VELO Module Production – Laser Test and Noise Analysis

Technical Note

Issue: 1
Revision: 0

Reference: LHCb-2007-083
Created: May 24, 2007
Last modified: November 14, 2007

Prepared by: Liverpool VELO Group[a]
Authors: Kurt Rinnert, Tony Affolder, Gianluigi Casse, Torkjell Huse, Girish Patel, Tony Smith, Mark Tobin
Editor: G.D. Patel

[a]Liverpool VELO Group: T. Affolder, T.J.V.Bowcock, J.L. Carroll, G.L. Casse, P.A. Cooke, L. Dwyer, K. Hennessey, T. Huse, D. Hutchcroft, D.E.L. Jones, M. Lockwood, D. Muskett, G.D. Patel, K. Rinnert, T. Shears, T. Smith, G. Southern, P. Sutcliffe, M. Tobin, P.R. Turner, M. Whitley, M. Wormald

Abstract

The work flow of the LHCb VELO module production comprises many steps. Some of them are hard or impossible to undo and very expensive in either time, involved components, or both. It is therefore essential to detect problems as early as possible to avoid wasting resources. In addition even production grade modules that pass all tests are still allowed to have a small number of problematic channels. These channels have to be identified and categorised. This note describes the algorithms used to detect problems by analysing datasets taken at different stages of module production.

Document Status Sheet

1. Document Title: VELO Module Production – Laser Test and Noise Analysis			
2. Document Reference Number: LHCb-2007-083			
3. Issue	**4. Revision**	**5. Date**	**6. Reason for change**
Release	1	May 24, 2007	Finish for first release.

Contents

List of Figures

List of Tables

1 Introduction

The work flow of the LHCb VELO [1] [2] [3] module production comprises many steps. Some of them are hard or impossible to undo and very expensive in either time, involved components, or both. It is therefore essential to detect problems as early as possible to avoid wasting resources. In addition even production grade modules that pass all tests are still allowed to have a small number of problematic channels. These channels have to be identified and categorised. The results of each test as well as the final channel categorisation is entered into the module production data base [4] before production proceeds to the next stage.

To this end each module is connected to a DAQ system and data is taken at various stages of its production. Two different DAQ systems are used: an NA60 system [5] and a TELL1 board [6]. Apart from the FPGA firmware the latter is the same system that will be used in the LHCb experiment. Table 1 lists the production stages at which the data is taken and the DAQ system used.

Production Stage	DAQ
Bare Hybrid	NA60
Front End Bonded (FEB)	NA60
Sensor Bonded	NA60
Vacuum Test of Complete Module	TELL1

Table 1 Production stages at which data is taken and the DAQ systems used

The differences in the output data format between the NA60 and the TELL1 made it necessary to use two different frameworks for the implementation of the algorithms. For the NA60 data a lean standalone framework was implemented from scratch. It makes use of a stripped down, performance enhanced version of the binary data decoder from the old VELO test beam software [7] [8]. For the analysis of the TELL1 data an algorithm is plugged into the Gaudi [9] application Vetra.

Different algorithms are employed to analyse the various data sets. A noise and pedestal analysis is performed on all data sets. At the Sensor Bonded stage a laser scan of the sensor is performed. This allows to look for problematic strips by analysing signal data. For all NA60 data the pipeline column numbers (PCN) of the Beetle chips [10] are checked for consistency and a header spill-over correction is applied.

The algorithms, their output as well as its interpretation are described in more detail in the following sections.

2 PCN Decoding and Spillover Correction on the NA60 Data

Each Beetle chip [10] handles 128 input channels, each of which corresponds to a strip on a silicon sensor. It can be operated in different configurations, affecting the number of readout lines used. For the VELO module readout the output data is distributed over four analogue readout lines, each of which corresponds to 32 channels. The chip stores the charge collected from the silicon sensor strips in a pipeline with 187 columns. When it receives a trigger from the readout board, the analogue data corresponding to the selected column is shipped to the readout lines. The first four signals on a readout line do not correspond to any of the 32 data channels. These non-data signals are called the *chip header*. They encode binary information useful for checking the chip status and the consistency of the data. Most notably the two signals closest to the first channel on each readout line encode two bits of the pipeline column number (PCN). These bits and the PCN parity bit are relevant to the analysis of the data obtained with the NA60 DAQ system.

It is desirable to decode and use the information in the chip header for two reasons:

- consistency checks
- correction of header spill-over into the data stream

There are two kinds of consistency checks: consistency of the output from a single chip and consistency of the data across chips. For the former the PCN parity bit is compared to the parity computed from the decoded PCN. A mismatch at this level is a signature for problems on at least one readout line on a single chip. Inconsistencies between PCNs across chips for the same trigger suggest problems with either the DAQ system or at least one of the chips. These checks can be performed even in the absence of a silicon sensor, allowing for an early detection of problems on a populated hybrid.

The problem of header spill-over arises due to the way the digital header information is encoded on the analogue readout line. Each bit is encoded by either a high charge (encoding 0) or a low charge (encoding 1). Since the digital information must be decodable without any ambiguities, there must be a considerable gap between the distribution encoding a high and a low bit. This leads to a high signal variation that unavoidably affects the following signal, resulting in a significant increase in the raw noise of the first data channel on a readout line. Decoding the chip header provides a handle to correct for this effect.

In order to distinguish high and low bits in the chip header one has to cut on the ADC count of the signal somewhere between the two charge distributions. The determination of the cuts for the header decoding turned out to be non-trivial. Since there is also spill-over from one header bit to the next inside the header, a whole tree of cuts has to be found. Furthermore, it was found to be impossible to find a tree of cuts that is valid for all data taking runs on the NA60 DAQ system. Hence a dynamic solution was implemented. It finds the cuts by looking for the gap between the high and low bit ADC distributions in histograms of ADC counts. The histograms are filled from an ntuple kept in memory, using the already determined cuts when descending down the tree. After some tuning, this approach has proved to be very robust.

In a second analysis pass the cuts found in the first are used to decode the PCNs and prepare the spill-over correction. Essentially it creates a two dimensional histogram of pedestals vs. PCN and channel number. In the final noise analysis, this is then subtracted from the raw ADC values[a].

Any PCN mismatches between chips or parity mismatches on a single chip are reported immediately during this stage. This gives the operator the opportunity to investigate whether they are caused by the DAQ or are genuine problems on the hybrid.

3 Noise Analysis on the NA60 System

3.1 Purpose

Detecting possibly problematic hybrids as early as possible by looking for:

- unusual noise in a chip or hybrid circuit
- chip problems showing themselves in PCN mismatches or dead pipeline columns

To this end the analysis is performed several times during production, at different stages of bonding [11] [12] [13]:

1. Bare Hybrid
2. Front End Bonded (FEB)
3. Sensor Bonded

3.2 Algorithms

The raw noise, pedestals and CM corrected noise per channel are computed per chain the final analysis pass[b]. As described in section 2, the raw ADC counts are corrected for the header spill-over by subtracting a correction matrix. The raw noise is simply the RMS of the raw ADC distribution on each channel. The pedestal is computed as a simple running average:

$$\mathrm{ped}_{n+1} = (\mathrm{ped}_n \cdot n + \mathrm{adc}_{n+1})/(n+1)$$

For the R sensors the CM is computed per event as the average of the ADC values of all 32 channels on a Beetle readout line. For the Phi sensors the situation is slightly more complicated: here the 32 channels are split into three categories. The categories are inner strip, outer strip and outer strip with routing line. The CM is then calculated separately for each category.

[a] Note that this correction does not completely solve the spill-over problem. This is due to the fact that it ignores the width of the high and low ADC distributions in the header bits. However, the noise is considerably more well behaved after applying the correction.

[b] The whole data sample is only read once before the first analysis pass and kept in memory afterwards. It was observed that the data written at the very beginning of a data taking run on the NA60 system is often corrupted. To work around this DAQ instability the first few events are skipped by all algorithms.

```
# This file was automatically generated by
# $Id: NoiseMaker.cpp,v 1.20 2006/09/19 14:24:30 rinnert Exp $
# running as user huse on LHCbFE.ph.liv.ac.uk
# Thu Feb 22 13:27:44 2007
# Spillover Corrections: ON
#
# strip chip channel      noise  pedestal  raw noise
   1663   03     000   1.457029   132.520   1.470639
   1662   03     001   1.451097   133.256   1.454617
   1661   03     002   2.351887   133.665   2.414000
   1660   03     003   1.440434   132.124   1.450379
   1659   03     004   1.407830   133.809   1.421839
```

Figure 1 Output format of the NA60 noise analysis

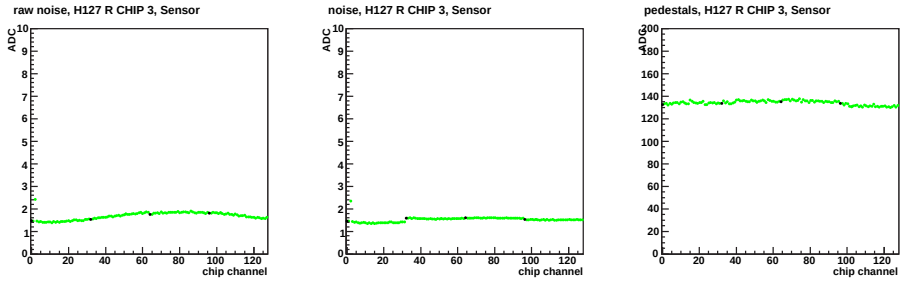

Figure 2 Example of NA60 noise and pedestal plots. The black dots mark the first channels on a readout line. The fact that they do not show higher noise than the others shows that the spillover-correction works well and the DAQ setup is stable.

Once the pedestals and the CM are available, the corrected noise per channel is calculated as the RMS of the corrected ADC counts:

$$\mathrm{adc}_n - \mathrm{cm}_n - \mathrm{ped}_n$$

No further analysis of the noise data is performed at this stage. I.e. even for the sensor bonded hybrids the noise data is not used to single out problematic channels. This is the purpose of the laser test analysis described in the following section.

3.3 Output

The main output of the NA60 noise analysis program are plain text data files of the format shown in Fig. 1.

These files are automatically copied to where the production data base expects them for reading.

The analysis program also produces a number of plots in EPS and PNG format, like the one shown in Fig. 2. These can be downloaded from the web, e.g at
http://hep.ph.liv.ac.uk/VELO_DATA/Hybrid127/R/Results/Sensor/plots/

3.4 Usage

The program is invoked on the command line via a wrapper script. The script has numerous options, most of which are only relevant for debugging purposes or expert use. A typical invocation would simply look like this:

```
noiseped -hybrid 127 -sensor R -status sensor
```

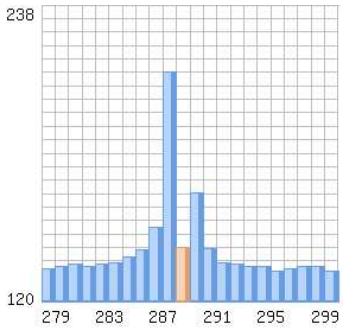

Figure 3 Dead strip found by dip search

Further documentation can be found at
http://hep.ph.liv.ac.uk/~rinnert/noiseanalysis.html

4 Laser Test Analysis on the NA60 System

4.1 Purpose

The purpose of the laser test analysis is to find dead/open or low gain channels, allowing to compare the results with the information in the production data base. The laser test is also important for verifying the correctness of the bonding pattern at the start of production.

4.2 Algorithms

Three different algorithms are employed to find problematic channels in the laser test data. In order of their precedence these are

1. search for significant dips in the signal region around the laser position
2. combination of threshold and behaviour of neighbouring strips
3. search for significant dip in a histogram averaged over several laser positions

The first algorithm searches for significant dips adjacent to the maximum ADC count in the signal region. Significance is defined as a multiple of the average noise on the data sample, the default is 4σ. A strip is required to show this behaviour in 90% of the cases to be considered as problematic. In edge cases (i.e. if the strip under consideration has only one neighbour), a much higher significance of 25σ is required. This may seem ridiculously high, but lower values lead to a flood of false alarms due to fluctuations. If a strip is considered problematic according to the above *and* never was the strip with the highest ADC count in any event, it is flagged as dead. As an example of a strip flagged by this algorithm, Fig. 3 shows a plot from the production data base.

The dip search algorithm is blind to adjacent bad strips by construction. Hence other criteria must be found to handle these cases. A simple, yet efficient, approach is to look for groups of neighbouring strips all of which never pass a certain absolute ADC threshold and never were the strip with the maximum ADC count in any event. The default value for this threshold is 160 ADC counts. As opposed to the dip search, the strips at the boundaries of this group do not enter the decision. I.e. this is not a search for a broad local minimum. In practise it turned out that groups of strips with this behaviour indicate problems with the DAQ rather than genuine problems with the hybrid. An example is shown in Fig. 4.

The dip search algorithm described above is very pure. The (expected) trade-off is a limited efficiency, even at the optimal working point. An algorithm with complementary features is the search for minima

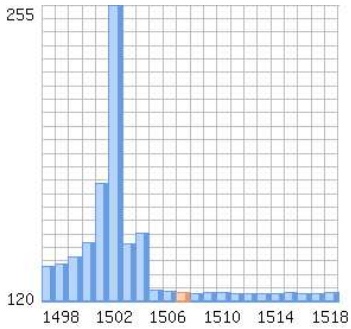

Figure 4 Consecutive strips with no signal

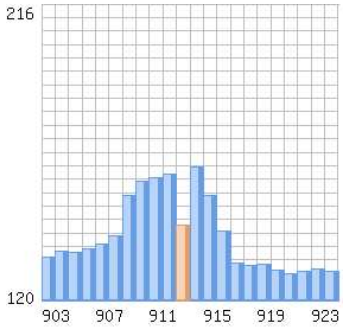

Figure 5 Bad strip found by averaged histogram method

in an averaged histogram. This works as follows. Assume we want to examine strip n. The algorithm then produces histograms of the ADC counts for laser positions $\mathrm{n}-\mathrm{k}$ to $\mathrm{n}+\mathrm{m}$, averaged over several events. Next these histograms are added and normalised to yield an overall average histogram of ADC counts around the signal region for laser position n. The algorithm then checks whether strip n is a local minimum in this histogram. For good strips this should never be the case. The minimum is required to have a certain significance. Here the cut is defined in absolute ADC counts, the default is 9. This cut is applied asymmetrically: if the significance is high compared to strip $\mathrm{n} \pm 1$ half the significance is required for the difference to $\mathrm{n} \mp 1$. As in the case of the dip search algorithm, a much harder cut of 24 ADC counts is applied for strips that only have one neighbour in the data sample. If a strip was found to be a local minimum in the average histogram *and* never passed a certain ADC threshold in any event, it is considered dead or low gain. Reasonable values for this ADC threshold depend on the amplitude of laser and then DAQ setup. The value is therefore often changed by the operator. Typical values range from 170 to 220. An example is illustrated in Fig. 5.

The algorithm is capable of finding dead strips missed by the direct dip search algorithm. However, does not only find dead/open strips but also strips with low gain that one would still consider perfectly usable. The decision between the two cases is made during visual inspection of the analysed data, see section 6.

Originally, the laser test analysis also featured a very simple short search algorithm. After we learnt that this algorithm never produced reliable results it was dismissed. More sophisticated ways for automatic short detection were investigated. All algorithms we tried out either suffered from high impurities or inefficiencies. Of course it is not impossible to find a reasonable algorithm. However, instead of wasting time and resources on this problem, we decided to rely on the visual inspection of the data to detect shorts as described in section 6.

4.3 Output

The output of the laser test analysis is a plain text datafile. The file format is illustrated in Fig. 6.

```
# This file was automatically generated by ltana
# ($Id: ltana,v 1.31 2006/08/10 16:37:53 rinnert Exp $)
# using algorithm
# $Id: ShortNDead.cpp,v 1.42 2006/08/10 16:08:55 rinnert Exp $
# running as user huse on LHCbFE.ph.liv.ac.uk
# Fri Aug 11 11:02:15 BST 2006
#
# Flags:
#    0 : OK
#   -1 : Found to be bad by steep dip in signal or edge drop (dead)
#   -2 : Found to be bad by neighbour/maximum algorithm
#   -3 : Found to be bad by average histogram method (dead or low gain)
#
# strip chip channel flag  [shorted strips]
   0000   12     000    0
   0001   12     001    0
   0002   12     002    0
```

Figure 6 Output format of the NA60 laser test analysis

The file is written to the proper location where the production data base expects it for reading. As reflected in the legend of the data file itself, the strips are flagged in the following way.

- -1 : found to be dead by direct dip search algorithm
- -2 : strip and at least one of its neighbours never exceeded minimum ADC threshold
- -3 : found to be dead or low gain by average histogram method
- 0 : OK

Here the flags are given in order of precedence: if a strip is flagged -1 it will not be inspected by the following algorithms, i.e. it will never be flagged -2 or -3. Similarly, if it is flagged -2 it will not be considered by the average histogram search algorithm and hence never be flagged -3. If a strip is not flagged by either algorithm it is OK and flagged 0.

4.4 Usage

The laser test analysis program is invoked on the command line via the wrapper script '`ltnana`'. The script has numerous options to control the behaviour of the various algorithms. The most simple invocation looks like this:

```
ltana -hybrid 70 -sensor R
```

Unfortunately, however, the optimal cuts for the various algorithms depend on the amplitude of the laser during the scan. This means that the person performing the laser scan usually has to specify more than the minimum number of options.

Further documentation can be found at
http://hep.ph.liv.ac.uk/~rinnert/lasertest.html

5 Noise Analysis on Burn-in Data with TELL1 DAQ

5.1 Purpose

The purpose of the burn-in data analysis is to spot possible problems occurred due to module handling or during the burn-in test itself. For instance channels rendered noisy or bad during the thermal cycle. It also serves as an independent confirmation of the laser test findings.

```
# This file was automatically generated by
# $Id: NoiseShortNDead.cpp,v 1.18 2006/07/04 17:08:23 rinnert Exp $
# running as user rinnert on LHCbFE.ph.liv.ac.uk
# Thu Aug 31 14:48:23 2006
#
# Flags:
#    0 : OK
#   -1 : Found to be bad due to high noise
#   -2 : Found to be bad due to very low noise
#   -3 : Found to be bad due to high raw noise
#   -4 : Found to be bad due to very low raw noise
#
# strip chip channel flag [shorted strips || significance ]
   0000   12     000    0
   0001   12     001    0
   0002   12     002    0
   0003   12     003    0
```

Figure 7 Output format of the TELL1 noise analysis

5.2 Algorithms

Only one algorithm is employed to analyse the burn-in data. It calculates the raw noise, pedestals and CM corrected noise and then looks for channels with exceptional noise behaviour. The pedestals and noise are computed the same way as in the NA60 noise analysis descried in section 3. There is no spill-over correction implemented in this analysis . For this reason the first channel in each readout line is ignored by the algorithms that flag strips as problematic. Note that this does not mean we are completely blind on these channels. Firstly the laser test analysis is as efficient on these as on any other channel. Secondly we have to inspect the output plots by eye anyway and the modulo 32 channels are clearly marked on them. It is therefore not hard to spot significant changes on these channels that might be due to handling or thermal cycling.

After the raw noise as well as the CM and pedestal corrected noise is calculated for each channel, the algorithm searches for channels with significantly high or low noise. The significance is defined as a multiple of the RMS of the noise distribution in a group of channels used for the CM computation. That is, 32 channels on a readout line for the R sensors and the channels in one category for the Phi sensors.

Four cases are considered: low or high noise in the raw noise and low or high noise in the corrected noise. The significance cuts are different in each case. For the corrected noise the significance is defined in multiples of the RMS of the noise distribution, the default is 5 RMS for high and 5 RMS for low noise. For the raw noise the cuts are defined in absolute ADC values, the default is 1.2 for high and 0.8 for low noise. Channels with low noise correspond to dead/open or low gain channels. Exceptionally high noise indicates simply a noisy channel or possibly a short. This decision is made during the visual inspection of the data described in section 6.

5.3 Output

The output of the TELL1 noise analysis is again a plain text data file of the format shown in Fig. 7.

The following flags are assigned by this algorithm.

- -2 : low noise (dead/open or low gain)
- -1 : high noise (noisy or shorted)
- -4 : low raw noise (dead/open or low gain)
- -3 : high raw noise (noisy or shorted)

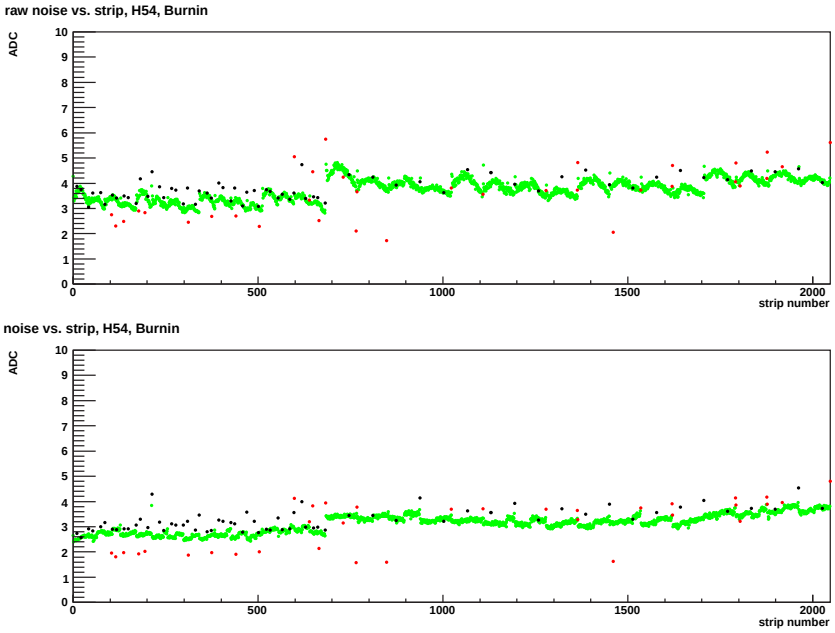

Figure 8 TELL1 noise analysis summary plots. Red dots indicate channels flagged as problematic by the algorithm. Black dots mark the first channel on a readout line. The latter are affected by the header spill-over and are ignored by the algorithms that flag problematic strips.

- 0 : OK

Again the flags are listed in order of precedence as in section 4. The output files are automatically placed where the production data base expects them for reading.

In addition to the summary data file the analysis program also produces per-chip noise and pedestal data files as well as some plots. These have the same format as in the NA60 noise analysis. But there are also summary plots over the whole strip range that can not be produced from the NA60 data. Examples of these are shown in Fig. 8.

5.4 Usage

Since the TELL1 DAQ system tends to be more stable than the NA60 system there is no tuning of cuts required when running the analysis. It is usually sufficient to run the wrapper script '`binoiseped`' with the minimal options:

```
binoiseped -hybrid 54 -sensor Phi
```

Further documentation can be found at
http://hep.ph.liv.ac.uk/~rinnert/noiseanalysis.html#runningburnin

6 Visual Inspection of Analysis Results

6.1 Purpose

The various algorithms used to find problematic strips in the laser test and burn-in data are neither perfectly efficient nor pure. Obviously, no algorithm ever can be. While there are ways to improve the existing algorithms, doing so would require a lot of time. We consider this a waste of resources

since we can not avoid to do a visual inspection of the analysed data anyway. It is necessary to decide whether inconsistencies indicate genuine problems (e.g. module damage due to handling) or represent algorithmic imperfections. In addition, the classification of bad strips can not be fully automated.

6.2 Procedure

The visual inspection of the analysis results is performed by two or more physicists with experience in the field. The data from all production stages is reviewed. We start from a summary page generated from the production data base. If this shows any inconsistencies between the strip evaluation at various production stages these are investigated by visual inspection of all available plots. After discussing the available data, we agree on the classification of each strip that was listed as possibly problematic in the data base.

Even if there are no inconsistencies the noise plots from the burn-in test are visually inspected. We use these because they represent the our latest knowledge about a module. In case there is something suspicious, the laser data in the area around the strip in question is inspected. Thus we can spot previously undetected problematic strips and classify them properly. It has to be said, however, that this is rarely necessary.

After this inspection session the strip flags in the data base are locked by the data base maintainer. The following flags can be assigned to a strip by this procedure:

- OK
- noisy (problematic, but usable)
- low gain (problematic, but usable)
- open (not bonded, bad)
- pinhole (also not bonded, bad)
- dead (bonded but not working, bad)
- short (shorted with another strip, bad)

7 Results

The results of the test procedure are summarised in Appendix A. Most notably the overall fraction of problematic strips is only 0.58%. This is considerably lower than the design limit of 1%. On a per-module basis the number of bad strips varies as shown in Fig. 9. Note that the quality of the modules improved during the production process. The differences between laser, burn-in and verified bad strip numbers visible in Fig. 9 are expected. The laser test was tuned to be over-efficient in order to reliably alert us of problems before the module is completed. The bad strip search on the burn-in data on the other hand was mostly geared towards detecting possible inconsistencies in the readout between the two DAQ systems and severe damage due to handling (e.g. broken bonds). The number of verified bad strips is then expected to lie in between the results of the two analyses. It is evident from Fig. 9 that this algorithm tuning also improved over time: in the later production period the numbers always follow the expected pattern described above.

Appendix B contains illustrations of the correlations among the various algorithms. We did not observe a single case where a bad strip could not be easily identified by noise analysis and visual data inspection alone. Especially it is possible to tune a bad strip search on noise data to be as efficient as a laser signal scan. Given that the laser test is a very time consuming procedure, it was therefore decided to drop it for the production of the VELO replacement modules which is due to start in 2008.

The results also show that the burn-in noise analysis is over-efficient in flagging noisy strips. This is on purpose. It is easier to spot a highlighted channel in the output plots and decide it is not noisy than the other way round.

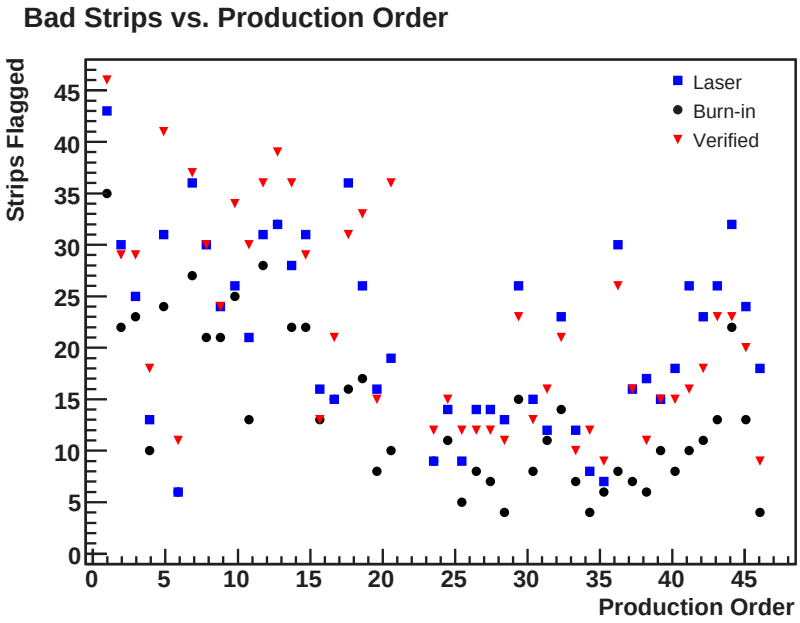

Figure 9 Bad strip flags vs. production order.

8 Conclusion

The consistency check algorithms implemented for the NA60 DAQ data reliably detect problems during early stages of the module production. They also were of great help in finding a stable configuration for the NA60 DAQ system. The consistency of the laser test and noise analysis results from the burn-in show that it is possible to detect bad channels by only looking at the noise. This allows to drop the rather time consuming laser test for the production of the replacement modules due to start in 2008. The results produced by the various algorithms in conjunction with the visual inspection of the analysis output, constitute a robust assessment of the electronic quality of the VELO modules produced in Liverpool.

9 References

[1] *LHCb : Technical Proposal*. Tech. Proposal. CERN, Geneva, 1998. CERN-LHCC-98-004.

[2] The LHCb Collaboration. *LHCb VELO (VErtex LOcator) Technical Design Report*. Number 5 in Technical Design Report LHCb. CERN, Geneva, 2001. CERN-LHCC-2001-011.

[3] The LHCb Collaboration. *LHCb Reoptimized Detector Design and Performance Technical Design Report*. Number 9 in Technical Design Report LHCb. CERN, Geneva, 2003. CERN-LHCC-2003-030.

[4] Girish Patel. LHCb Velo Data Base. Technical report, CERN, Geneva, Nov 2007. LHCb-2007-088. CERN-LHCb-2007-088.

[5] A. David for the NA60 Collaboration. The NA60 microstrip telescope readout electronics. *Nucl. Instrum. Meth. A*, (509):118–126, 2003.

[6] F Legger, A Bay, G Haefeli, and L Locatelli. TELL1: development of a common readout board for LHCb. Technical report, CERN, Geneva, Nov 2004. LHCb-2004-100. CERN-LHCb-2004-100. 1-2.

[7] J Palacios. Velo testbeam software reference page. `http://lhcbproject.web.cern.ch/lhcbproject/velo/testbeam/sw-ref.html`, July 15 2005. Last checked April 27 2007.

[8] Doxygen [14] Documentation of class **`BinDataUnpacker`**. `http://lhcbproject.web.cern.ch/lhcbproject/velo/testbeam/software/doc/VeloTB/v10r2/class_bin_data_unpacker.html`, July 15 2005. Last checked April 27 2007.

[9] The Home Page of the Gaudi Framework Project. `http://proj-gaudi.web.cern.ch/proj-gaudi/`, April 27 2007. Last checked April 27 2007.

[10] Löchner, S and Schmelling, M. The Beetle Reference Manual - chip version 1.3, 1.4 and 1.5. Technical report, CERN, Geneva, Nov 2006. LHCb-2005-105. CERN-LHCb-2005-105.

[11] Whitley, M and Wormald, M. VELO Module Production - Back End Bonding. Technical report, CERN, Geneva, Nov 2007. LHCb-2007-078. CERN-LHCb-2007-078.

[12] Whitley, M and Wormald, M. VELO Module Production - Front End Bonding. Technical report, CERN, Geneva, Nov 2007. LHCb-2007-079. CERN-LHCb-2007-079.

[13] Wormald, M and Whitley, M. VELO Module Production - Sensor End Bonding. Technical report, CERN, Geneva, Nov 2007. HCb-2007-080. CERN-LHCb-2007-080.

[14] The Home Page of the Doxygen Documentation Tool Project. `http://www.stack.nl/~dimitri/doxygen/index.html`, April 25 2007. Last checked April 27 2007.

A Summary of Bad Strip Flags

Hybrid	Module	Laser bad	Burn-in bad	Laser and Burn-in bad	Laser and Burn-in overlap %	Verified bad	Burn-in noisy	Verified noisy
53	23	25	23	22	96	29	6	3
55	24	13	10	7	70	18	5	1
56	27	36	27	23	85	37	10	0
60	26	6	6	5	83	11	5	2
64	31	26	25	22	88	34	26	0
67	37	31	22	21	95	29	5	3
68	29	30	21	20	95	30	8	1
69	25	31	24	24	100	41	15	0
70	21	43	35	32	91	46	9	4
71	28	30	22	21	95	29	11	3
73	33	31	28	25	89	36	3	2
75	41	15	15	8	53	21	6	6
78	30	24	21	19	90	24	4	0
79	32	21	13	9	69	30	10	0
82	35	32	32	28	88	39	12	6
83	44	26	17	15	88	33	10	1
84	36	28	22	16	73	36	8	1
85	38	16	13	9	69	13	6	6
88	45	16	8	5	63	15	2	2
89	47	19	10	10	100	36	16	0
93	42	36	16	12	75	31	17	2
94	61	8	4	3	75	12	4	2
95	54	13	4	4	100	11	4	0
96	53	14	8	6	75	12	5	1
97	52	9	5	3	60	12	8	2
99	58	12	11	7	64	16	0	0
104	50	9	9	6	67	12	8	2
105	51	14	11	7	64	15	1	5
106	55	26	15	14	93	23	0	0
107	56	15	8	7	88	13	1	0
109	57	14	7	7	100	12	4	0
112	59	23	14	12	86	21	3	1
113	62	7	6	6	100	9	0	0
114	63	30	8	7	88	26	7	2
115	64	16	7	7	100	16	6	1
116	7	32	22	20	91	23	1	3
118	74	26	10	10	100	16	1	3
119	71	17	6	6	100	11	0	0
120	72	15	10	10	100	15	4	1
121	60	12	7	7	100	10	0	0
125	73	18	8	7	88	15	3	2
126	75	23	11	10	91	18	2	1
127	4	26	13	9	69	23	3	3
134	6	18	4	3	75	9	1	1
135	5	24	13	11	85	20	1	0
Average		21.2	14.0	12.0	84.8	22.0	5.8	1.6

Table 2 Summary of strips flagged as bad or noisy by various algorithms and after human inspection of the data. The last line shows the average values for all production modules.

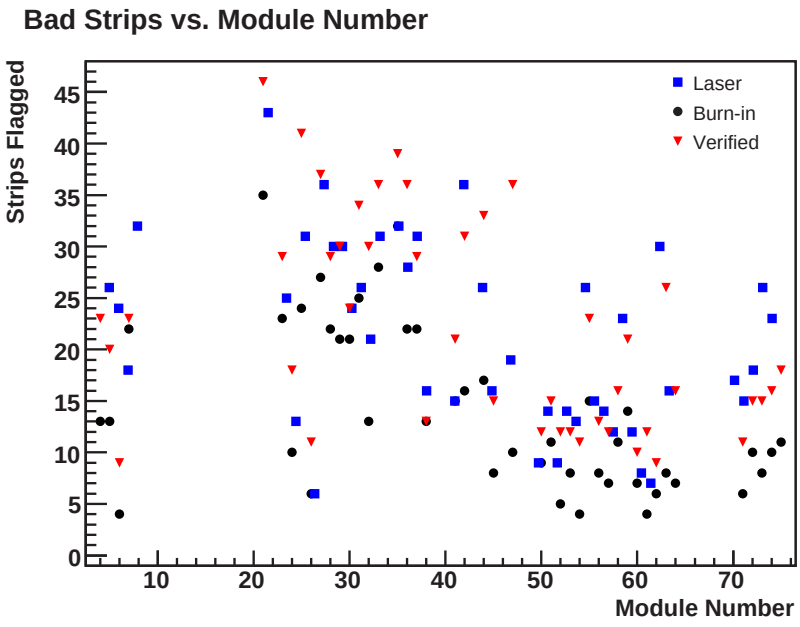

Figure 10 Bad strip flags vs. module number.

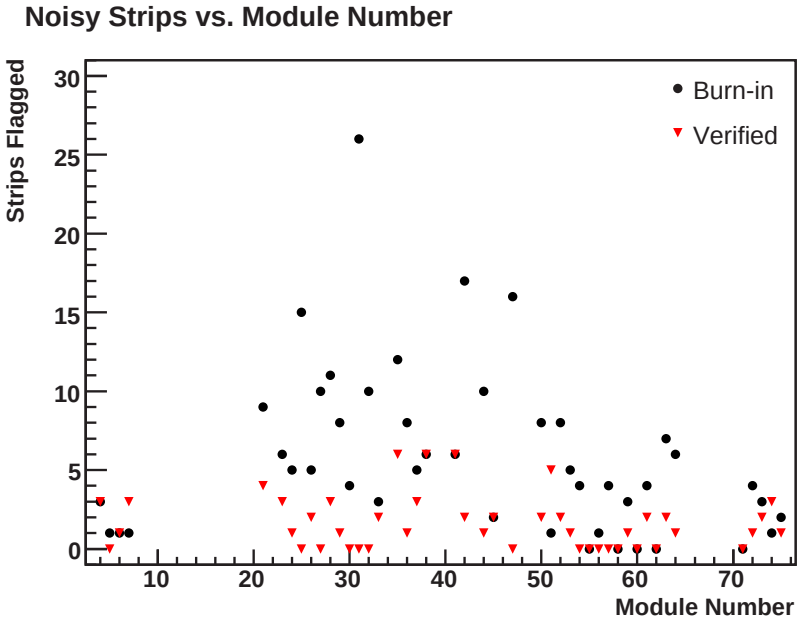

Figure 11 Noisy strip flags vs. module number.

B Correlations

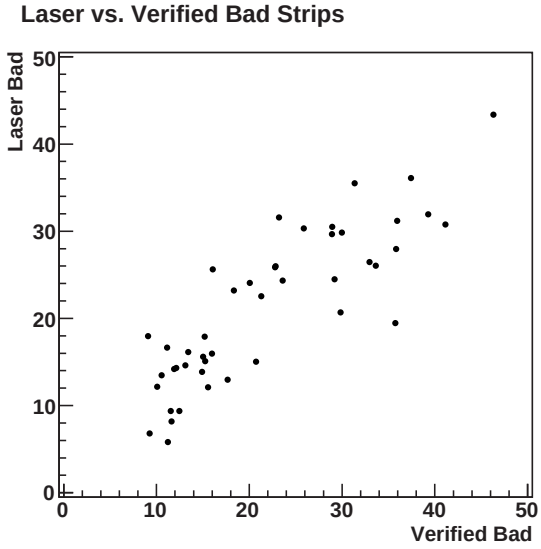

Figure 12 Correlation between laser test and verified bad strip flags.

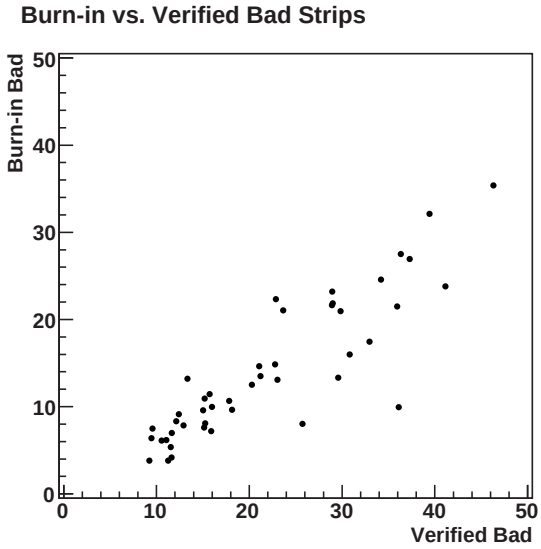

Figure 13 Correlation between burn-in and verified bad strip flags.

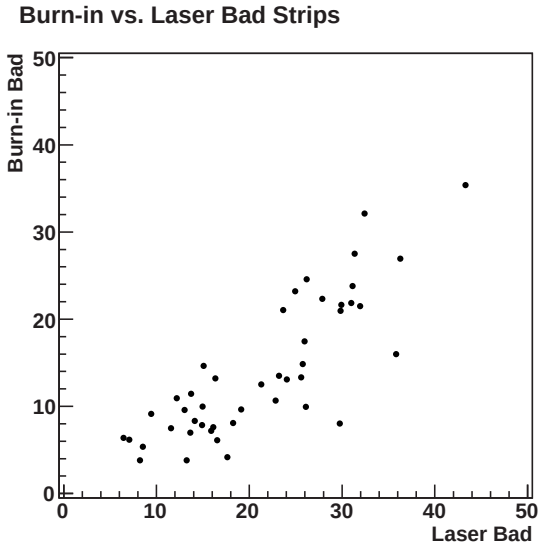

Figure 14 Correlation between burn-in and laser test bad strip flags.

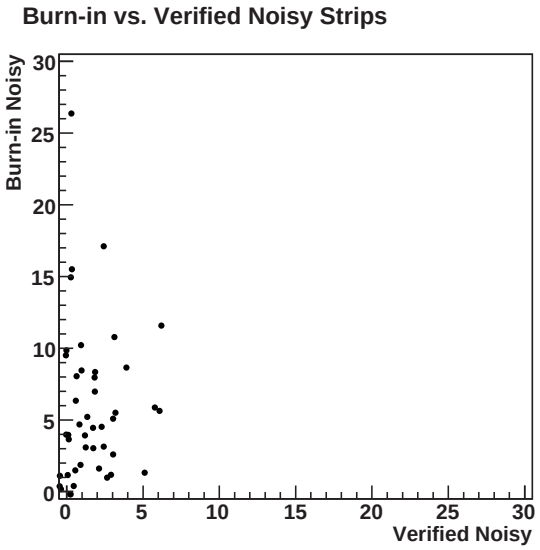

Figure 15 Correlation between burn-in and verified noisy strip flags.