ATLAS Level-1 Calorimeter Trigger 18 November 2004
Software Note 017 Version 1.0

Internal consistency checking of 2004 testbeam data

L1Calo Group [1]

1 Introduction

This note summarises the results of an analysis of the internal consistency of data taken at the 2004 Testbeam with the Level–1 Calorimeter trigger. The scope is limited to those checks of the performance of the trigger system which are possible using L1Calo data alone, and does not attempt to assess the matching between L1Calo trigger data and the calorimeters themselves.

The data used consists only of the ROD data read out from some of the modules in the testbeam setup, and not any peripheral monitoring data. This data was recorded as several ROD fragments which form part of the full event record in the standard ATLAS event format structure. The data was taken in combined runs with the calorimeters at various points during the 25ns beam run in October 2004.

2 Context of recorded data

In order to understand what tests are possible with the recorded data, it is necessary to know some details of the hardware and circumstances at the testbeam. This section tries to concisely summarise the relevant details.

2.1 Modules at the testbeam

The setup of the most important modules at the testbeam is shown in figure 2.1. There was some variation on the ROD connectivity during the course of the testbeam, but for the runs considered in this analysis, the connectivity was stable. The readout therefore consisted of 1 CPM (DAQ and RoI outputs), 1 JEM (DAQ only) and 2 CMMs. Note that there was no data recorded with PPM readout, so the only information available about incoming calorimeter data is that recorded by the CPM and JEM, which had already been processed by the PPM BCID algorithm. This data is seen and recorded at full granularity by the CPM, and exactly the same data should be seen by the JEM in 2 by 2 sums.

[1] Please send any comments and corrections to Stephen Hillier.

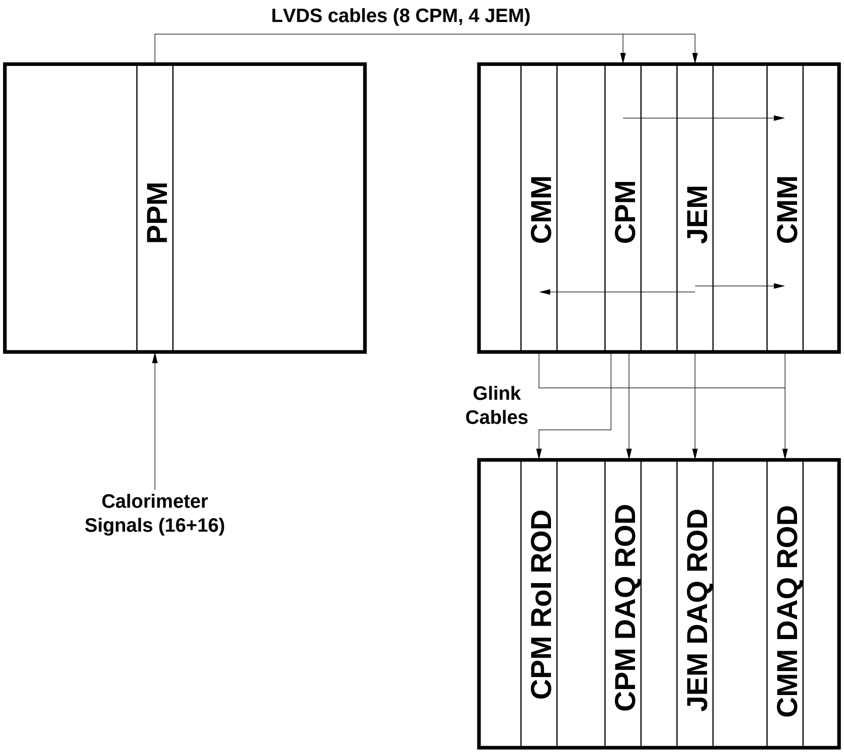

Figure 2.1: Real-time and readout modules at the testbeam

The CMMs were configured as a JEP energy CMM on the left, and a CP cluster CMM on the right. The CPM was configured such that it sent no data (i.e. parity correct zero data) to the energy CMM on the left. This meant that the energy CMM just recorded the energy data coming from the single JEM. The CMM on the right recorded eight hit sums coming from the CPM and the JEM. To try to avoid confusion between CP and Jet hits, the lowest four thresholds were assigned to the CPM and the highest to the JEM.

2.2 Runs of Interest

The runs chosen for this analysis are those highlighted in Murrough's testbeam web pages [1]. They occur in four batches which took place on different days during the 25ns period of the combined testbeam. All consist of data recorded in a combined run, and so in most cases data from the Tile Calorimeter and Liquid Argon Calorimeter should be available, although calorimeter data is not used in this analysis. The four run batches consist of these runs:

- 2101558, 2101563–1566
- 2101662–1666, 2101668, 2101670

- 2101727–1729
- 2101813–1816

In all, there were about 500,000 events recorded in these runs.

2.3 Readout Settings

For all of the DAQ data, the readout for each module was set to output five slices of data. In principle, the idea was for the data related to the L1A to be in the middle of these five slices, and this was indeed carefully set for both the CPM and JEM trigger tower data. Though the peak itself does sometimes slip into the fourth (and occasionally second) slice for some towers, it is true that the majority of the towers were peaked in the right place most of the time, and the CPM and JEM data were correctly lined up with respect to each other. The advantage of reading five slices is that when data did slip, we did not lose the BCID peak information, and all five slices can also be checked for consistency throughout the system.

The processed real-time data (e.g. hits, energies) are also read out into the RODs, and the offsets for these are independent of the offsets needed to locate the tower data. Many of these were also set to the correct value to line up the L1A in the third slice, however some were close, but not optimal. Fortunately, it is possible to trace the third, and most important, slice throughout most of the processing chain right through to the CMM system outputs for the majority of the runs analysed, but often the surrounding slices are lost off the edge of the five slice readout. The RoI readout for the CPM only consists of one slice, and unfortunately that was set to point at the fourth slice for essentially the whole running period. This means that, in general, very few RoIs are seen, since a signal peaked in the third slice virtual guarantees no hits (and so no RoIs) in the fourth slice. However, there are some runs where a large number of channels slip into the fourth slice, and the RoI formation can be checked in those cases.

Table 1 summarises the situation of the readout settings for the runs 2101662–2101670 in terms of which slice (0-4) the L1A data appear at. These settings are true for the majority of the data, and the run range of validity for each setting is given in the table. Obviously, data that appears with a negative number in this table was not read out, and so cannot be checked for consistency with the input tower data. As can be seen from the table, the CMM settings were different for several of the runs of interest. In fact, in some runs (e.g. 1816) they were disastrous enough that essentially no checking is possible, and so this analysis does not attempt to analyse CMM data for runs other than those listed in this table. It is clear from the logbook that CMM settings were being changed between these runs, so the differences are understood. However, a more interesting case is the CPM hits

Data	L1A slice	Runs
CPM towers	2	all
CPM hits	0	2101558–2101728
CPM RoIs	3	all
JEM towers	2	all
JEM hits	2	all
JEM energy	2	all
CMM left backplane	-1	2101662–2101729
CMM left local sum	-1	2101662–2101729
CMM left system sum	2	2101662–2101729
CMM right backplane (CPM)	1	2101662–2101729
CMM right backplane (JEM)	-2	2101662–2101729
CMM right local sum (CPM)	1	2101662–2101729
CMM right local sum (JEM)	-2	2101662–2101729
CMM right system sum (CPM)	3	2101662–2101729
CMM right system sum (JEM)	0	2101662–2101729

Table 1: Readout settings for various data types

offset where it appears to be stable up to a point halfway through run 2101729. It seems strange that this value should have been altered mid-run, so perhaps there is some indication of hardware behaviour that is not understood during that run. After this run, the hit offset is changed for the rest of the data, but in such a way as to make the L1A appear closer to the optimal at slice 1, so this increases the quantity of checking possible.

3 Analysis of data

The basic philosophy of the data analysis was to assume that the 5 slices of CPM data for the 32 connected calorimeter towers was correct, and then check that every other piece of data that could be derived from these quantities was in fact consistent with with these values. For data types where the L1A slice was aligned correctly at the middle slice, it was possible to check all five slices, but for others, only the overlap of readout slices could be checked. For data such as hit results, this check was only really relevant for the slice in which the hit occurred, but for other quantities (e.g. JEM tower data, JEM energies) the usual presence of some small trigger tower values meant that the checks were not trivial for non-L1A slices.

Layer	JEM Phi	CPM Channels	working?
EM	0	0/2,0/3,1/2,1/3	yes
EM	1	0/0,0/1,1/0,1/1	yes
EM	2	0/6,0/7,1/6,1/7	no
EM	3	0/4,0/5,1/4,1/5	yes
Hadronic	0	0/0,0/1,1/0,1/1	no
Hadronic	1	0/2,0/3,1/2,1/3	yes
Hadronic	2	0/6,0/7,1/6,1/7	yes
Hadronic	3	0/4,0/5,1/4,1/5	yes

Table 2: CPM input channels that make up each JEM input

For an individual slice, it is in fact possible to derive ALL the other data from the CPM tower input readout — the JEM towers are just sums of the correct set of CPM towers (when the tower mapping is understood) and the algorithm results and data passing through the CMM are all calculated from either the CPM or JEM tower inputs. Below, the results of this analysis for each data type is presented.

3.1 JEM input trigger elements

The bulk of the JEM module data consists of the incoming LVDS data corresponding to two by two sums of calorimeter trigger cells. With 32 analogue inputs populated in the PPM, this means that eight JEM input elements should have been available (four in the EM layer, and four hadronic). However, there was a known problem with two of these eight inputs, and so these were disabled in hardware (meaning their results did not get used in the jet and energy algorithms) and these are also ignored for the purpose of this analysis. The correspondence of the eight JEM inputs to CPM inputs, as observed from the data, is shown in table 2. The CPM channels are labelled eta/phi where the eta and phi refer to the logical module eta, phi coordinates rather than any calorimeter coordinate system. Calorimeter coordinates are irrelevant for this analysis since no mapping between L1Calo trigger and calorimeter data is attempted. JEM input channels were all at the lowest eta, and phi is labelled 0–3 going up in phi.

The first thing to be noted from this mapping is that it is not quite the mapping that one would expect. Ideally, the JEM phi coordinate should increase in parallel with the CPM phi. The reason for this lack of correspondence is not in the cables themselves, but in a mismatch in the PPM and JEM specs as to the pin usage of the LVDS cable assemblies. This was first spotted before the testbeam, and causes the pairwise swapping in phi of the JEM inputs relative to the correct mapping.

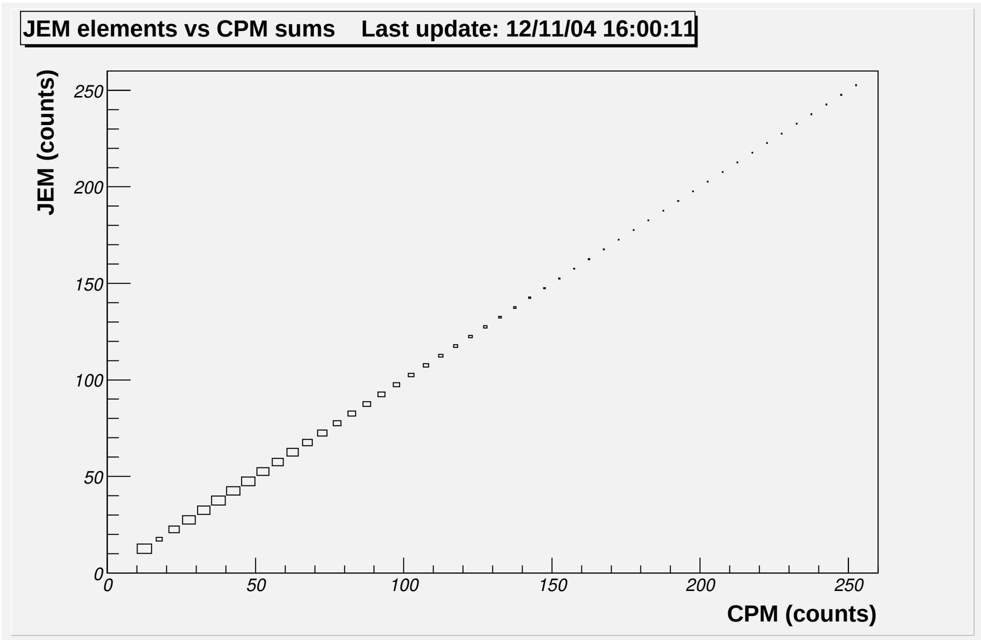

Figure 3.1: JEM input energies vs sum of corresponding CPM towers

Attempts were made to fix this in the PPM LVDS output FPGA, but the final mapping was almost as expected if this problem had not been fixed. There is however one more puzzling observation from the data — the swap did appear to be fixed for hadronic channels 0 and 1. It is not understood at this point why this should be.

Having established the relative mapping of CPM to JEM inputs, it is possible to compare on an event by event, slice by slice and channel by channel basis if the data seen by the JEM corresponds exactly to that seen by the CPM. The results of this can be seen in figure 3.1. In fact this histogram does not tell the full story, since it wouldn't show small discrepancies. A more detailed check of the data looking for any difference in the values shows that no problems occurred in any of the runs analysed. The only difference between CPM sums and JEM readout data is when the CPM sum is above 511 counts, the JEM element correctly saturates at 511. This proves that the PPM was providing the same data to both CPM and JEM and that the LVDS links, and all the algorithms associated with this data (e.g. BCMuxing, parity checking) was working very reliably. It should be noted that the analysis also checked if the LVDS parity error or link down flag was set on any slice for both the CPM and JEM inputs, and, for the channels that were actually connected and known to be good (32 for the CPM, 6 for the JEM), these errors were never seen.

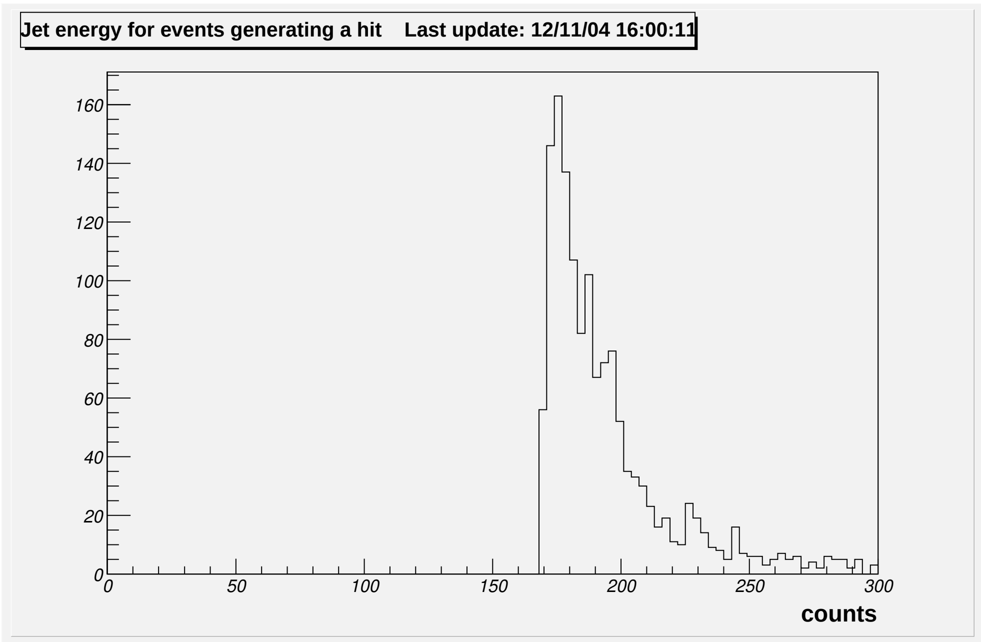

Figure 3.2.1: Jet Energy for events with a jet hit

3.2 Jet trigger hit outputs

The jet trigger was set such that only one threshold was enabled. For the majority of the runs, this threshold was set at 170, but for the final batch (runs 2101813 onwards) the threshold was set at 40. The other thresholds were, in theory, disabled by setting them at their maximum value of 1023. The behaviour of the trigger output bits were checked in the JEM data, which contains the results of both the Jet and Energy algorithms. In particular, it was investigated whether the hit bit was either set when it shouldn't have been (false trigger), or not set when it should (missed trigger). There was no example in any of the runs of a problem of this sort. This can be illustrated by the histograms in figures 3.2.1 and 3.2.2.

Clearly the Jet algorithm worked well throughout the testbeam period, but there was one aspect in which the trigger did not behave quite as required. With the threshold set at 170, events with an energy of exactly 170 fired the hit bit. This is not in line with the way the CP thresholds work, or the generally accepted scheme that L1Calo thresholds should only fire if the quantity is greater than the threshold [2]. This minor firmware glitch had also earlier been noticed in slice tests, but the corrected firmware had not been tested before the testbeam period. The minor discrepancy between actual and expected behaviour is reflected in the fact that setting the other thresholds to 1023 did not entirely switch them off. Occasionally (usually on Tile Calorimeter calibration events) the total energy in

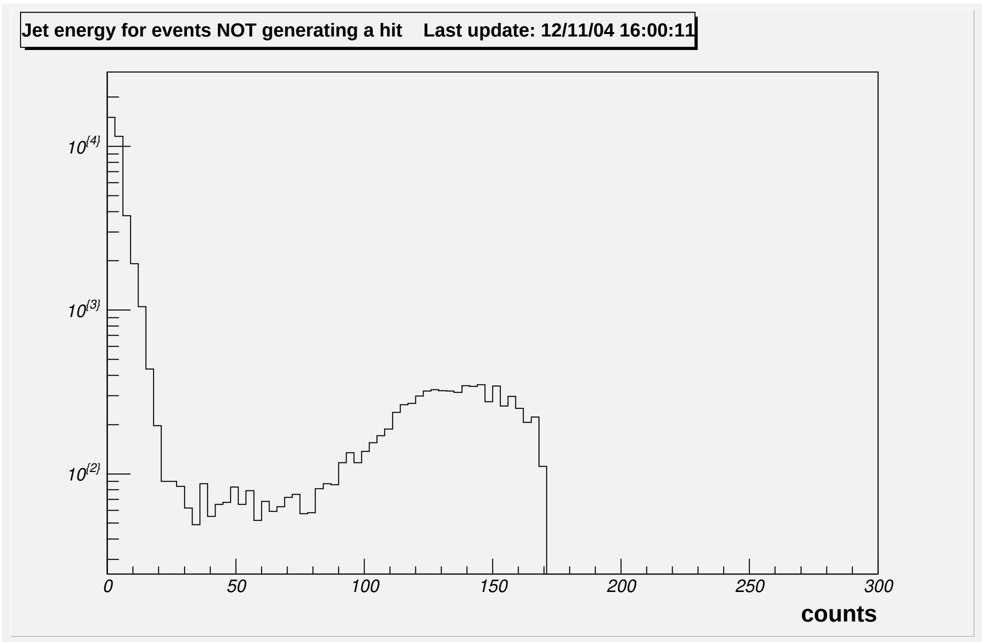

Figure 3.2.2: Jet Energy for events with no jet hit

the calorimeters would be very large and the jet elements would saturate. This caused the jet sum to be set to maximum (1023). For a threshold set at 1023, this should still not cause the hit bit to be set, but, in these cases, it was seen that all the jet thresholds fired, so again the incorrect 'greater than or equal' logic was being applied.

3.3 JEM energy sums

The energy sums are also available in the JEM data, and the firmware calculation of these sums was repeated in software on the six enabled JEM input elements. These calculated energies were compared against those in the readout for every slice of every run, and again, like the jet hits, no discrepancies were observed. The comparison was made for the three quantities, Ex, Ey and Et. This shows that the JEM energy algorithm was working reliably throughout the testbeam period. Note that in order to get perfect agreement, the exact geometrical coefficients must be used, and rounding performed to the right number of bits at exactly the right points of the calculation.

3.4 CPM hit outputs

The only major problem with the testbeam data was observed when looking at the results of the CP algorithm. It was found that for all runs, a large proportion (typically around 70%) of the events that should have fired some of the CPM hit outputs did not do so. This problem was traced to the fact that the CP chips had been loaded with firmware designed to work with the new (correct) MCM bunch-crossing multiplexing (BCMUX) logic. Unfortunately, the PPM at the testbeam was instrumented with the old MCMs which produce incorrect parity on events with no data. The consequence was that most of the data seen by the CPM was zeroed before entering the algorithm logic due to a bad parity sum. However, some data got through to the algorithm when, by chance, the BCMUX logic was reset by some parity-correct data.

This problem made the analysis of the CPM hits more difficult and a little imprecise. The BCMUX logic adds a history dependant element to the data seen by the the algorithm logic, so the data seen in previous slices has an important influence on whether data is seen by, or zeroed before the algorithm. Since five slices were read out, with two before the most important slice, some history is known, but unfortunately the BCMUX logic can depend on data much further back than just two slices. Without going into details, the upshot was that for some data, it is possible to say whether the data was zeroed or not, but for a significant proportion, we cannot know for sure just by looking at the input data.

The approach taken to cope with this problem in this analysis was:

- Zero all data *known* to be zeroed
- Pass all data *known* to be good
- For all other data, use clues in the data results to decide if the data was seen by the algorithm

This means that the expected hit results of the analysis, when it simulates the algorithm, are not unbiased — the analysis uses the results to influence the choices made in trying the reconstruct the parity zeroing of algorithm input data. Thus the scope of the tests is not really to check if the CP chip performed the algorithm correctly, but a rather weaker test that the results are not impossible given the known input data. Another point to note is that although the final step — a sort of educated guess at what actually happened — can be made quite sophisticated, it is difficult to always guess correctly.

Having constructed the data which was thought to be sent to the algorithm, the CP algorithm was then performed on this data to give the simulated hit results.

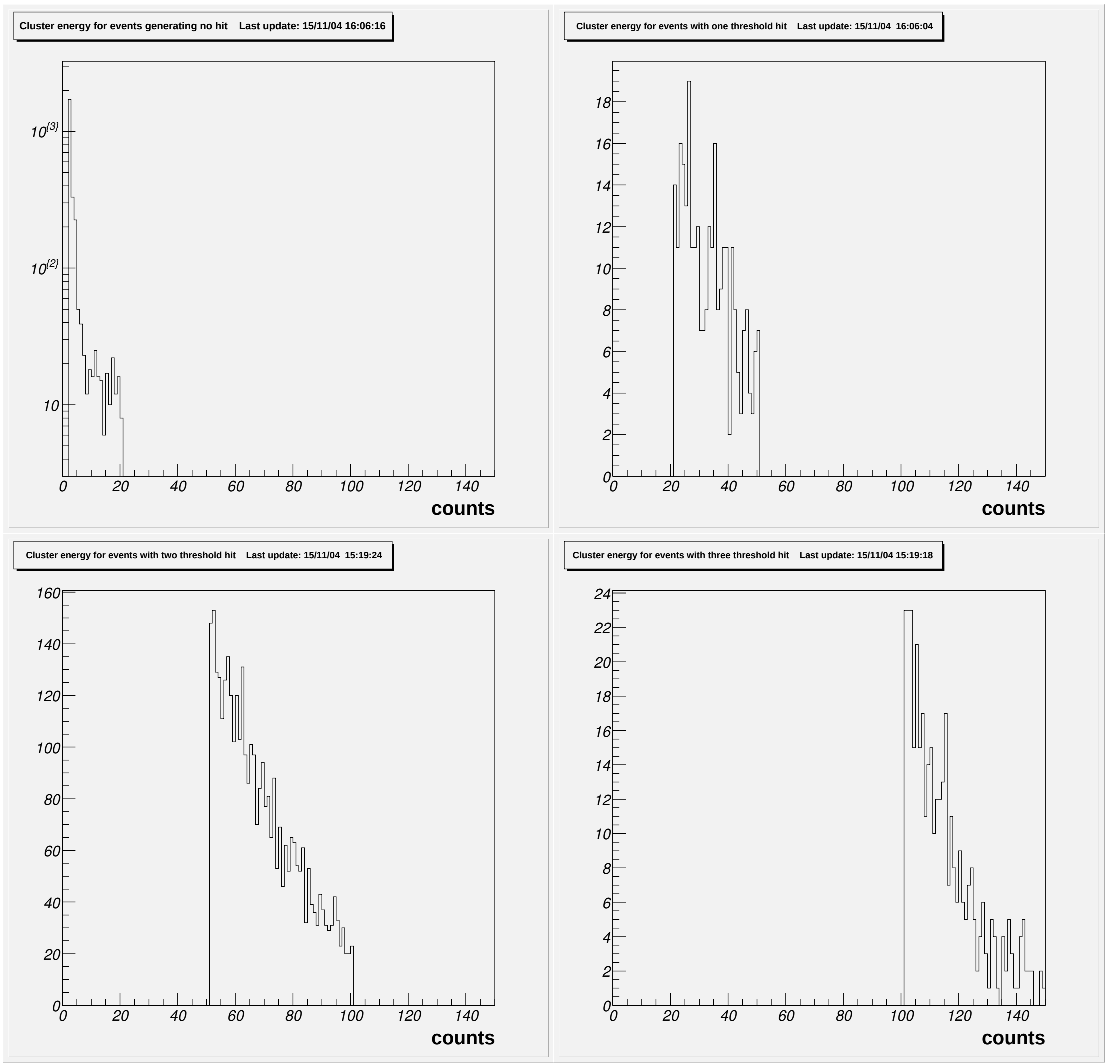

Figure 3.4: CP cluster energies for each threshold hit

These were then compared with the results seen in the data itself. As seen in Table 1, in this case the result data was shifted by two slices, so only three slices (one being the L1A slice) could be checked. The results can be displayed by plotting the energies which fired each trigger hit bit. Four bits were active, with the thresholds set at 20, 50, 100 and 200 counts, and the results can be seen in figure 3.4, which show the thresholds behaving as expected. There were a few anomalous events, but a more careful individual study of these rogue events revealed that they were just cases where the analysis had guessed wrongly about which towers were zeroed by the parity checking logic, and there were no examples of events which could not be explained by a suitable choice of channels to be zeroed.

3.5 CPM RoI data

As mentioned in section 2.3, the single CPM RoI slice read out on every event was set to point at the slice after the L1A. This meant that few, but greater than zero, RoIs were actually detected. However, for these RoIs, it was possible to check if they were consistent with the input data and the hits seen for that slice. RoIs are another result from the CP algorithm processing, and so, like the hits, are subject to the same problems associated to the BC-demuxing as described in the previous section. So the same qualification has to be applied, here we cannot really check that the algorithm was fully working, just that the results are at least consistent with a possible interpretation of the inputs.

Having applied the algorithm, as in the last section, the RoIs were formed as well as the hits. These were then checked against the data read from the CP-RoI ROD in all aspects. The data consists of the thresholds passed, the RoI location and error bits associated with the RoI. In all aspects, the RoI data was found to be correct, i.e. the number of thresholds passed matched with the hits seen, and the RoI location matched with the input data. The parity error bit was found to be always on, but this is not surprising given the fact that the majority of the channels going into the CP chip would be zeroed due to their apparent incorrect parity.

3.6 Energy CMM data

The JEM energy outputs were sent to the left hand energy CMM and the data passing through the CMM was also read out into the CMM–ROD. Since there was only one JEM, there was no merging to be done, but the this JEM energy should be seen at three points in the CMM — at the backplane input, the crate summing stage and the system summing stage. Unfortunately, not all of these were timed in correctly, but all five slices of the JEM data were seen at the system summing stage, so the analysis concentrated on this part of the CMM readout, since for this final stage to be correct, the other stages would also have to have been working. It was found that the energies matched precisely for all slices of all runs. Note that the Ey component at the CMM system stage was actually negative due to the JEM placement, but identical in magnitude to the JEM Ey energy component.

The CMM data output also consists of the Et and missing-Et result bits. There was thought to be a problem with the firmware for the missing-Et logic, but the sum Et logic was working, and used as a trigger for some runs. The threshold was set to 50 GeV for runs up to 2101664, and then 150 GeV thereafter. The behaviour of this output bit was investigated in the data, checking that the bit was set when expected given the input energies (derived initially from the CPM data). There was no problem found with this logic in any run, and the threshold behaviour

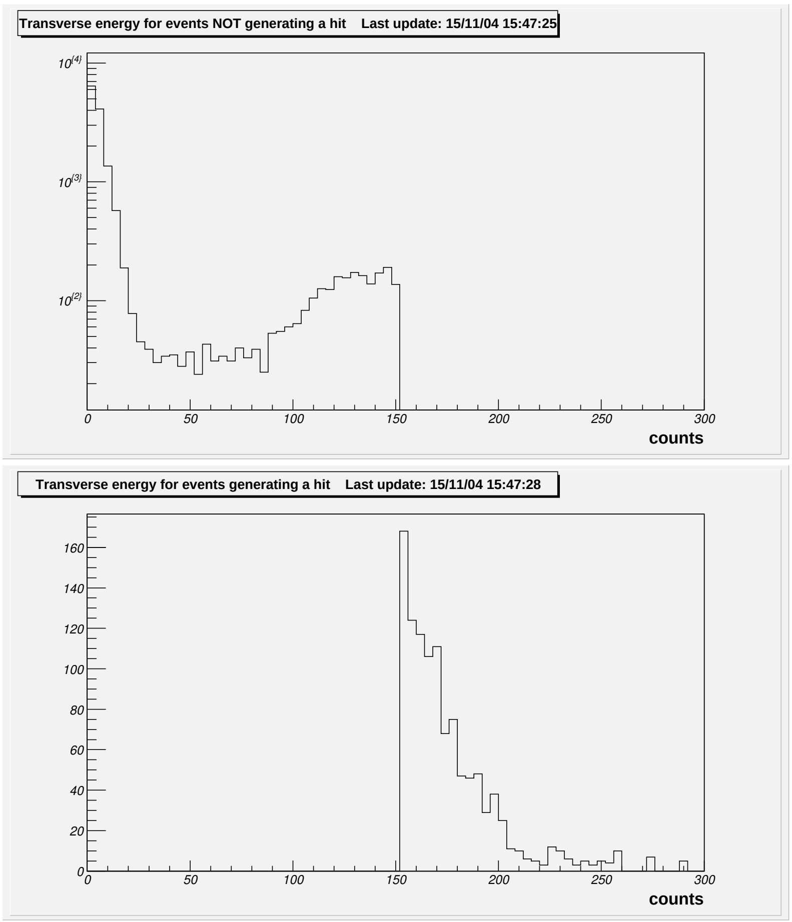

Figure 3.6: Transverse energies for CMM Et passed/failed

can be seen in figure 3.6. Like the jet trigger logic, this is another case where the trigger is fired if the the result is greater than or equal to the threshold. For consistency with other triggers, it may be better to change this to ‘greater than’.

Other than the actual data values, the analysis also looked at the flag data associated with the CMM data. Each flag was checked to see if it had the value it should have — in most cases the flag should be zero. For example, it was checked that none of the backplane inputs had a non-zero parity error flag, and indeed no events were seen with parity errors. In the case of the system flags, the situation is more complex, since non-zero flags were expected for overflow events when the JEM inputs saturated (for example on Tile calibration events). A problem was found in this area that the flags were not located exactly where they were expected from the specifications. This is probably partly due to a conflict in rapidly changing specifications (for example there are some conflicts between the CMM and ROD specifications of the CMM glink format [3][4]). Once the correct location was found, the flag bits did seem to be set appropriately. However, another strange observation was that one of the flag bits (for system Ex data) seemed to be flagging if the Ex was non-zero, rather than anything more dramatic. The behaviour that was not quite as specified is spelt out below:

- System Ex flag at bit 15 of glink pin 18: should be overflow bit, but actually indicates if Ex is non-zero
- System Ex flag at bit 16 of glink pin 18: should be zero, but actually is the overflow bit
- System Ey flag at bit 32 of glink pin 18: should be overflow bit, but actually is zero for all data
- System Ey flag at bit 33 of glink pin 18: should be zero, but actually is the overflow bit
- System Et overflow flag appears at bit 33 of glink pin 13 rather than at either of the two conflicting specifications in the CMM and ROD specifications

3.7 CP/Jet CMM data

In a similar way to the JEM energy data, the CPM and JEM hits could be followed through the various stages of the second CMM processing to check if the CMM handled the hits correctly. Both individual module inputs were checked at the backplane input level, and then these were also checked against the crate-level and system-level sums. The summing itself was essentially trivial for two reasons:

- The thresholds were assigned 4 to the JEM, 4 to the CPM, so there should have been no overlap, except in the cases noted above when all jet thresholds fired.
- The latency of the CPM was three ticks greater than the JEM (as seen in table 1) so CPM hits for the same event appeared later in the CMM data.

The readout offsets could never be ideal for both sets of data, and were also very different for some runs. This meant that it was in general impossible to check that all slices were seen by the CMM correctly, but for the majority of runs, the L1A slice could be checked for both the CPM and JEM in the system level readout.

No problems were seen in the transmission of any of the hit data to this CMM. The error flags (such as backplane parity error detection) were also investigated, and again, there was no evidence of any flagged errors in any of the runs.

3.8 Conclusions

A substantial quantity of consistency checking has been performed on the L1Calo data taken at the 2004 testbeam. From looking at the CPM input data, it was possible to derive what should have been read out by almost all of the other elements of the system (i.e. CPM results, JEM and CMM data). This predicted output was checked against the actual readout and found to match extremely well in most cases. For example, the CPM and JEM input data matched perfectly for all channels known to be good. The threshold behaviour of the output trigger bits was also investigated and found to work well.

Only one major error was found, which was due to the loading of an inappropriate version of the CP chip firmware. Otherwise, only small differences were found between expected behaviour and actual hardware performance, and these are mostly understood and hopefully can easily be fixed by small changes in the firmware. Note however that many important aspects of the L1Calo system cannot be addressed by the this type of analysis of testbeam data. In particular only the last step of the PPM can be checked (the formation of LVDS outputs) and the high speed data transmission on the common processor backplane was not used at all in the testbeam setup.

References

[1] ATLAS testbeam 2004
`http://hepwww.ph.qmul.ac.uk/~landon/atlas/testbeam`

[2] ATLAS L1Calo Group, ATLAS Level–1 Calorimeter Trigger Algorithms, ATL-DAQ-2004-011
`http://cdsweb.cern.ch/search.py?p=ATL-DAQ-2004-011&f=wrn`

[3] I.P.Brawn, C.N.P.Gee, ATLAS Level–1 Calorimeter Trigger Common Merger Module, ATL-DA-ES-0021
`https://edms.cern.ch/document/321069`

[4] L1Calo Group, ATLAS Level–1 Calorimeter Trigger Read-out Driver
`http://hepwww.rl.ac.uk/Atlas-L1/Modules/ROD/ROD_spec-9U-version1_0.pdf`