroups which fire in a given layer of a CSC chamber. Each wire is either ”on” or ”off” for a given event. The digi contains information about the wiregroup number which created it, as well as the relative timing of the signal in the readout window. CSCValidation makes plots of both these quantities, with detailed consideration given to the signal timing (Fig. 2 and 3), which plays an important role in triggering and reconstruction. The total number of wires fired per event is also recorded (Fig. 4).

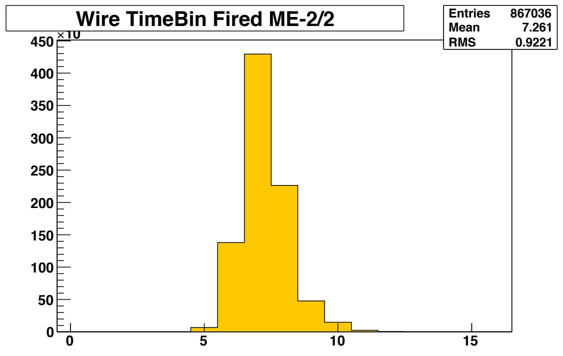


Figure 2: Average time bin fired for chambers in station ME -2/2.

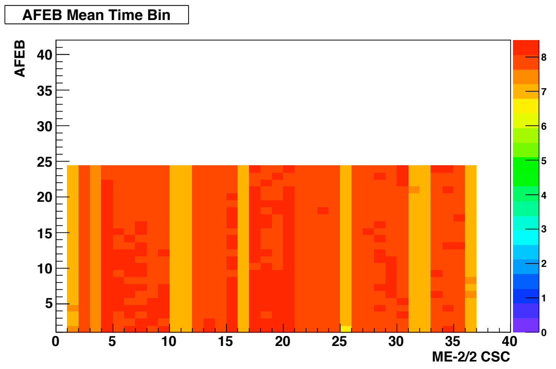


Figure 3: Example plot of time bin fired for all chambers in station ME -2/2.

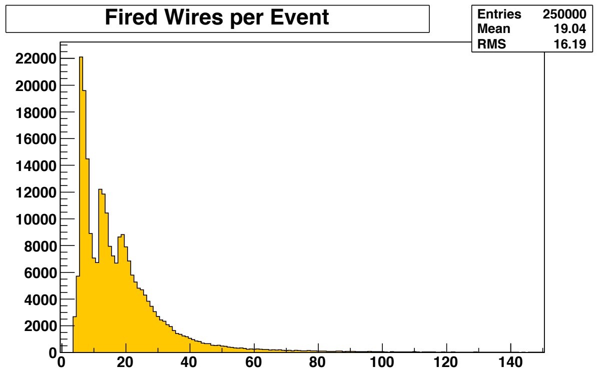


Figure 4: Total number of wires groups fired per event.

2.3.2 Cathode Strip Digis

Similarly to the wire digis, the strip digis relay information from the cathode strips. Strip digis are read out in blocks associated by which CFEB electronics board is reporting data. When a CFEB reports data, a digi is produced for every strip, regardless of whether the strip contains signal. Each digi contains information on which strip it represents, as well as the signal charge collected (in ADC counts) in each of eight 50 ns wide time samples. In CSCValidation, a strip is counted as having fired if and only if the ADC counts in one of the 8 time buckets exceeds 13. This keeps empty strips and noise fluctuations from inflating occupancies (the typical noise level is approx. 2.5 ADC counts). The strip numbers of all fired strips per event is recorded, along with the total number of strips fired (Fig. 5).

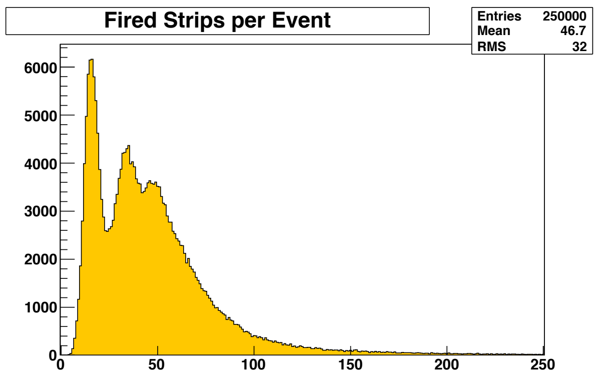


Figure 5: Total number of strips fired (above threshold) per event.

In addition, a second module is employed in order to monitor the cathode strip signal noise. For normal reconstruction in the CSCs, a large, nearly constant signal (referred to as the pedestal) must be subtracted off to get the true signal in the cathode strips. The timing of the readout from the strips is such that the first two time samples should be empty of real signal, and can be used to subtract off this pedestal value. However the pedestal itself has some fluctuation around it's base value. The pedestal noise monitoring module uses an analysis of strips far from those giving signal, which should contain only the pedestal signal. It compares the average pedestal calculated using all 8 time samples in the empty strips to the value in each individual time sample. The difference is entered into a histogram, an example of which is show in Fig. 6. Deviation from zero or a general widening of this distribution can be a sign of noisy channels.

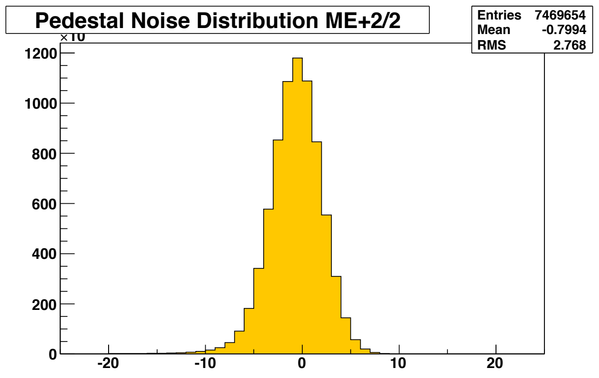


Figure 6: Example pedestal noise plot for all chambers in ME +2/2.

2.4 L1 Trigger

Currently, CSCValidation can analyze information from the L1 trigger to determine if the CSCs have supplied a trigger for the event in question. A simple plot is made of the number of total events along with the fraction of events with a CSC L1 trigger, as well as fraction of L1 triggers from other muon subdetectors. This module also allows for the filtering of events such that only events with a CSC L1 trigger may be analyzed by all other modules, as described earlier. The code for this module is currently undergoing development to add more sophisticated tools for L1 trigger monitoring.

2.5 RecHits

The next step in the reconstruction chain after digis is to combine information from the strip and wire digis into reconstructed hits. Basic information about each reconstructed hit is analyzed and processed by CSCValidation.

First, the position of each hit on the cathode strip from which it was constructed (in units of the width of the strip) is analyzed. A histogram for each chamber type records this value (Fig. 7). The distribution of hits across the width of the strip is expected to be flat and deviations from this behavior can indicate problems in reconstruction or calibration. The estimated error on this strip position measurement is also recorded and monitored for appropriate behavior.

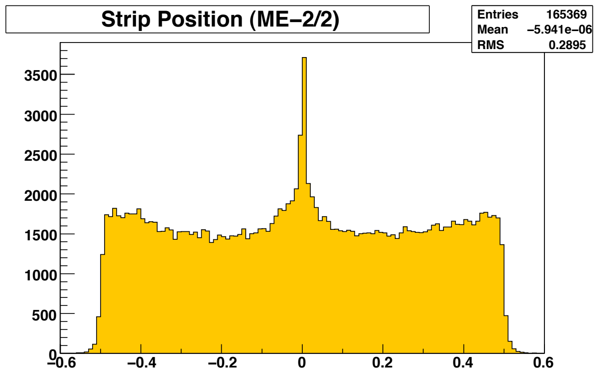


Figure 7: Hit position on the strip for all chambers in ME -2/2.

Next, coordinates (and estimated errors) in the local coordinate frame of each CSC chamber are stored. This information is not recorded by CSCValidation, but instead these coordinates are first transformed into the global CMS coordinate system. The new global coordinates are stored both in a ROOT TTree object and in a 2D histogram. A scatter plot of hit positions can be created from the TTree which is extremely useful in spotting problem areas of the detector. Figure 8 shows an example of such plots, where holes can be seen indicating a non-working chamber or chamber subsection.

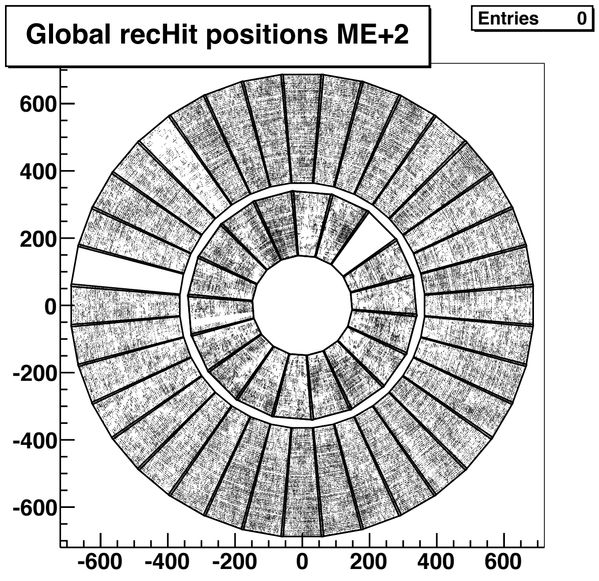


Figure 8: Global position of all reconstructed hits in Station +2.

Another important part of recHit reconstruction is the charge deposited on the cathode strips used to reconstruct the hit. This charge is stored in the recHit object (again in units of ADC counts) and accessed by CSCValidation. Plots are made of the 3 time bin x 3 strip sum of ADC counts for each rechit (or 3 time bin x 1 strip sum for those hits reconstructed from only one strip). Both this sum charge as well as the ratio of charge in side strips to total charge are histogrammed by CSCValidation. These distributions are expected only to change if the gas gain of the CSCs change.

In addition, a dedicated module has been created to monitor these deposited charge distributions more closely. This

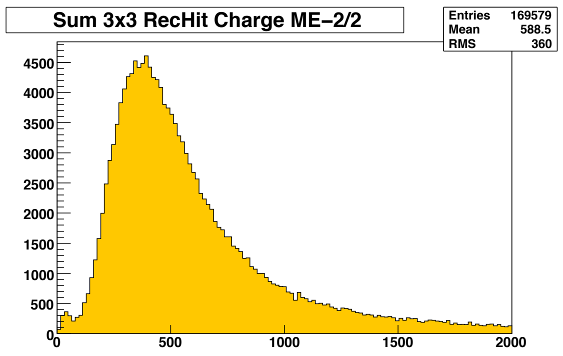


Figure 9: Total charge collected (3 time bin x 3 strip) for each reconstructed hit in ME -2/2.

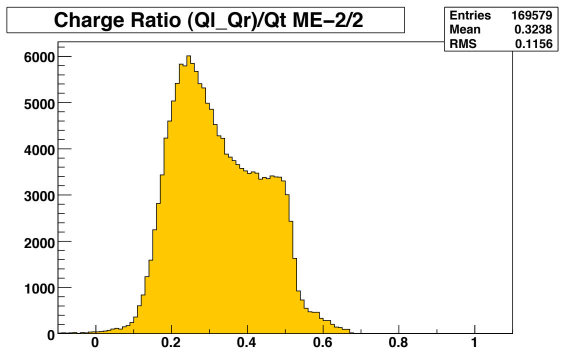


Figure 10: Ratio of charge in adjacent strips to charge in center strip for each reconstructed hit in ME -2/2.

code uses the same 3 time bin x 3 strip sum of ADC counts, but divides and analyzes the results for every high voltage region in the CSC system. High voltage regions are defined by the wire group configuration in each layer of the CSCs. In this way, gas gains across all regions of the CSCs can be analyzed and monitored in detail. An example of the average charge per high voltage region is given in figure 11.

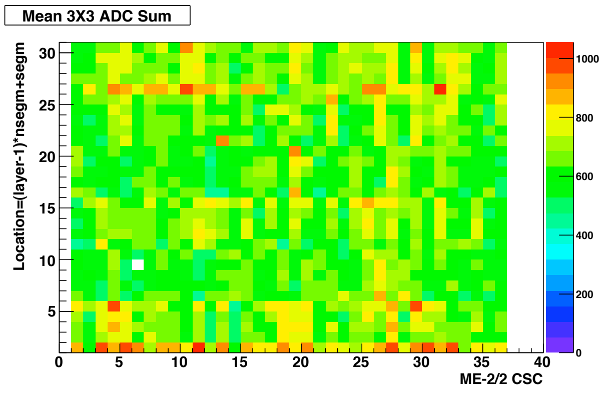


Figure 11: Average charge per high voltage region for ME -2/2.

Finally, each reconstructed hit contains relative timing information constructed using the cathode signal shape. In the current RecHit reconstruction code, a simple fit is performed to extract the signal peak position in the eight time sample readout window described previously, and this value is stored in the rechit object. The signal must be late enough in the window to allow accurate calculation of the pedestal, but not so late that information is lost outside of the window. This value is also used in the application of calibration to the reconstruction of hits.

As with the charge deposition, the RecHit timing makes use of a dedicate module for a more detailed analysis. An average timing for each CFEB board in the system is computed and recorded. This also allows for dead or malfunctioning CFEB boards to be identified. An example of the average signal timing per CFEB in ME +2/2 is shown in Fig. 13.

Finally, the total number of RecHits reconstructed per event is stored (Fig. 14).

2.6 segments

After hits are reconstructed, they are then combined into track segments by a pattern recognition algorithm. These track segments have a maximum of one hit per CSC Layer (six total layers per CSC). A local position is stored for each segment, and CSCValidation again converts this to a position in global CMS coordinates, entering the values into a TTree and 2D histogram. Fig. 15 shows an example of segment global positions in Station +2.

Other segment quality quantities are also histogrammed, including normalized chi squared and chi squared probability of the straight line fit to the segment hits, direction of the segment in global CMS coordinates, the number of

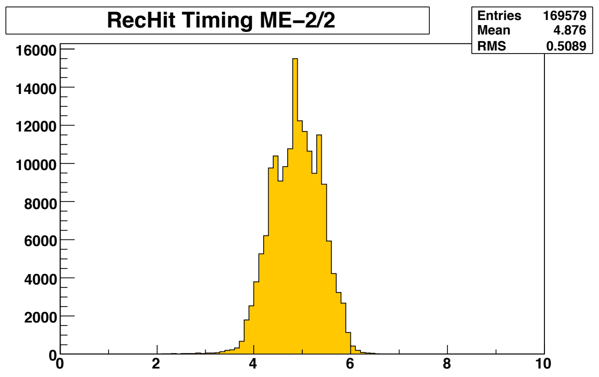


Figure 12: Timing values for RecHits in ME -2/2.

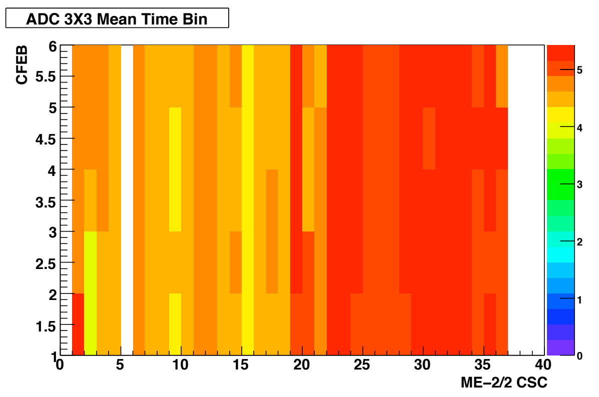


Figure 13: Average timing for RecHits in ME -2/2.

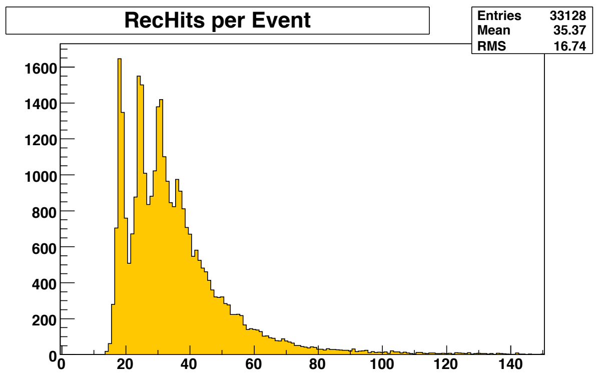


Figure 14: Total number of reconstructed hits per event.

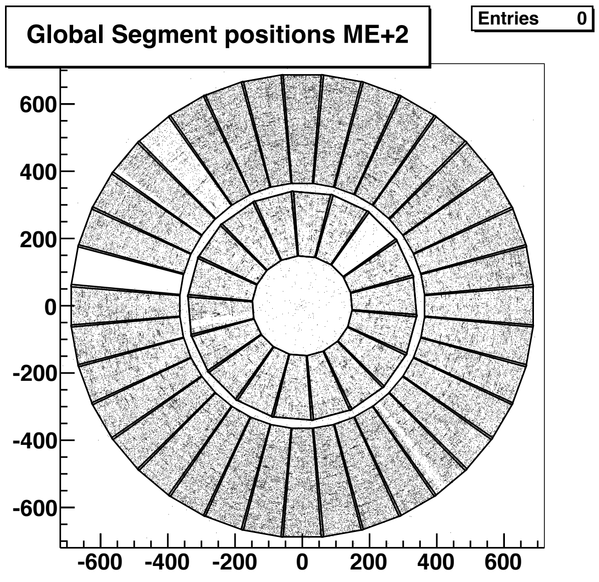
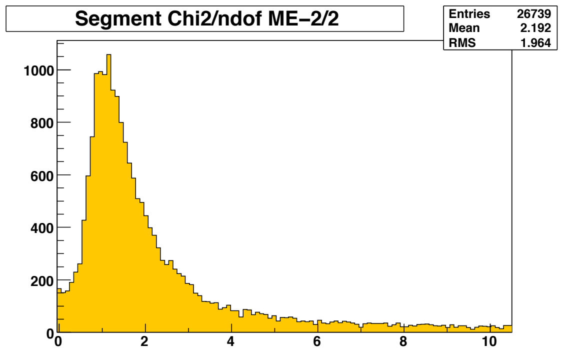


Figure 15: Segment global positions in station +2.

hits on each segment, and the total number of segments reconstructed per event.



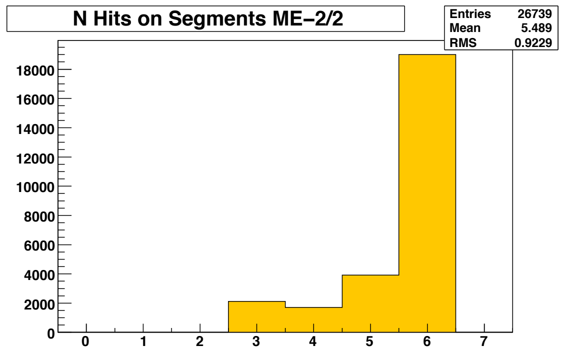


Figure 16: Chi squared of the straight line fit to all segments in ME -2/2.

Figure 17: Number of hits per segment for all segments in ME -2/2.

2.7 StandAlone Muons

CSC track segments can then be combined to reconstruct muon tracks. Hits on these tracks can come from any muon subdetector, however CSCValidation only analyzes tracks in the endcap. Plots are made of the number of hits per track, normalized chi squared of tracks, and reconstructed momentum. Work is underway to make a more sophisticated monitoring of standalone muons.

2.8 Spatial Resolution

The spatial resolution of reconstructed hits is monitored in two ways. The first is a data-driven method which makes use of reconstructed segments. A quality segment containing six hits is required in a chamber, from which the hit on the third layer is then removed and a refit is performed to the other five points. An expected position on the third layer is interpolated from this refit, and the residual between the expected position and the actual reconstructed position of the hit is entered into a histogram (Fig. 20). This distribution can then give a handle on the spatial resolution of hits. It should be noted that this only measures the spatial resolution of the global phi

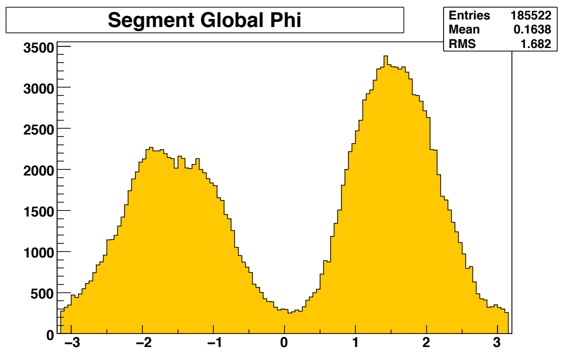


Figure 18: Global Phi direction for all segments.

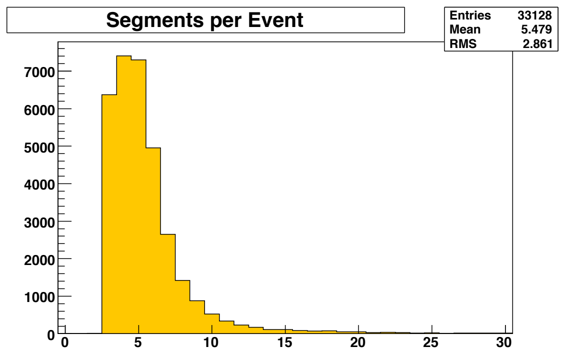


Figure 19: Total number of reconstructed segments per event.

coordinate, which is measured by the strips. The r coordinate is measured by the wires (which have a much coarser resolution) and is not monitored with this method.

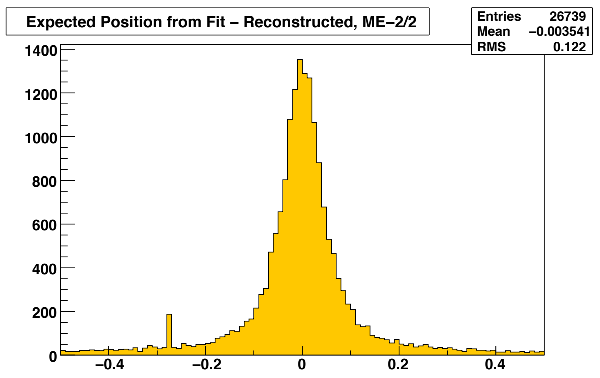


Figure 20: Residuals (in units of strip widths) of expected position from interpolation versus reconstructed position, for all chambers in ME -2/2.

A second method can also be used but only for simulated data. It simply compares a simulated hit to the reconstructed hit, entering the spatial difference for both local x and y coordinates into a histogram.

2.9 Efficiencies

Efficiencies are monitored at a basic level in CSCValidation using fast and non-statistics limited methods. Strip digi, wire digi, rechit, and segment efficiencies are all examined. The method is to first find a high quality segment in any chamber. We assume that the high quality segment is associated with a muon, and extrapolate the segment to a probe chamber. If the extrapolated point is within an active region of another chamber (the active region is defined in the code itself) then one would expect to find a strip/wire/rechit/segment in that chamber. If one is found, this is an efficient event for the probe chamber. The fraction of efficient events is recorded for all chambers, and an example of the result (in this case RecHit efficiency) can be seen in Fig. 21.

A second, less careful, but still useful method is employed for RecHit and Segment efficiencies. For RecHits, CSCValidation looks for any segment, and assumes that each segment should have six hits (one per layer). If it doesn't, it treats this as an inefficient event (one layer didn't have a hit which should have). This does not give a very good handle on absolute efficiency, but is very useful in spotting relative changes between runs or reconstruction code changes. Fig. 22 shows the output histogram of this method. Efficiencies are sorted by chamber type.

For Segments, one assumes that for a given chamber, if 2 or more layers contain hits, a Segment should be reconstructed. If no Segment is reconstructed, the event is counted as inefficient for that chamber. An example is shown in Fig 23.

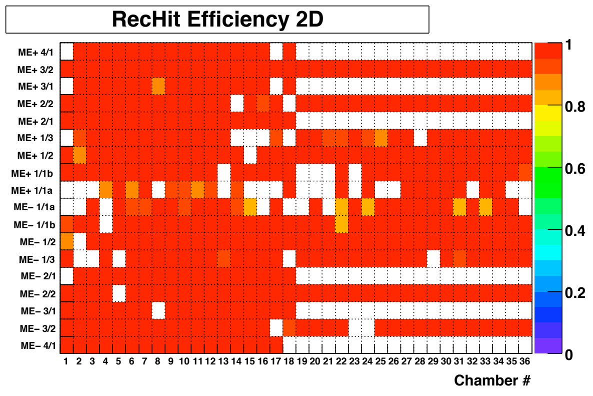


Figure 21: Efficiency for reconstructing hits in all chambers, based on Segment extrapolation from other chambers.

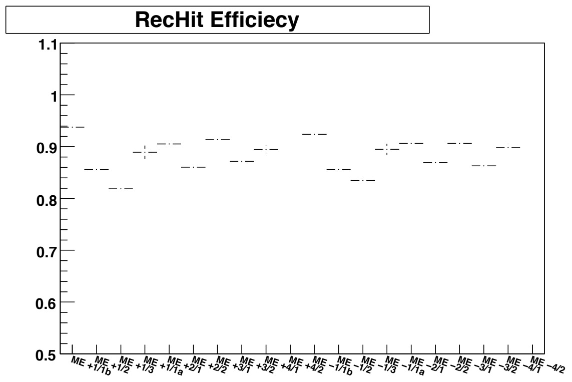


Figure 22: Efficiency for reconstructing hits in all chamber types, based on number of hits per Segment.

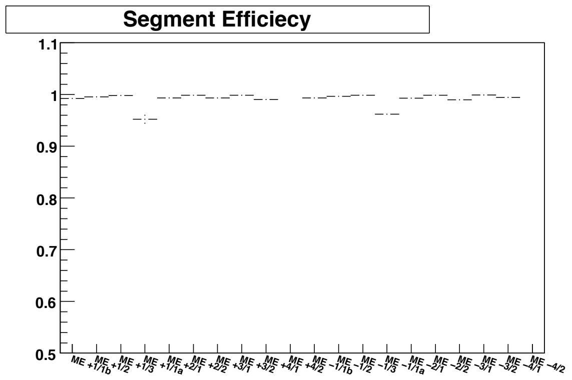


Figure 23: Efficiency for reconstructing segments in all chamber types, when at least 2 layers contain RecHits.

3 Implementation

3.1 CMSSW Release Validation

The CMS Software framework (CMSSW) is extremely complex and constantly evolving. Therefore it becomes necessary to validate that things are working as expected whenever major changes are introduced. This is accomplished primarily through the generation and reconstruction of simulated events in each new major release of CMSSW. These simulated datasets are referred to as RelVals.

For each new CMSSW release, CSCValidation is run over the new set of RelVals which accompany it (usually a single-muon sample). Results are compared to results from a reference, usually the last release which passed the validation procedure. If no major changes are spotted, or only expected changes are found, the release is reported as OK for the CSCs.

3.2 CSC Prompt Analysis

CMS has recently taken large amounts of cosmic ray data, and will soon be taking data with real beam. It becomes very important to check the data and make sure they look OK as quickly as possible. CSCValidation has slowly evolved to improve at this roll. In the CRAFT cosmic data taking of 2008, and again in CRAFT 2009, the code ran in an automated way over all CMS data runs. A HTML template was employed to organize and display the plots in web page format, allowing the results to be checked and feedback given to the teams running the detector, as well as those using the data for analysis. The results are presented and archive from CERN hosted web space at http://cern.ch/CSCValidationResults.

The CSCValidation code has also been ported and is being further developed to run within the data quality monitoring (DQM) framework of CMS. This allows the code to run in real time over CMS physics data (with some prescaling), with results appearing within minutes on the DQM results web page. In addition, the code will be run offline over all CMS physics events, available within a few days of the data being taken.

4 Running the Code

The CSCValidation code can be found in the CMSSW CVS repository under

```
RecoLocalMuon/CSCValidation
```

along with example configuration files for running over both local and global data, as well as Monte Carlo generated events (local data refers to data written from the local CSC DAQ in the form of .raw files, while global DAQ data is in the format of .root files). Here is presented the configuration needed in a CMSSW Python configuration file to run the module. Note that the code requires, at minimum, CSC RecHits and Segments to be present in the target data, or reconstructed in the path before CSCValidation is run.

```
process.cscValidation = cms.EDFilter("CSCValidation",
    rootFileName = cms.untracked.string('validationHists_global.root'),
    isSimulation = cms.untracked.bool(False),
    writeTreeToFile = cms.untracked.bool(True),
    useDigis = cms.untracked.bool(True),
    detailedAnalysis = cms.untracked.bool(False),
    stripDigiTag = cms.InputTag("muonCSCDigis","MuonCSCStripDigi"),
    wireDigiTag = cms.InputTag("muonCSCDigis","MuonCSCWireDigi"),
    compDigiTag = cms.InputTag("muonCSCDigis","MuonCSCComparatorDigi"),
    cscRecHitTag = cms.InputTag("csc2DRecHits"),
    cscSegTag = cms.InputTag("cscSegments"),
    useTriggerFilter = cms.untracked.bool(False),
    useQualityFilter = cms.untracked.bool(False),
    makeStandalonePlots = cms.untracked.bool(False),
    saMuonTag = cms.InputTag("cosmicMuonsEndCapsOnly"),
    l1aTag = cms.InputTag("gtDigis"),
    simHitTag = cms.InputTag("g4SimHits", "MuonCSCHits")
)
```

rootFileName The name of the output file to which CSCValidation will save plots. It can be ¡anything¿.root.

isSimulation A flag to tell CSCValidation if it will be analyzing data or MC. If set to false, it will not run modules which rely on simulated hits to be presetn.

writeTreeToFile Both rechit and segment positions are written to a ROOT TTree object by default, so that plots of global positions can be made after running. These output can be quite large however, so it is included as an option to turn off this feature.

useDigis It is possible in the future that Digi information will no longer be saved in data, therefore by setting this switch to true, one can avoid any calls to Digi information.

detailedAnalysis Turn on this switch to create all standard plots for every layer of every chamber, rather than just each chamber type. This is mainly useful for debugging specific problems. Makes a LOT of histograms.

stripDigiTag The input tag name of the Strip Digi collection.

wireDigiTag The input tag name of the Wire Digi collection.

compDigiTag The input tag of the Comparator Digi collection.

cscRecHitTag The input tag of the CSC RecHit collection.

cscSegTag The input tag of the CSC Segment collection.

useTriggerFilter If set to true, CSCValidation will only process events in which the CSCs have reported a L1 accept.

useQualityFilter If set to true, CSCValidation will only process events which pass a set of event quality criteria (under development)

makeStandalonePlots Set to true to analyze and make plots of Standalone muons.

saMuonTag The input tag of Standalone Muon collection

l1aTag The imput tag of the level 1 trigger collection

simHitTag The imput tag of the simhit collection (only needed in MC if isSimulation set to true above)

After running CMSSW with CSCValidation included in the path, an output root file specified by the parameter 'rootFileName' will be created. This file contains raw unformatted plots which can be further processed using the provided macros with the macros/ folder of the CSCValidation module. There are two main macros, one which creates a standard set of results (in .png image format) from running the code and a second which creates overlaid comparison plots between the current results and a reference set (the reference can be any output root file created by CSCValidation). Both macros use ROOT and therefore it must be available in order for them to work. Example usage of the macros is as follows (typed at the shell prompt):

```
./makePlots.sh <your_file_name>.root
./makeComparisonPlots.sh <your_file_name>.root <reference_file_name>.root
```

References

[1] **CMS Note 2005/000**, X.Somebody et al., *"CMS Note Template"*.