A new DAQ for the ZEUS MVD telescope

Summerstudent Program 2008, DESY

Christian Rudolph[a]

[a] Technische Universität Dresden

16th September 2008

Abstract

This project covers modifications and improvement of the data acquisition software for the ZEUS MVD telescope at the DESY test beam. It shortly characterises the ZEUS test beam and explains basics of the telescope and hardware. The main part describes the data acquisition and data analysis of the telescope data.

Contents

1 Introduction

Test beam facilities are very important during the phases of research and development but also during the commissioning of a high energy physics detector. DESY offers at its site a 6 GeV electron test beam facility and necessary infrastructure, such as two beam telescopes. Beam telescopes are used to determine the particle tracks of the test beam very precisely. This information is then used to qualify detector modules, for example figures of merit such as efficiency and resolution are measured. One of the available test beam telescopes is the so-called ZEUS MVD telescope.

In the following sections the DESY test beam and the MVD telescope will be described. The main work of this summer was focused on the improvement of the data acquisition system of this telescope.

2 The DESY Test Beam

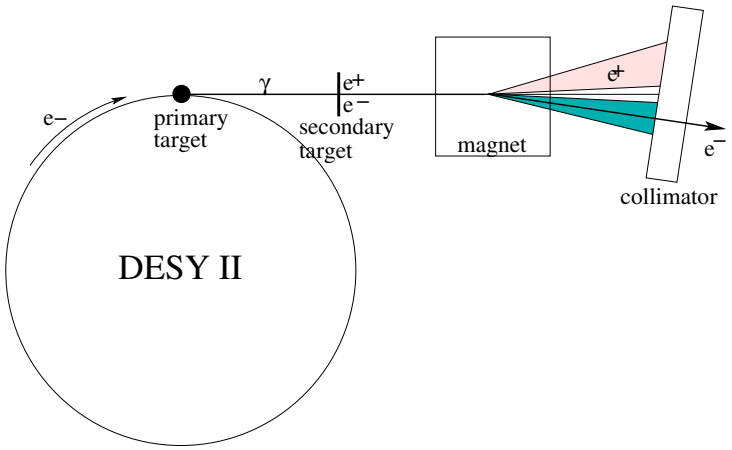

Figure 1: *Generation of electrons for the DESY test beam.*

The electrons/positrons for the test beam are provided as sketched in Figure 1: a Bremsstrahlung beam is generated by a 7 μm carbon fibre (primary target) in the circulating electron synchrotron beam DESY II, a metal plate (secondary target) is used to convert these Bremsstrahlung photons into electron/positron pairs, a dipole magnet spreads the beam out into a horizontal fan, and a set of collimators form the extracted beam. The magnet is used to control the energy of the beam. This parasitic test beam provides electron energies from 1 to 6 GeV/c. In this range the electrons are minimal ionising particles (MIPs). Therefore the physics of the test beam line is simple. The Bremsstrahlung spectrum has a 1/E dependence. The energy distribution of the electron/positron pair conversion is nearly flat. The accelerator control room handles the

fibre target and the beam intensity in DESY II. The test beam user contacts the control room if changes are necessary. The magnet settings for the selection of the momentum, the choice of the conversion target and the collimator settings are under control of the test beam user.

3 The ZEUS MVD Telescope

The ZEUS MVD telescope was was originally designed and constructed by the ZEUS collaboration to test the modules for the Micro Vertex Detector (MVD). Since then it was used by many different collaborations to study newly developed detector systems. It mainly consists of three detector reference planes, a trigger system, readout electronics, and software. The silicon detectors are located in an electric shielded metal box with very thin aluminium windows on beam line to minimize multiple scattering. On both ends of the telescope, plastic scintillators with photo multipliers are placed for triggering purposes, one at the rear and two at the front. All active components are mounted on an optical bench, which is situated on a translation stage controlled by the user from the testbeam hut.

3.1 The Telescope Reference Planes

The beam telescope consists of three reference detector modules (Ref.det 1, 2 and 3). Each of them provides two space coordinates (X and Y) of a track through the telescope. The Z coordinate is defined by the telescope geometry. The modules are a version of a CERN development. They consist of two 300 μm thick single-sided silicon sensors of 32mm$\times$32mm size with a strip pitch of 25μm and a readout pitch of 50μm; the strip directions of the two sensors in one module are perpendicular to each other. The 640 readout strips on each sensor are read out by the VA2 chips[1]. All modules have a very good signal-to-noise (S/N) ratio for MIP ($80 < S/N < 130$), and a high intrinsic position resolution of better than 15 μm.

3.2 The VME Crate

The main data acquisition electronics is positioned in a VME crate, which holds two V550A ADC modules, one V551B sequencer module, and a single-board computer (CETIA) holding a PowerPC processor providing the VME-bus and network access. A picture of the crate with its modules and wiring is shown in Figure 2. The V550A modules each provide two ADCs. Since there are only three telescope planes, one of them is unused.

[1]VA2 chip product of IDE AS, Norway

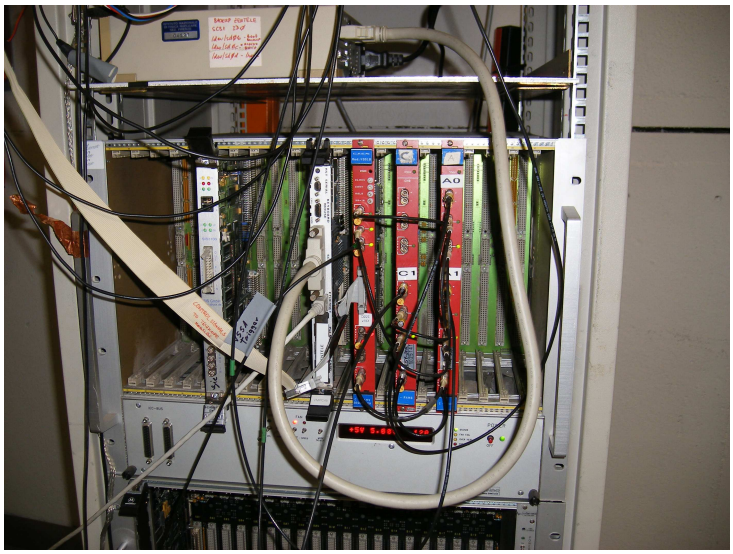

Figure 2: *The VME crate at the test beam area 22.*

3.2.1 The CAEN V550A ADC module

The V550A C-RAMS (CAEN Readout for Analog Multiplexed Signals) is a VME module which holds two independent ADC blocks. A photograph of this module is shown in Figure 3. Each of the blocks can handle positive, negative and differential input signals.

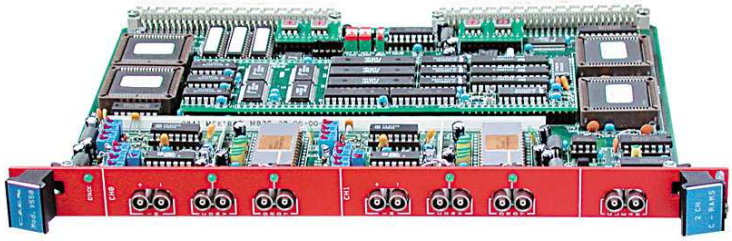

Figure 3: *The CAEN V550 ADC.*

The number of input channels must be specified before data taking. When an external CONVERT signal (from the V551 sequencer module) occurs, the input signal is sampled by the ADC into a 12 bit digital value. The module has an internal storage for pedestal and threshold values for every input channel. Thus, after the conversion, if the digital value is over threshold for the appropriate channel, the corresponding pedestal is substracted and the result is put into a first-in-first-out (FIFO) logic. Also, if data are in the FIFO, the module goes into DATA READY state signaling that data can be read via the VME bus. The V550A module holds several important registers which can be accessed by the VME bus, given the right address. The two most important are:

- the WORD COUNTER registers, which are read only registers holding 11 bit for each ADC block, containing the number of data in FIFO
- the FIFO registers, read-only registers of 32 bit, where the first 12 bit contain the channel data, the next 11 bit contain the channel number, followed by 7 unused bits and two control bits which are of no interest right now

3.2.2 The CAEN V551B Sequencer Module

Figure 4: *The CAEN V551 Sequencer*

The V551B C-RAMS sequencer (see Figure 4) is a module which handles the data acquisition of multiplexing chips (such as the VA2 chips used at the telescope). It controls the signals going to the V550 modules. Again the number of channels must be given to the module first. After receiving the multiplexing trigger signal, the module runs a programmable duty cycle in which it sends signals to both the front-end chips (VA2) for multiplexing purposes and the V550A modules via VME to make them start the analog-to-digital conversion. Time steps in this duty cycle can be set by the user; with the current settings one duty cycle takes about 5 μs. The module also generates a data ready signal when one of the V550A modules is in data ready state, this signal can be read out via VME.
A sketch how the modules in the VME crate are linked is shown in Figure 5

3.3 Trigger System

As mentioned above the trigger of the telescope consists of 3 scintillators. Their output signals are combined in an AND coincidence logic to form a trigger signal. This signal is then used by the V551B sequencer module to initiate the conversion cycle (see section 3.2.2). An automated trigger signal for the use without beam can also be created using a NIM gate generator.

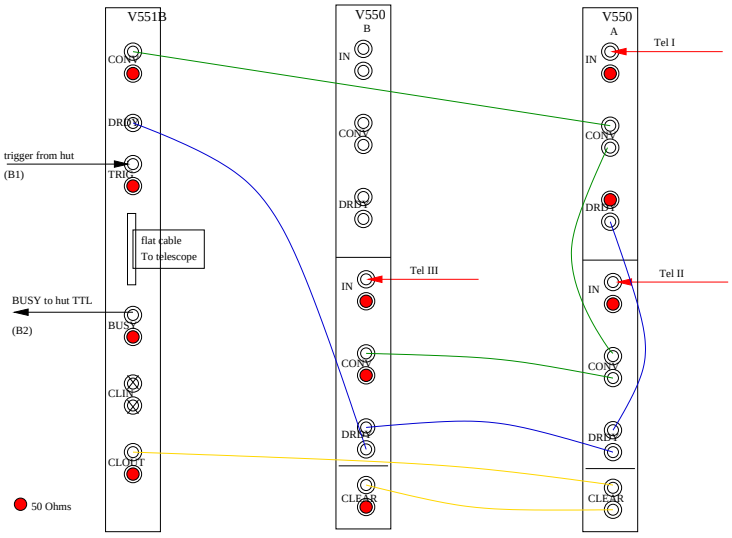

Figure 5: *The cabeling of the CAEN modules within the VME crate.*

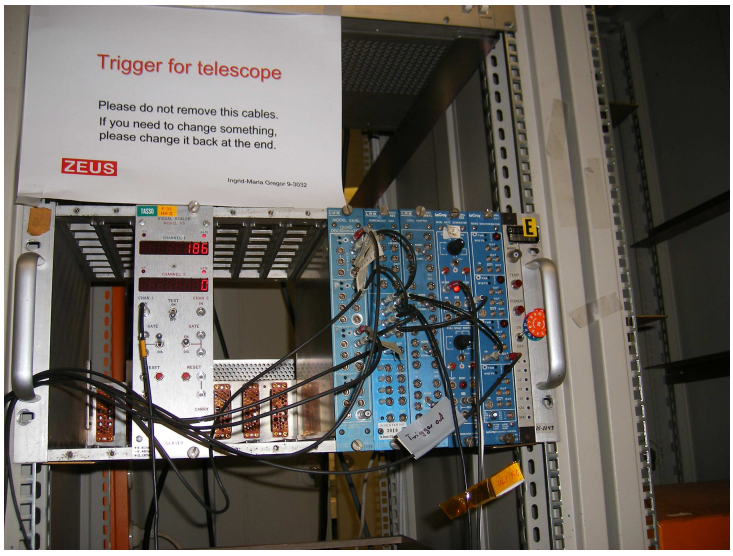

Figure 6: *The telescope trigger coincidence logic and gate generator*

4 Software

4.1 The Data Acquisition Software

During my work as a summer student I modified the existing telescope data acquisition software written in June 2008 by Lukasz Macewski, called *DAQ-test*. Different versions of the program are located on the zenpixell2 machine, directory */home/Lynx/C604b/Home/eudet/*. The executable can be found in different folders (see below) and is called daq.exe It is based on a program written by Sergey Fourletov called SI-tele. The program itself is divided into 4 sections, two of those with two subsections each, and three additional functions:

- a main routine
- the user menu
- run part, optional with data display
- calibration part, either with software or external trigger
- a function to reset all pedestals and thresholds in the V550A modules to zero
- two functions for interrupt signal handling

The main routine initializes the V551B and V550A modules with a specified number of channels (1280) using functions defined in an external VME library. It furthermore links the signal handler functions with the appropriate signals and then starts the user menu. The user menu shown in Figure 7 is where the user interacts with the program.

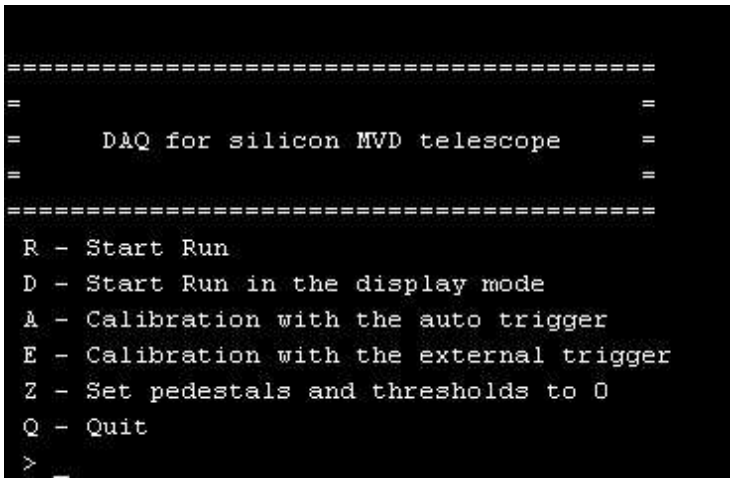

Figure 7: *The console view of the DAQ menu*

By selecting one of the shown letters (not case-sensitive), the user can choose whatever he likes to do. Before a new run can be started, though, one has to make a pedestal and noise calibration run.

4.1.1 The Calibration

In this part of the program it simply takes 1000 events and determines pedestal and noise values for every channel of the three telescope planes. In auto trigger mode, the program itself sends a signal to the V551 module causing it to start a conversion cycle, in external trigger mode the trigger signal is taken from the trigger input. The calibration should only be made with closed beam shutter, as no particle tracks should be taken into account. The procedure is quite simple. At first, the program waits for the DATA READY signal from the V551 module, then it looks into the WORD COUNTER register of each (active) ADC of each V550 module and then takes as much data out of each FIFO as written in the WORD COUNTER. As the taken data should simply be background, an easy calculation is done by the program to calculate the pedestal and threshold for each channel:

$$\mathrm{Ped_i} = \frac{1}{\mathrm{n}} \sum_{\mathrm{j=1}}^{\mathrm{n}} (\mathrm{ChannelData_{i,j}}) \tag{1}$$

$$\mathrm{Thr_i} = \mathrm{Ped_i} + 2 \times \sqrt{\frac{1}{\mathrm{n}} - 1} \times (\sum_{\mathrm{j=1}}^{\mathrm{n}} (\mathrm{ChannelData_{i,j}^2}) - \frac{1}{\mathrm{n}} (\sum_{\mathrm{j=1}}^{\mathrm{n}} (\mathrm{ChannelData_{i,j}}))^2) \tag{2}$$

Where i is the channel number and n is the number of entries in the channel. So the pedestal for a channel is simply the mean value of all entries in that channel, while the threshold is set to the mean value plus a 2-sigma deviation. These values are then written to the pedestal/threshold memory of the V550 modules for each channel. At the end of the calibration run, several simple ASCII histograms with the gained pedestal and threshold information for each telescope plane are printed on the screen.

4.1.2 The data taking run

The run mode is quite similar to the calibration run, except that the user is required to give some extra specifications on the run. At first, a run number must be entered. To store the run data, a file si_runX.dat in the /DATA/ subfolder is created, where X is the entered run number. If that file already exists, the user is asked if it should be overwritten. Next the user is asked for a number of events, after which the data taking should automatically stop. If zero is entered, the data taking runs infinite. Finally the user has the chance to enter a comment which will be stored in the data file aswell. Now

the program writes some header information to the file, containing start date and time of the run and some information about ADC status and V551 module conversion cycle time configuration. The run itself is analog to the calibration run, except that all data of one event is put into one single integer array called EVENT. At the beginning of each event,the event number, and the event time in seconds and microseconds is added to the EVENT array. Then, all different ADCs are asked for the amount of data in their FIFOs, and this data is read out and written to the EVENT. Note that both channel number and channel data are now stored in one integer number,and must be separated later in data analysis. After each ADC a separation pattern of 4 bytes(= memory size of one integer value)is added to the EVENT. This allows to separate the different telescope planes in one event later during data analysis. At the end of the event (when all FIFO data has been read out), an end pattern is added and the EVENT array is written to the data file. For the next event the EVENT array is reset afterwards.

4.1.3 The display data taking run

This was the main part of my work, and I had several approaches to online data display from different directions. They all have in common that the display run is basically a normal data taking run with all its specifications, plus an additional part where the online display is handled. The idea is to take every 10th event in the detector and display it as follows: For each of the three telescope planes two histograms should be drawn, one for the silicon strips in X-direction and one for the strips in Y direction (the Z axis is the beam axis). To achieve this I made three different approaches:

- writing a display file:
  In this program the display routine simply puts a display file containing the EVENT array to a separate display folder every 10 events. If this display file already exists, nothing is done at all by the program. The data display itself is made by another (ROOT-based) program called *disp* running on zenpixell2, which has ROOT libraries installed and access to the location of the display file. It is located in *zenpixell2:/home/Lynx/C604b/Home/eudet/ODisplay/*. This additional program reads the display file and uses the separators set by the DAQ to distinguish between the different telescope planes and X- and Y-orientated strips. It then fills an TTree object with the event data, which is then used to easily fill a set of 6 histograms and draw them. After the display file is fully read, it is deleted by the *disp* program, which then waits in a loop for another display file to be created by the DAQ. Although the principle should work fine, it did not during several test runs. The problem is not the creation and read-out of the display file, but the time it takes for the different machines to recognise that changes inside the display file directory

have been made. So the DAQ still sees an existing display file, therefore does not create a new one, while the online display program has deleted the file almost one minute before. Because of these file system problems, the attempt has not been followed further. The executable of the program can be found in the /DAQ-disp-1/ subdirectory on zentele:/nfs/C604b/Home/eudet/.

- using sockets to send the display data through the network:
  After the display file approach did not work as expected, my next intention was to use the socket(); command in c++ to create a server-client-relationship between the DAQ (server) and an online display program analog to the one described above. Unfortunately all attempts to add network capabilities to the DAQ have resulted the program collapsing, and I did not find the reason for that so far.However this option seems to be the most promising possibility to add effective online data display capabilities to the DAQ software. The executable of the program can be found in the /DAQ-disp-2/ subdirectory on zentele:/nfs/C604b/Home/eudet/

- using ASCII histograms:
  Finally, I remembered the built-in ASCII-histogram option of the DAQ mentioned in the calibration section. To create and fill appropriate ASCII histograms, several arrays besides the EVENT array described in section 4.1.2 were needed because of the given program structure. At first, three additional arrays are created, one for each telescope plane. Each of these arrays is filled with the event data when the corresponding ADC is read out by the DAQ program, while all data is still written to the EVENT array. Every 10th event, two additional arrays are created, one for X-strips and one for Y-strips. Then the 3 arrays for each plane, which still contain both information about channel number and channel data, are converted into the two arrays for X and Y. This is done by first separating the channel number from the channel data, and then determining if the channel number corresponds to a X or Y readout strip: the first 640 channels are linked to the X-strips, channels 640-1280 are connected to Y-strips. At the end two histograms are drawn, each virtually separated into three parts (one for each telescope plane), each covering 640 channels:

  This solution is rather slow and ineffective, as one might lose some events due to the time it takes to create the ASCII histograms. It is also not quite comfortable to see the histograms more or less flashing by in the console. But at the moment this version is the only one that works properly.

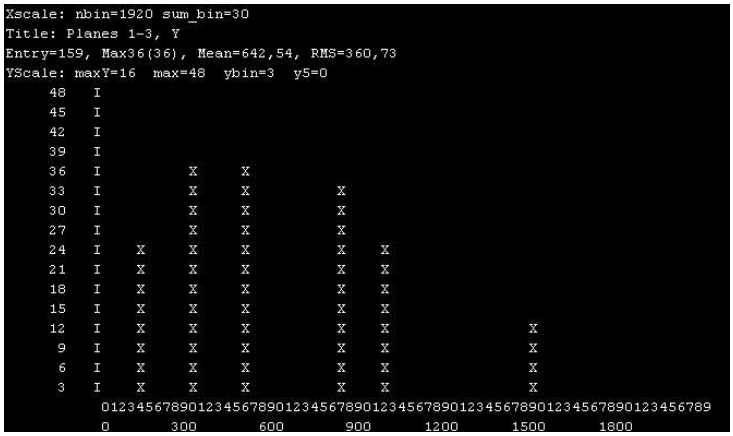

Figure 8: *Example for the ASCII histogram online display output*

4.2 The Data Analysis Software

To analyse data taken by the DAQ, an existing ROOT-based software called TELAna is available on *zentauruspc:/home/eudet/data1/TELE_zentele/TELAna*. To run it, however, one not only has to copy the data file from the DAQ /DATA/ directory to the TELAna /DATA/ directory, but also has to create a configuration file, in which the telescope characteristics such as number of planes, Z-distance of the planes, number of silicon strips for each direction (X and Y) for each plane and several other values have to be stored. Once this is done, one has to type ./tel X to run the analysis, where X is the run number. The program then automatically creates several of histograms. The program is very modular; to change its behavior, though, one has to edit the source code of the main program and recompile it. The main program calls the contructor of the class TELFile, in which the run data file is opened and all data is read in and filled to a TTree object, which in turn is written to a root-file. Besides the constructor, the TELFile class has 4 different procedures:

- SimplePed()
  This function is quite similar to the calibration run in the DAQ, since it simply calculates pedestal and noise values for a run without beam. It reads the TTree from the root-file created in the contructor and creates ROOT histograms for pedestal, noise and the ADC signals for each plane's X and Y orientated strips.

- PedNoi()
  In this function pedestal and noise for each channel are calculated and displayed aswell, but this time each single event is compared with all other events before and is rejected if it does not lay within 3 standard deviations from the mean value of all events before.

- FindClusters()
  This function finds detector hit clusters and performs an alignment of the detector planes, correcting possible angular misalignment of the telescope planes. It uses another class called TELTrack to reconstruct the particle tracks.

- HitNtuple()
  Here a ROOT ntuple with the track detector hits is built from the cluster and track information gathered in the HitNtuple() function and the TELTrack class functions.

The analysis software is very complex and powerful. Unfortunately, it is also poorly commented, so it is very difficult to extract information how it actually works.

5 Results

Since the DESY test beam was only available right at the end of the summer student program 2008, only one real data run with the new data acquisition software could be done. I started a simple data taking run with pedestals and thresholds in the V550A ADC modules set to zero. The intention was to determine pedestals and noise of the readout channels with the TELAna analysis software. As I did so, it turned out that the results looked not as we expected (see figures 9 and 10).
Instead of being rather flat and commonly distributed, the pedestals and noise are very chaotic. In addition, they are also pretty high. The first intention was that something is wrong with the wiring, as this had been modified and improved since the last data run. However, no error could be found there.
A comparison of the analysis of binary run data files acquired with the older SI-tele software (see section 4.1) with the analysis of latest binary run data showed that the error is obviously software-generated. Somehow the output binary data files of the new data acquisition program are corrupted. Since this comparison had never been done before in my summer student program, the error remained undiscovered till the end.

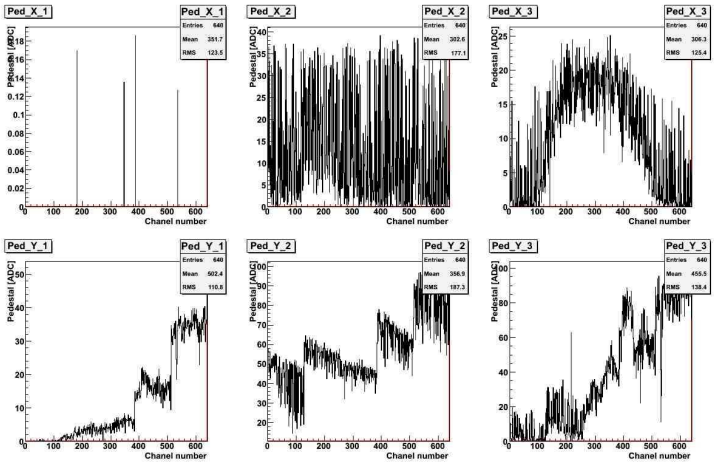

Figure 9: *reconstructed pedestals of the test beam run on 11th September 2008*

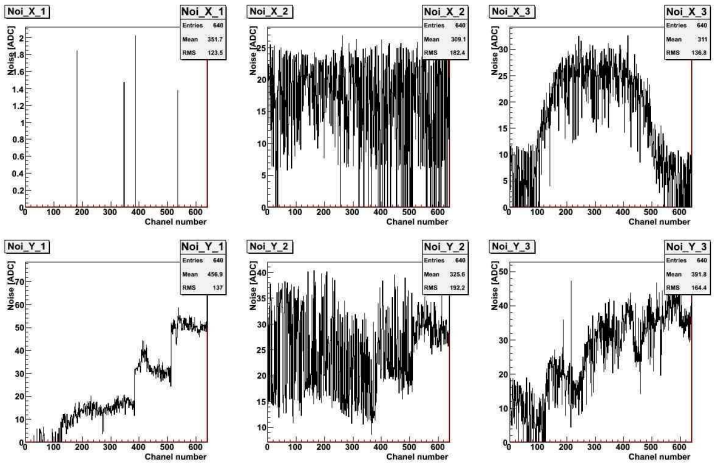

Figure 10: *noise distributions of the test beam run on 11th September 2008*

6 Conclusion

During my work I proved that online data display with the current DAQ is possible. I outlined different possibilities to implement the online data display. In conclusion one can say that the server-based data display is probably the best sollution, even though it is not working yet.
Thus, the next steps are very clear:

- One has to find and fix the error in the new DAQ that creates corrupted binaries
- One has to find and fix the bug in the server-client-based data acquisition software. I assume that this approach is most the most promising attempt to build in effective online data display in the current DAQ
- With proper network capabilities added to the DAQ, one might think of enhancing the data acquisition to become completely server-based. In this case the user could easily program his own client to receive the event data, and even control the DAQ program remotely.

7 Acknowledgements

I want to thank Ingrid-Maria Gregor for overseeing my project work, giving me hints where to find essential information, explaining the test beam and telescope setup and answering my questions. Thanks to Jolanta Sztuk who provided lots of information about the data acquisition and data analysis software and helping me whenever I had problems with programming. Furthermore I want to thank Ulrich Kötz who explained several hardware components and their functions. My thanks also go to Krzysztof Wrona, giving programming solutions and hints whenever I got stuck somewhere.

References

[1] Characterisation and Monte-Carlo study of the T22 Electron Test Beam Line at DESY II, *H.de la Torre Perez et al.*, EUDET-Memo-2007-49

[2] MOD V550 Technical Information Manual, Revision n. 3, *CAEN S.p.A*, 2002

[3] MOD V551 Technical Information Manual, Version 1.2, *CAEN S.p.A*, 2003