The LHCb TT production database and an overview of quality characteristics of the detector modules

Nicola Chiapolini

19.03.2007

Abstract

The LHCb TT detector station consists of four planar layers with modules built of seven silicon sensors with readout electronics at one end. The 148 modules of this detector are inspected in a test stand and the results of this inspection are stored in a production database.

In this paper I will give an overview on the database structure and the web interfaces used to store the data. In addition, a first analysis of the quality characteristics of silicon sensors and detector modules is given.

Contents

1 Introduction

1.1 Motivation

The Physics Institute of the University of Zurich designed and builds a silicon strip detector with a sensitive area of about 8 m^2 [1]. The detector will be installed as a tracking detector in the magnetic spectrometer of the LHCb experiment at CERN. The detector station consists of four planar layers with modules built of seven silicon sensors with readout electronics at one end [2]. The 148 modules of this detector are inspected in a test stand and the results of this inspection are stored in a production database.

The assignment was to complete the database and implement overviews for the quality characteristics of silicon sensors and detector modules. Based on this overviews the modules are then matched for the detector assembling.

1.2 Subtasks

The actual work consisted of different subtasks in succeeding phases. In a first phase the data form the burnin-tests had to be uploaded into the database tables. In the second phase the implementation of web interfaces was necessary to retrieve the uploaded data. The third phase consisted of the implementation of the grading interface and the grading itself.

The following sections will be used for documentation of the database software, explaining also some technical details.

2 Detector and Tests

As mentioned in the introduction, the detector consists of 4 planes with silicon sensor modules[1] including readout electronics (see Figure 1(a)). A modules contains seven sensors with 512 readout strips. These sensors are connected either into 2 or 3 readout sections named L, M and K. The L section has always 4 sensors while the M section combines 3 sensors in case of only two sections and 2 sensors otherwise. If a K section exists, it contains one sensor. Each readout section has its own hybrid containing the front end readout electronics. On each hybrid 4 readout chips called Beetles are placed. Each Beetle collects the signals of 128 strips. Figure 1(b) shows the configuration of a module.

The remaining part of this section gives some insight into the different quality assurance tests conducted during the module production and the data that is stored in the database. Before the module assembly each individual component was inspected visually for scratches or other damages. In addition the most relevant electronic properties of the different components were analysed. For each sensor, for example, the leakage current was measured as a function of the applied voltage and the voltage needed to achieve full depletion was determined. Data from these tests were stored in the production database.

A second part of the database stores data collected during the assembly process of the modules. Table 1 shows the steps of this process. For each step at least a timestamp and a comment are stored. Column 3 gives the type of data, if more data is collected.

Table 1: The assembly process for halfmodules

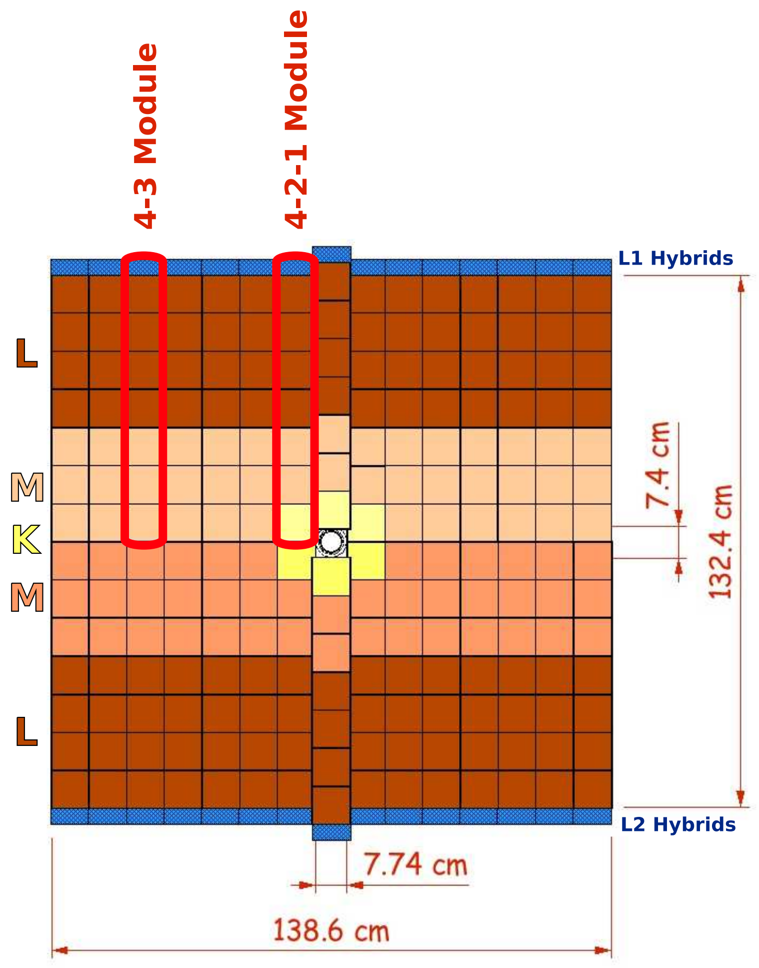
Step	**Task**	**data**
1a.1	align sensors & glue rails/copper blocks	
1b.1	Visual Inspection	pictures
1b.2	Metrology I	shift & tilt
1b.2	Metrology II	flatness
1c.1	Flip, glue HV-cable	
1c.2	Bond sensors	
1c.3	Inspect bonds	pictures
1c.4	Glue Kevlar caps	pictures
1c.5	Stage I burnin	
2.1	Mount upper hybrid	
2.2	Provide GND + HV	
2.3	Bond	
2.4	Inspect bonds	
2.5	Glue Kevlar caps	
if module contains 3 sections the second group of steps is repeated.		

After the completion of the assembly process, the modules had to undergo

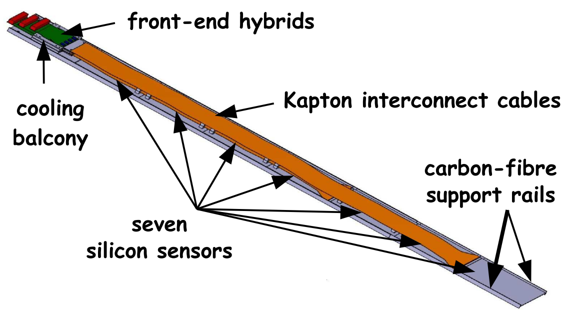
[1]Originally the term "halfmodule" was used to refer to modules as two of these will be combined into full modules before installation. Due to the fact, that the assembly and testing worked only with halfmodules, these now are referred to simply as modules.

[2]of each sensor

[3]of each sensor and overall for the module



(a) One detector plains



(b) One module

Figure 1: Detector Layout

exhaustive so-called burnin tests. With these the functionality and the electronic properties of the modules were measured. The stability of the current as a function of time and as a function of temperature was analysed. During stability measurements the temperature in the test box was cycled multiple times between room temperature and about 5°C. During these cycles all modules were biased at 500V and leakage currents where monitored continuously. Data on the strip noise of the modules was collected at both warm and cold temperatures. In the last cycle the signal pulse shape and the charge collection efficiency as a function of the applied voltage were analysed at high and low temperatures. For charge collection and pulse shape measurements signals were created in the silicon sensors by a sharp 4ns long laser pulse. In addition tests where conducted with the test pulse implemented on the Beetle chip instead of the laser created signals. For the analysis of these tests, empirical functions were fitted to the data.

To the pulse shape data the function $f(t) = A \cdot (\frac{(t-t_0)^2}{2 \cdot \tau^2} - \frac{(t-t_0)^3}{6 \cdot \tau^3}) \cdot e^{-\frac{t-t_0}{\tau}}$ was fitted [5]. The so called "signal remainder" was then defined as the value of this function 25ns after the maximum of the fitted pulse. For the charge collection test a function derived from a charge collection model [3], [4] was fitted to the data. The operation voltage of the sensor is then given by the voltage where this function saturates.

Table 2 shows the names used for the burnin tests in the database.

Table 2: The burnin tests

DB Name	**Measurement**
IV	leakage current as a function of voltage (per hybrid)
Itemp	current as a function of temperature (per hybrid)
IT	current as a function of time (per hybrid)
noise	noise (per channel)
charge collection	signal amplitude as a function of bias voltage (two channels per hybrid)
pulsshape	signal amplitude as a function of sampling time (per channel)

3 Documentation of the Database Software

The database and all used interfaces have grown over time so the most natural way to explain the database software will be to follow this evolution. The web interface documented here is located at

http://lhcb.physik.unizh.ch/tt/database/index.php.

For the administration of the database *phpPgAdmin* [7] was used. The installation is located at

http://lhcb.physik.unizh.ch/admin/phppgadmin/

but is password protected. Appendix A contains screenshots of many of the pages described in the following.

Before starting the documentation of the database, the notation used has to be explained. All names of elements in the database or links on web pages will be printed in *italics* and a field of a table is sometimes referred to in SQL syntax as *table_name.field_name*. To refer to groups of tables the wild card * is used. *sensors** therefore refers to any table with a name beginning on *sensors*.

3.1 General Information

3.1.1 Schemas

The production database is divided into two parts called schemas. Each schema contains a set of tables. Most of the data is stored in the schema *public* but for special tables related to the module assembly the schema *module_data* was used. More details on this will follow later. Besides tables each schema contains so-called views that allow easy access to data stored in different tables.

3.1.2 Naming Conventions

To keep the database easy to understand some naming conventions were used. Unfortunately the convention was not followed strictly since several people were involved in the development at different times. During this documentation the scheme will be explained and the exceptions will be pointed out.

Table names The table names generally start with the name of the component the data stored in the table belongs to. An example is *sensors_automatic_tests*. There are of course tables that do not belong to a specific component. In these cases we tried to choose an informative name like *files* or *operators*. The exception here are the three construction tables which belong to the modules. We will come back to these in Section 3.3.

Auxiliary Tables Another big class of tables start with an underscore. These are auxiliary tables that contain the possible values a field in another table can assume. A good example is the table *_hybrids_type* that contains the values L, L2, M and K.

Serialnumber and ID All components we received in Zurich carried a serial number for identification. In many cases these are not unique. For example there exist three hybrids with serial number 101: one type L, one type M and one type K. We therefore decided to use an internal ID to uniquely identify each component in the database. For example the table *sensors* contains a field *serialnumber* with the serial number printed on the device and used for identification during testing and module assembly as well as a field *sensorID* storing the internal ID. In the *modules* table and some of the tables for module tests the internal ID is called Nr. The field *moduleNr* contains the internal ID and not the serial number of the module. Adding to the confusion is the fact that in the web interface the module serial number is sometimes referred to as ModuleNr too.

3.1.3 Special Tables

As mentioned above there exist some tables that do not belong to any specific component. Some of them are auxiliary tables and the mapping of those to the fields that reference them can be found in Table 3. For the other special tables further explanations will be given in the following.

Table 3: Mapping of auxiliary tables to fields

_badchannels	modules_bad_channels.defect_type
_badstripdefects	sensors_automatic_tests_badstrips.defect_type sensors_badstrip_test_defect_infos.defect_type sensors_vendor_badstrip_defect_infos.defect_type
_burninbox	modules_burnin_tests.box
_burninboxposition	modules2burnin_participation.position_in_box
_cables_type	cables.cabletype
_cfsupport_type	cfsupport.cfsupport_type
_hvcabletype	modules2burnin_participation.hv_cable
_hybrids_burnin_cabletype	hybrids_burnin_tt_tests.cable
_hybrids_type	hybrids.hybridtype
_longtermtype	sensors_longterm_tests.testtype
_mimetypes	files.mimetype
_modules_jig	modules.jig
_modules_stop	modules.stop_nr
_modules_type	construction_stages.moduletype modules.moduletype
_pitchadapterdefects	pitchadapters_badstrips.defecttype
_pitchadaptertype	pitchadapters.pa_type
_pitchadaptervendors	pitchadapters_vendor_infos.vendorname
_sensortype	sensors.sensortype

operators As the name suggests the table *operators* contains the data for the persons that can enter or edit content in the database. Besides the first and the last name a *loginname* is stored in *operators*. The *loginname* is the internal ID for this operator. The status of an operator can be controlled via an additional field, named *active*. If this field is set to true, the operator will appear in the operator selections of the web interface. If the value is false the operator will be hidden. No operator can be deleted from the table as long as he is referenced by any entry in the database.

files Files are not stored within the database. Instead they are copied into a folder structure and only the file name and some additional information is stored in the *files* table of the database. The files are usually related to entries in other tables of the database. We therefore need to link these tables with the entries in *files*. For the linking we use special tables with names ending on *2files*. Using a special table for linking allows to store multiple files for one entry. As an example *sensors_vi_tests* stores the results from the visual inspection of the sensors. During this test pictures of damages where taken. When uploading the test results to the database, the pictures are copied into the folder structure and for each an entry in *files* is created. The test results in *sensors_vi_tests* are then linked to these entries through *sensors_vi_tests2files*.

paths The *paths* table contains the full path names for the directories in which the files are stored. The field *purpose* contains a keyword describing the type of files saved at this location. This allows to easily adjust the storage directory of files. As *files.fullpath* references *paths.fullpath*, all entries in *files* are automatically updated in case of a modification in *paths*. The web interfaces refer to the purpose field and do not need any update if the storage directory is modified.

shipping A dedicated part of the database was implemented to keep track of shipments leaving and coming to the University of Zurich. It consists of five tables. The relations between these tables are displayed in Figure 2. *shippings* identifies the different shippings and basically contains only a counter numbering the shippings. *shipping_location* stores the names of the institutes Zurich collaborates with. *shipping_outgoing* and *shipping_incoming* contain information about the sender and the recipient and allow them to add comments about the shipment. Each entry in *shipping_part_infos* links a component to a shipment. This linking is done by saving an identifying expression like *sensors.serialnumber=40315*. The part before the dot identifies the table the components information is stored in. The part between dot and equal sign defines a field of this table that can serve as a key. After the equal sign follows the value of this field for the selected component.

3.2 Sensors & Hybrids

After this first general block the rest of the database structure will be explained now. The development of the ST production database software started with the silicon sensor information since the silicon sensors were the first component the group in Zurich produced data for. All sensors had to undergo extensive testing. The resulting data was uploaded using the first parts of our web interface and was stored in the *sensors** tables. For each test there exists at least one table containing the main test data. These tables are named *sensors_⟨test name⟩_tests*. Some of the tests have additional tables storing more detailed data. The names of these additional tables extend the name of the test's main

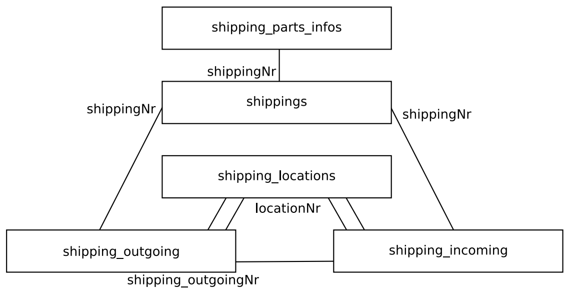


Figure 2: The Structure of the Shipping Tables

table (e.g. *sensors_iv_tests_points* for the data points from the IV curve messured on the sensor.). In addition to our own test data we wanted to store vendor information. The tables *sensors_vendor_** and *sensors_batch_** contain this data.

The web interface offers several forms that allow to upload data for sensors and to display the entries stored in the database. The *upload interface*[2] consists of one or more web pages for each test. The tests are all finished by now so no details are given here. The stored test results can be displayed using the link *display summary*[3]. A filter allows to set criteria for the sensors that will be displayed. On the summary page a link on the sensor's serial number leads to a page with detailed information for this sensor. The details page shows all test results stored for this sensor and allows to plot test curves and access stored pictures. In addition, the sensors can be graded based on the test information. Distributions of the most important test results can be displayed in histograms using the link *display histograms*[4]. For this the interface allows to define a sensor selection too.

Next, hybrids were added to the database and its interface. Originally the tables for hybrids were called *pitchadapters_**. The database still contains these tables, however all the important data stored in these tables has been migrated to the new *hybrids_** tables. The naming of these tables is analogous to the one described for the sensors. The group of tables called *hybrids_burnin_**. does not belong to the hybrids testing and will be described in Section 3.4.

The web interface offers three forms to the hybrid part of the database. First there is an interface to *add new hybrids*[5] to the database and store a comment. Second the results form the only test for hybrids, the visual inspection, can be stored with the *VI upload for hybrid*[6]. In case the hybrid entered during VI upload does not exist yet, a new database entry will be created. The third interface allows to display a summary similar to the one for the sensors.

[2]Figures 12 & 13 in Appendix
[3]Figures 14 & 15 in Appendix
[4]Figures 16 & 17 in Appendix
[5]Figure 18 in Appendix
[6]Figure 19 in Appendix

3.3 Module Assembly

The next step was to realise the module assembly. Figure 3 shows the database structure described in this section. First there is a table called *modules* that contains general information on the module like its serial number or its grade. The other tables with names *modules_** are not related to the module assembly and will be explained in Section 3.4.

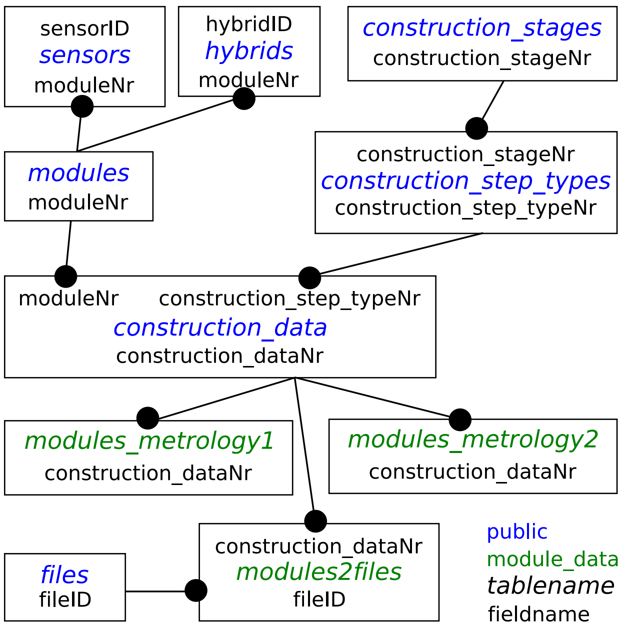


Figure 3: The Structure of the TT module assembly.

To store the data from the assembly process a database structure was chosen that can easily be adjusted in case of changes in the assembly process. Instead of creating a specific table for each construction step, only two tables are used. Table *construction_step_types* defines the individual steps. To fix the order of the steps a linked list is used with the next step stored in *construction_step_types.next_step*.

The table *construction_data* stores the data from the construction steps. An entry in *construction_data* is then linked to the according entry in *construction_step_types*, so that the step this entry belongs to can be identified. This solution was possible as most of the assembly steps do produce the same type of data, namely a date and a comment (see Section 2. If a production step yields additional data this is stored in an extra table defined in the field *construction_step_types.data_table*. This field contains the name of the table the data should be stored in. Unlike the tables mentioned so far, the data tables do not belong to the schema *public* but to the schema *module_data*. This division facilitated the coordination between the different programmers writing the database software.

As the module assembly is organised in stages, the steps in the database are grouped into stages too. Therefore a third table exists called *construction_stages*.

It contains one entry for each stage and is linked to by the construction steps in *construction_step_types.construction_stageNr.*

The web interface for the module assembly has two starting points. The first starting point is the creation of a new module entry[7] in the database (a new entry in the *modules* table and the corresponding links in tables *sensors* and *hybrids*). After this, one can upload the data from the assembly process into *construction_data*. The interface for this upload groups the steps of one production stage together and allows to navigate through the different stages. The second starting point is to continue the assembly of an existing module[8] and to enter the interface directly at the last stored stage of the assembly process. The data stored for a module can be displayed by clicking on the *Info* link on the *Module Selection* page.

In addition to the data from TT module production explained above, the *construction_** tables were also used to store information from the module assembly for the Inner Tracker detector built in Lausanne. For this data a web interface similar to the TT one was implemented but as the need arose for a more direct access to the data from Lausanne, the database was cloned. The IT data is therefore stored in its own database now.

3.4 Module Testing

3.4.1 Database

During and after the module assembly so called "burnin tests" take place. The results of these tests are stored in the tables *beetle_**, *hybrids_burnin_**, *modules_burnin_** and *modules2burnin_participation*. The table *beetles* identifies the Beetle chips on each hybrid. This is necessary as some burnin tests produce data per Beetle or even per readout channel.

As Figure 4 shows, the structure of this part of the database is quite complex. A module burnin test contains up to four modules that are placed in the test box. These modules undergo the different tests that produce the data stored in the database. Such a series of tests performed for a given set of modules is referred to as fill. As the number of modules in one fill can also be smaller than four we use the *modules2burnin_participation* table to link the modules with the fill number used to identify the test. Additionally, we store in this table the position of the module in the box and the identifier of the readout cable used. The *module_burnin_tests* table is referred to by *hybrids_burnin_tests* and this table in turn is then referred to by the tables containing the testresults. This cascade of tables allows a high flexibility with respect to the numbers of tested modules per fill and hybrids per module as well as the number of repetitions of a particular test.

3.4.2 Upload

There exists a web interface for uploading the results of the burnin tests but as these tests produce large amounts of data we normally use a combination of bash- and python-scripts to upload this data. Therefore, the web interface will

[7]Figure 20 in Appendix
[8]Figure 21 in Appendix

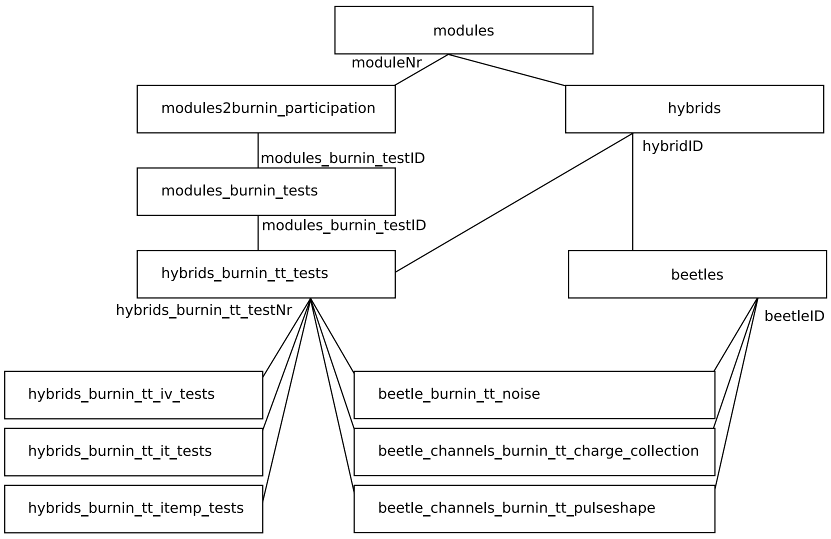

modules
moduleNr
modules2burnin_participation
hybrids
hybridID
modules_burnin_testID
modules_burnin_tests
modules_burnin_testID
hybrids_burnin_tt_tests
beetles
hybrids_burnin_tt_testNr
beetleID
hybrids_burnin_tt_iv_tests
beetle_burnin_tt_noise
hybrids_burnin_tt_it_tests
beetle_channels_burnin_tt_charge_collection
hybrids_burnin_tt_itemp_tests
beetle_channels_burnin_tt_pulseshape

Figure 4: The Structure of the Main Burnin Tables

not be presented here and two of the upload scripts will be explained instead. All upload scripts can be found at

lhcb.physik.unizh.ch:/home/hep/lhcbweb/BurnInResults/scripts

First, the upload script for charge collection data will be explained. The script uploads data from one fill at a time. The data files are expected to be stored in the folder structure under */disk/groups/hep/lhcb/burn/*. Before uploading, the data needs to be analysed as described in Section 2. This analysis is done by the C-Program *DepletionFit_BurnIn* in */home/hep/lhcb/HVscan*. The output of this program then needs to be checked manually and results with bad fits removed. The upload script can then be called by the command *./StoreCCCData FILL* where *FILL* is the number of the fill for which the data shall be uploaded. The bash script will select all charge collection files for this fill in the folder structure and run the python script *UploadCC.py* on each of these files. The python script handles the database interaction while the bash script takes care of the file handling.

Uploading noise data is mostly the same but instead of a fill number the bash script takes a file name as parameter. This file should contain a list of all data files prepared for upload. The command is *./StoreNoiseData FILE*. This script calls a python script too (*UploadNoise.py*). The amount of noise data is too large to calculate the average noise value of all 128 channels of each beetle chip dynamically when this is needed in summaries. We therefore store this value in the database in addition to the individual noise values for each of the 128 channels. The calculation is done by the script *CalcNoiseAverage*. This script needs to be called after each noise upload.

3.5 Test Analysis and Grading

3.5.1 Displaying Burnin Data

The web interface *Summary of the Fills*[9] offers direct access to the data from each fill. On the first page all fills with the corresponding modules are listed. Numbers in parentheses give the production stage the module was in when it was tested. When a module is selected from this page, all test results for this module are displayed. From here it is possible to display the graphs for most of the measured curves. For example clicking on an average noise value leads to a more detailed page for the corresponding Beetle chip. On this detailed page, the noise values from different tests are displayed. It is also possible to display the noise per channel or to generate a histogram of the channel noise values.

A useful tool is the *Burnin Histograms*[10] interface. This tool allows to create histograms of various variables based on user adjustable settings. The first group of settings influences the histogramming. The second group defines the way the data is selected form the database. The histogram mean, root mean square, number of entries, number of over- and underflows are calculated and displayed. In addition to the plot of the histogram the tool identifies the different entries in each bin of the histogram. This is especially useful to determine and identify outliers. Finally the histogram tool allows to download the complete binned or unbinned data set as a text file for further processing for example in matlab or root.

3.5.2 Grading

The last part of the web interface is the *Grading Summary.* The grading was meant to facilitate the selection of modules for the full-module construction. The functionality of the page is similar to that of the sensor summary. A filter allows to select modules by properties like type or grade and then a listing gives an overview of relevant data for the selected modules. Special in this interface is the ability to specify whether the general comments on the module or the average operation voltage of each section should be displayed. The reason for this is the large amount of time that it takes to calculate the operation voltage for a large number of modules.

From the overview page the link *details* leads to a grading page for the selected module. Here all module characteristics are summarised. The displayed values are mostly averages over all measurements that were produced for the set of parameters displayed in the interface (i.e. noise values are averaged over all measurements for the given stage, hybrid type and temperature). In addition, high and low values are marked to help with the grading process. The conditions used for highlighting are listed in Table 4. The values are based on the overall distribution of the measurements for all modules and set such that they end at about 2σ from the mean. At the bottom of the grading page there is a button bringing up a new window that allows to set the grade of the module. Modules with values within the limits are graded A, those with values in the red or blue ranges are graded B and modules with even worse results get C.

Approximately half of the modules were graded B or C for a wide variety of reasons. Because of this result, this grading was finally not used for matching

[9] Figures 22, 23 & 24 in Appendix
[10] Figures 25 & 26 in Appendix

the modules and the interface was abandoned.

Table 4: Conditions for Highlighting

Metrology Shift	red	> 25
	orange	> 50
Metrology Flatness	no limits	
Bad Channels	red	> 2
Noise	blue	< 2
	red	> 3
Remainder	blue	< 10
	red	> 30
Operation Voltage	blue	< 170
	red	> 270
I @ 500V	red	> 1
	orange	> 10

4 Short Overview of Module Properties

The histograming tool (see Section 3.5.1) allows to easily get an overview of the main module properties. The following sections will give some insight into these properties.

4.1 Operation Voltage

As mentioned in Section 2 the operation voltage of each readout section is determined by fitting a function to the data from the charge collection measurement. First an algorithm is applied to the data in order to substract the common mode noise. Afterwards two of the measured channels are selected and the function is fitted to the corresponding data points. The two results of these fits are then stored in the database and the operation voltage of the module in this measurement is defined as the average of these two values.

Due to poor results in the first analysis, the operation voltage is not used as a matching criteria. First the correlation of the modules operation voltage to the full depletion voltage of the used sensors is not very strong. As an example Figure 5 shows the correlation of operation voltage for all L readout sectors to the average full depletion voltage of the four sensors in this readout sector. The correlation coefficient is 0.30.

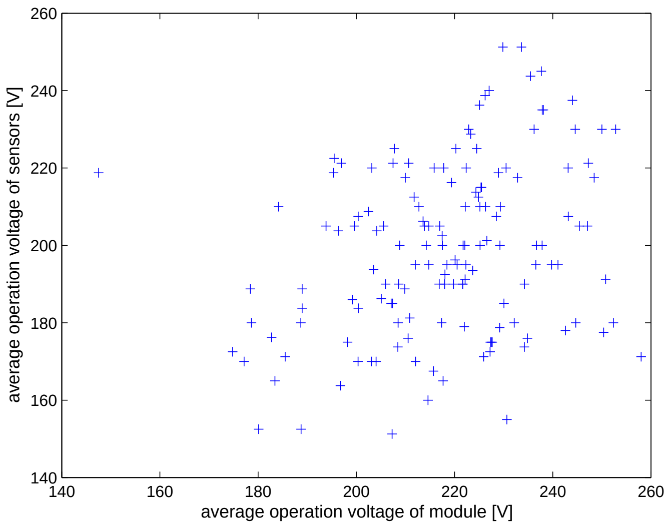
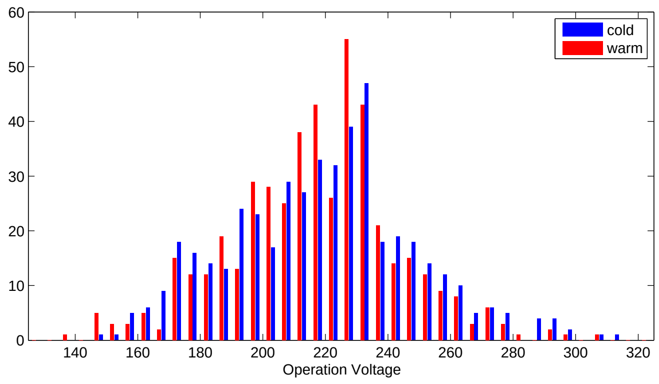
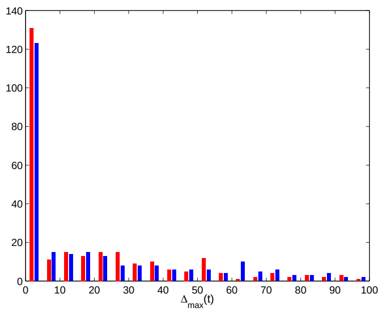


Figure 5: Correlation of Operation Voltage to Average Full Depletion Voltage

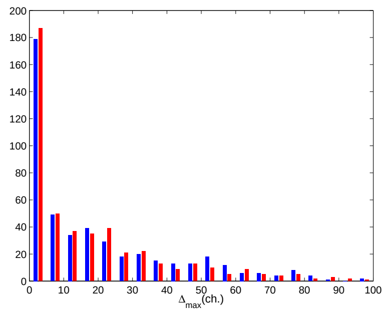
Second and more important the error on the calculated operation voltages is too large. Figure 6 shows three histograms analysing the operation voltage. Figure 6(a) shows the measured distribution of operation voltages. One entry in the histogram corresponds to one measurement as described above. The figure distinguishes tests conducted at high and low temperatures. Figure 6(b) shows the distribution of the maximal differences of the operation voltage found for a given readout sector on a given module, over different burnin runs. Many values are close to zero but the tail to larger values indicates a serious problem with the analysis. Figure 6(c) confirms this result. This histogram shows the difference between the operation voltages calculated for the two channels in each measurement for each readout sector. Here the tail indicates a problem, too.



(a) Distribution



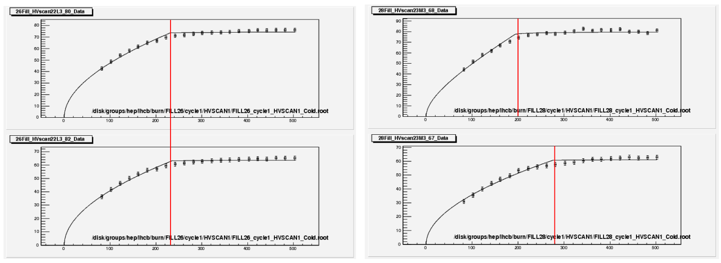
(b) Reproducibility



(c) Variation

Figure 6: Analysis of Operation Voltage

The main reason for this problem is expected to be in the common mode subtraction. The laser used to create the signal in the silicon sensors illuminates several readout strips. Its focussing is not precisely reproducible from fill to fill and sometimes the number of illuminated strips is large. This can lead to problems in the common mode subtraction algorithm which relies on the assumption that only a small number of strips actually see a signal. In addition, the fitting of the function used for the analysis might cause some further problems. The function was originally derived for heavily irradiated sensors and not for new sensors like the ones we test. Figure 7 shows a good and a bad result from a charge collection analysis. Some further studies are required to fully understand and solve these problems. Due to lack of time it was decided not to use these operation voltages in the module matching and to rely on the sensor depletion voltages instead.



(a) Good Result (b) Bad Result

Figure 7: Analysis of Operation Voltage

4.2 Noise

A simple analysis of the noise gives two results. Figure 8 shows that the noise is temperature dependent. Tests conducted at high temperatures have a lower average noise (given in ADC counts) than measurements at low temperature. The reason for this is that the gain of the Beetles decreases with increasing temperature.

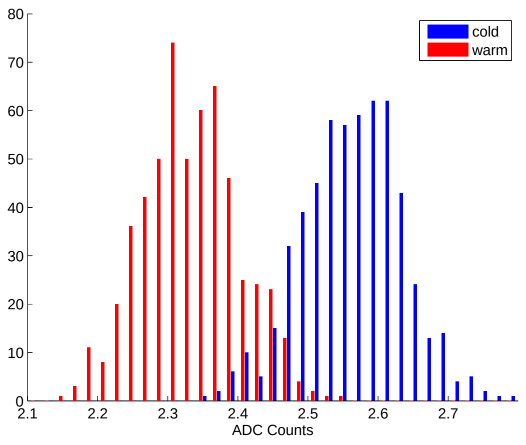


Figure 8: Noise Distribution for the L Hybrid

Figure 9 contains three histograms. Each of them displays the distribution of the average noise per Beetle for Beetles on L hybrids, but from different module assembly stages. There is one entry per Beetle per stage, even if it was tested more then once in a stage. The figure shows that the noise level is independent of the module stage. This result shows that the noise performance did not deteriorate during module production. That there are not as many entries form stage III as from stage II is expected as only a small subset of all modules has a third stage.

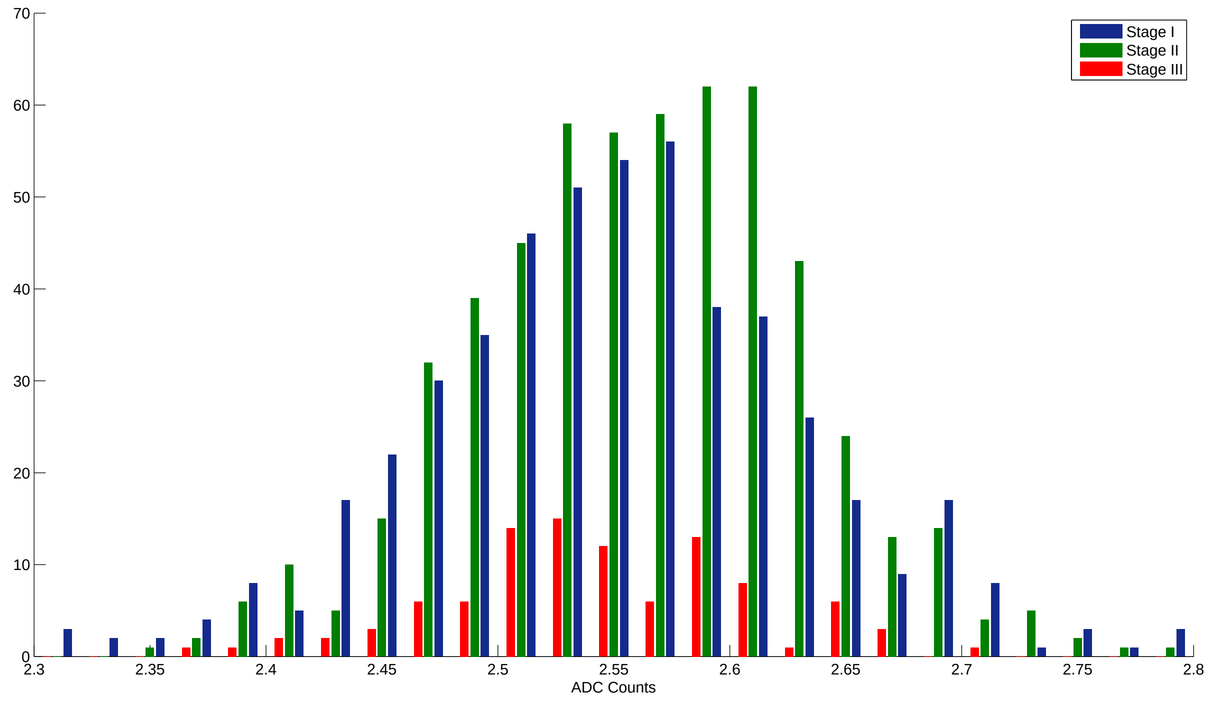


Figure 9: Noise Distribution for the L Hybrid

4.3 Leakage Current at 500V

Figure 10 contains the distribution of leakage currents from L and M readout sectors at production stage II. The entries correspond to the average of all leakage current measurements for a given readout section during this stage. Only values from tests at low temperatures were used and a small number of overflows and underflows were dropped. The L sectors have larger leakage currents as they contain 4 sensors while M sectors consist of either 2 or 3 sensors.

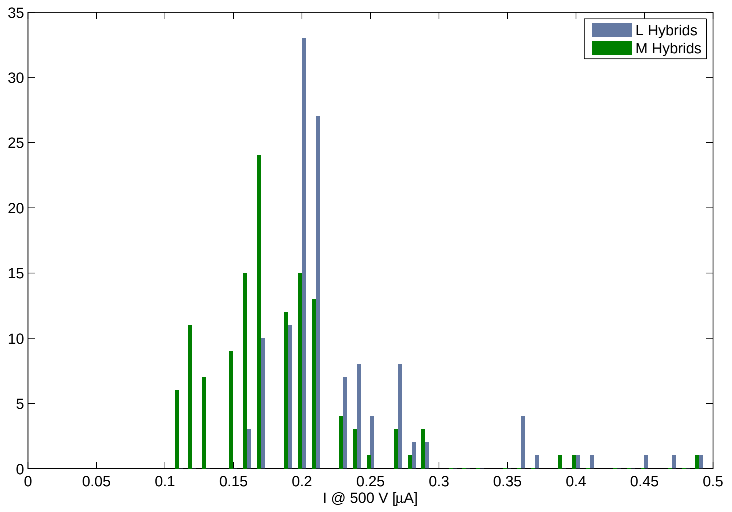


Figure 10: Leakage Current Distribution on Stage II

4.4 Remainder

Figure 11 shows a similar plot as Figure 10. Instead of the leakage current the remainder value from laser and internal testpulse scans are displayed. The entries were selected based on the same criteria as above. Here, the dependence on the readout section is inverse to the one of the leakage currents. This behaviour is unexpected and not yet fully understood. The missing distinction between the testpulse types can not explain this completely. A more in detail study of the remainder distribution will be presented in [6].

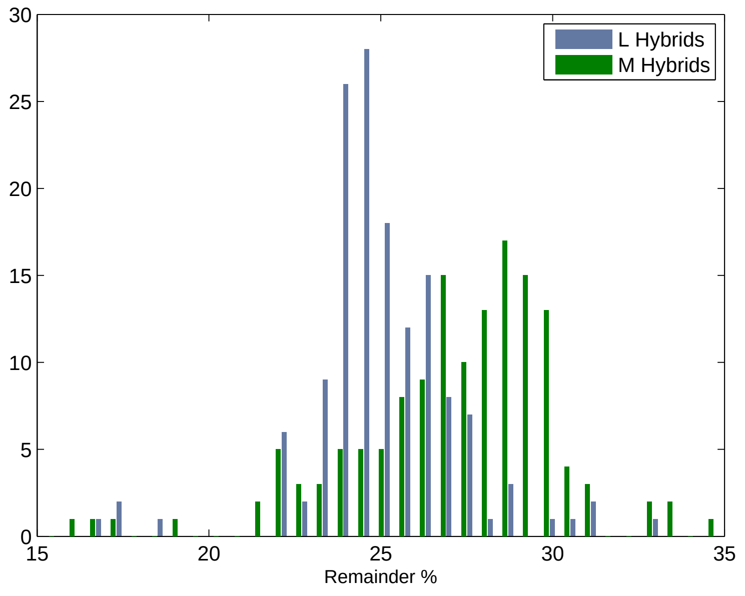


Figure 11: Remainder Distribution on Stage II

References

[1] R.Antunes-Nobrega et al., LHCb-LHCC-2003-030: “LHCb Reoptimized Detector Design and Performance - Technical Design Report”.

[2] J.Gassner, F.Lehner and S.Steiner, LHCb note 2004-110: “The Mechanical Design of the LHCb Silicon Trigger Tracker”.

[3] S.Marti i Garcia et al, Nucl. Instr. and Meth. A473 (2001) 128-135: “A model of charge collection for irradiated $p^{+}n$ detectors”.

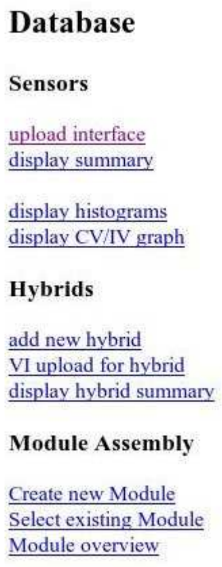
[4] R. P. Bernhard,Fermilab Thesis 2005-56: “Search for rare decays of the B/s0 meson with the D0 experiment”.

[5] A.Büchler, Forschungspraktikum (2006): “Research Work Project”

[6] V. Hangartner, Bachelor Thesis (2007): “LHCb TT Module Properties”.

[7] http://phppgadmin.sourceforge.net/

A Screenshots of the Web Interface



Database

Sensors

upload interface
display summary

display histograms
display CV/IV graph

Hybrids

add new hybrid
VI upload for hybrid
display hybrid summary

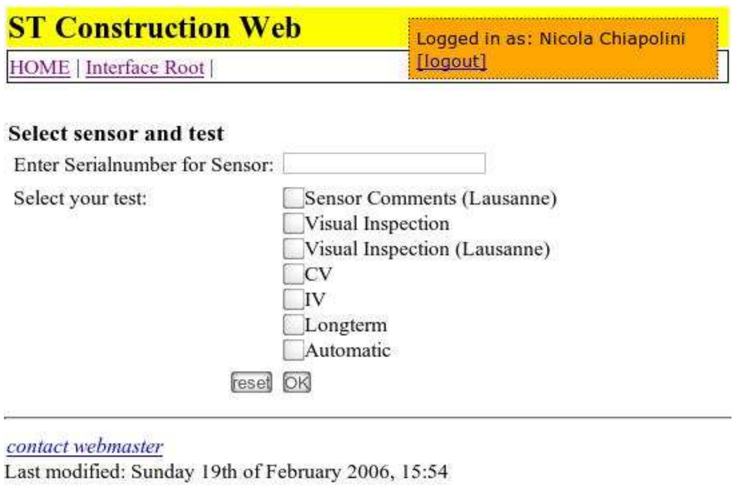
Module Assembly

Create new Module
Select existing Module
Module overview

Figure 12: Main Menu of the Interface

A.1 Sensors & Hybrids

A.1.1 Sensor Upload



ST Construction Web

Logged in as: Nicola Chiapolini [logout]

HOME | Interface Root |

Select sensor and test

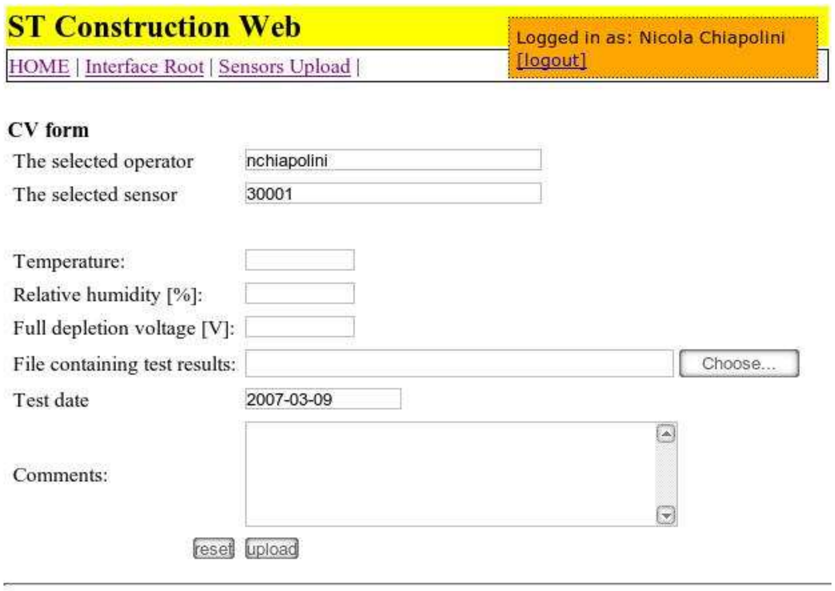
Enter Serialnumber for Sensor:

Select your test:
- ☐ Sensor Comments (Lausanne)
- ☐ Visual Inspection
- ☐ Visual Inspection (Lausanne)
- ☐ CV
- ☐ IV
- ☐ Longterm
- ☐ Automatic

reset OK

contact webmaster
Last modified: Sunday 19th of February 2006, 15:54

Figure 13: Upload Interface


ST Construction Web

HOME | Interface Root | Sensors Upload |

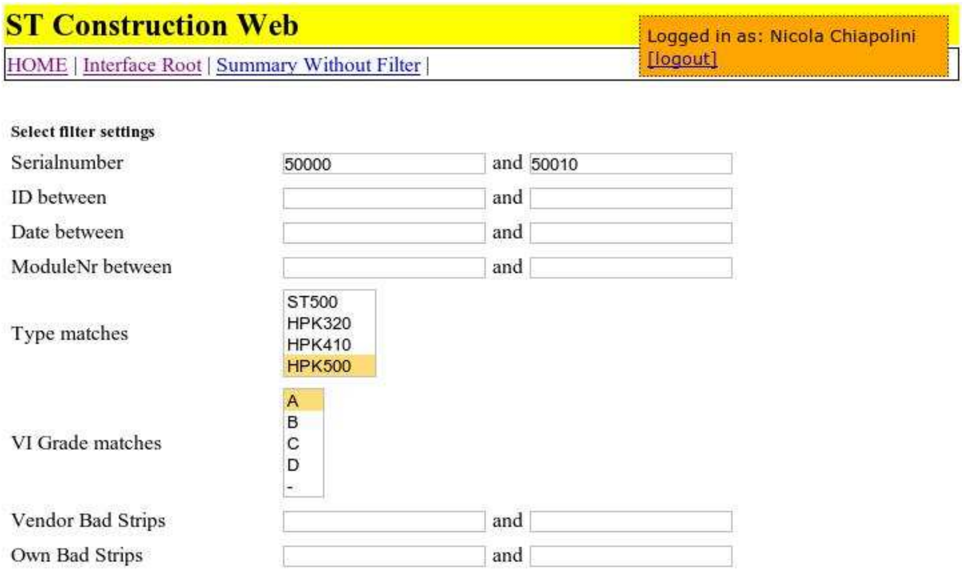
Logged in as: Nicola Chiapolini [logout]

CV form

The selected operator	nchiapolini
The selected sensor	30001
Temperature:	
Relative humidity [%]:	
Full depletion voltage [V]:	
File containing test results:	
Test date	2007-03-09
Comments:	

Figure 14: CV Test

A.1.2 Sensor Summary


ST Construction Web

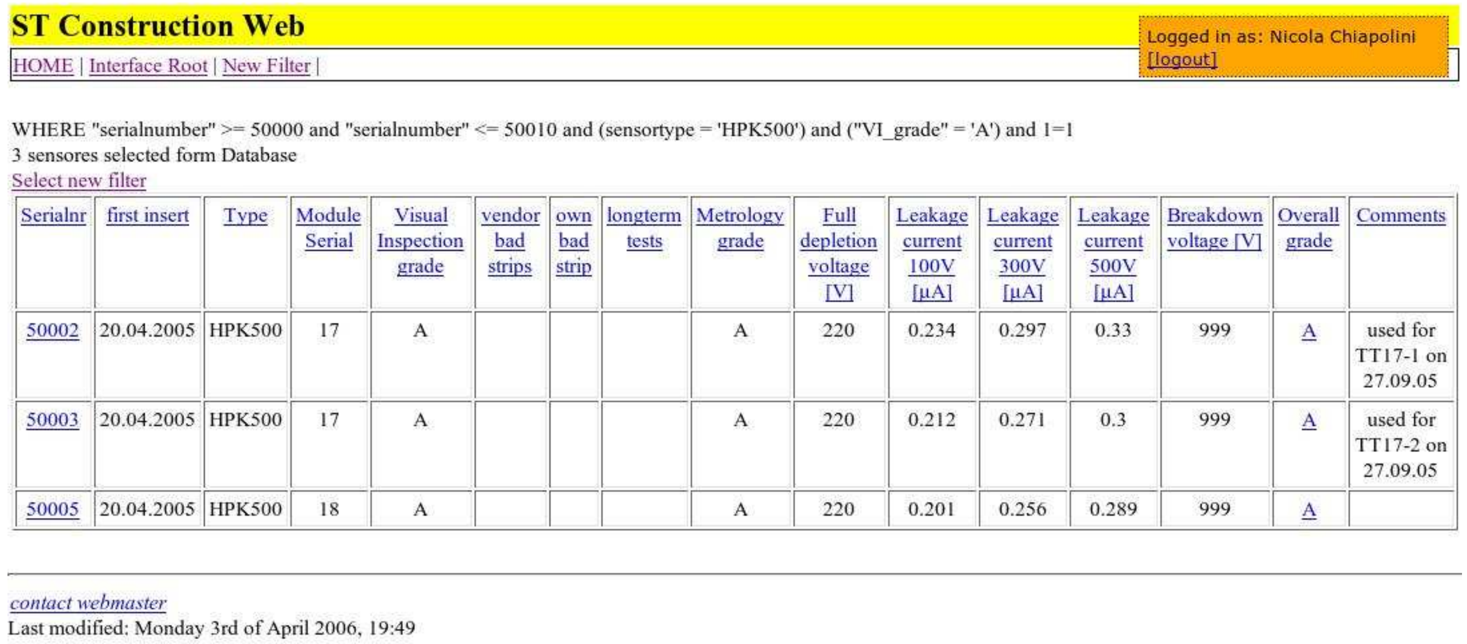
HOME | Interface Root | Summary Without Filter |

Logged in as: Nicola Chiapolini [logout]

Select filter settings

Serialnumber	50000	and	50010
ID between		and	
Date between		and	
ModuleNr between		and	
Type matches	ST500, HPK320, HPK410, HPK500 (selected)		
VI Grade matches	A (selected), B, C, D, -		
Vendor Bad Strips		and	
Own Bad Strips		and	

Figure 15: Filter for Sensor Summary



ST Construction Web

Logged in as: Nicola Chiapolini
[logout]

HOME | Interface Root | New Filter |

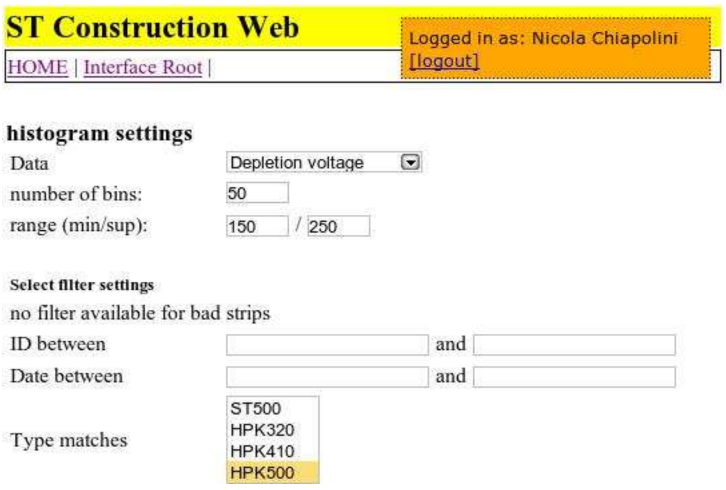
WHERE "serialnumber" >= 50000 and "serialnumber" <= 50010 and (sensortype = 'HPK500') and ("VI_grade" = 'A') and 1=1
3 sensores selected form Database
Select new filter

Serialnr	first insert	Type	Module Serial	Visual Inspection grade	vendor bad strips	own bad strip	longterm tests	Metrology grade	Full depletion voltage [V]	Leakage current 100V [µA]	Leakage current 300V [µA]	Leakage current 500V [µA]	Breakdown voltage [V]	Overall grade	Comments
50002	20.04.2005	HPK500	17	A				A	220	0.234	0.297	0.33	999	A	used for TT17-1 on 27.09.05
50003	20.04.2005	HPK500	17	A				A	220	0.212	0.271	0.3	999	A	used for TT17-2 on 27.09.05
50005	20.04.2005	HPK500	18	A				A	220	0.201	0.256	0.289	999	A	

contact webmaster
Last modified: Monday 3rd of April 2006, 19:49

Figure 16: Resulting Sensor Summary

A.1.3 Sensor Histogram



ST Construction Web

Logged in as: Nicola Chiapolini [logout]

HOME | Interface Root |

histogram settings

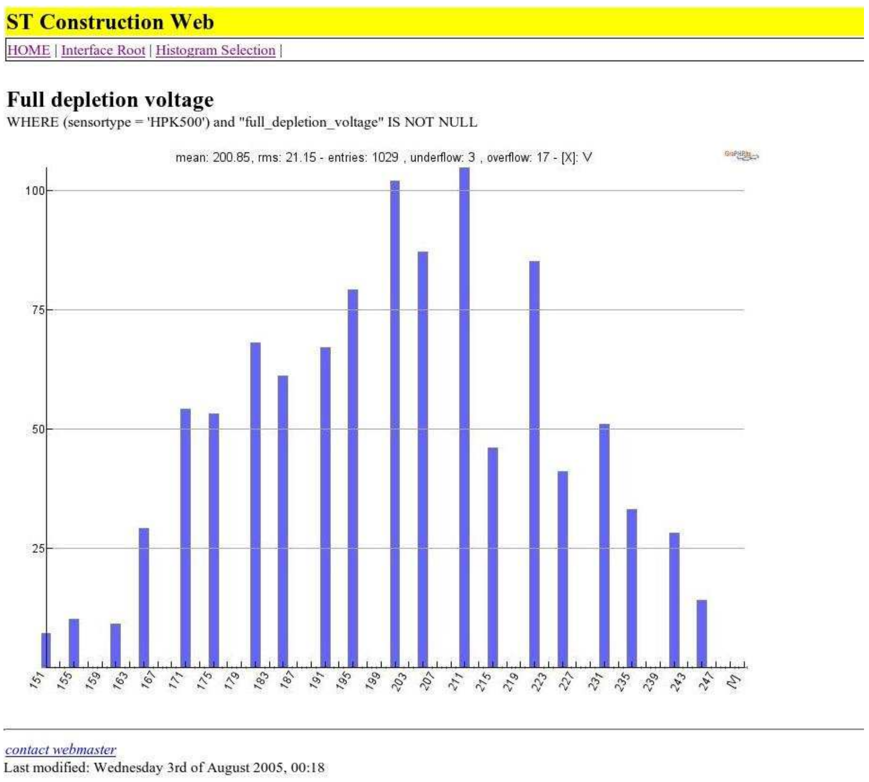
Data	Depletion voltage
number of bins:	50
range (min/sup):	150 / 250

Select filter settings

no filter available for bad strips

ID between		and	
Date between		and	
Type matches	ST500, HPK320, HPK410, HPK500		

Figure 17: Setup for Sensor Histogram



ST Construction Web

HOME | Interface Root | Histogram Selection |

Full depletion voltage

WHERE (sensortype = 'HPK500') and "full_depletion_voltage" IS NOT NULL

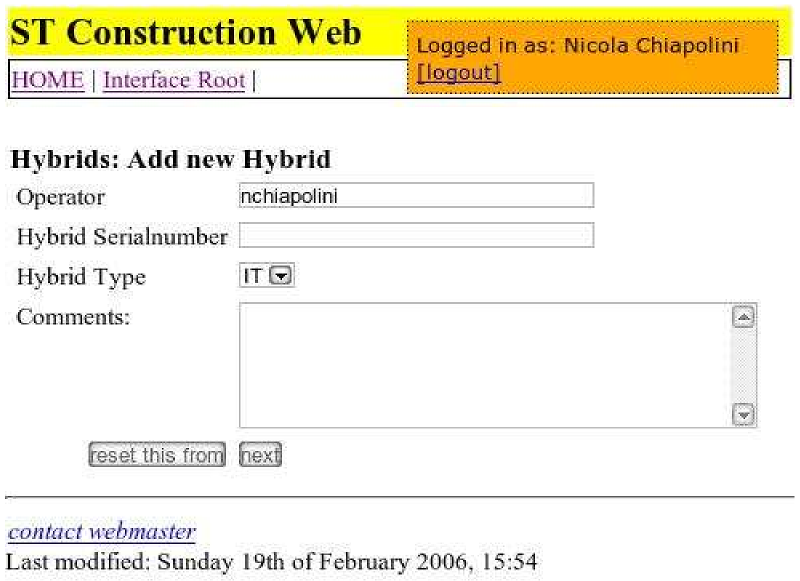
mean: 200.85, rms: 21.15 - entries: 1029 , underflow: 3 , overflow: 17 - [X]: V

contact webmaster

Last modified: Wednesday 3rd of August 2005, 00:18

Figure 18: Resulting Histogram

A.1.4 Hybrids



ST Construction Web

Logged in as: Nicola Chiapolini [logout]

HOME | Interface Root |

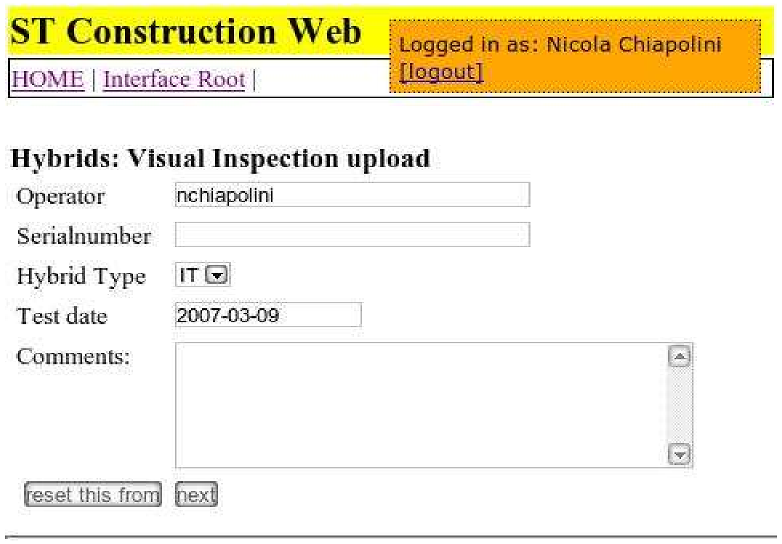
Hybrids: Add new Hybrid

Operator	nchiapolini
Hybrid Serialnumber	
Hybrid Type	IT
Comments:	

contact webmaster

Last modified: Sunday 19th of February 2006, 15:54

Figure 19: Add New Hybrid



ST Construction Web

Logged in as: Nicola Chiapolini [logout]

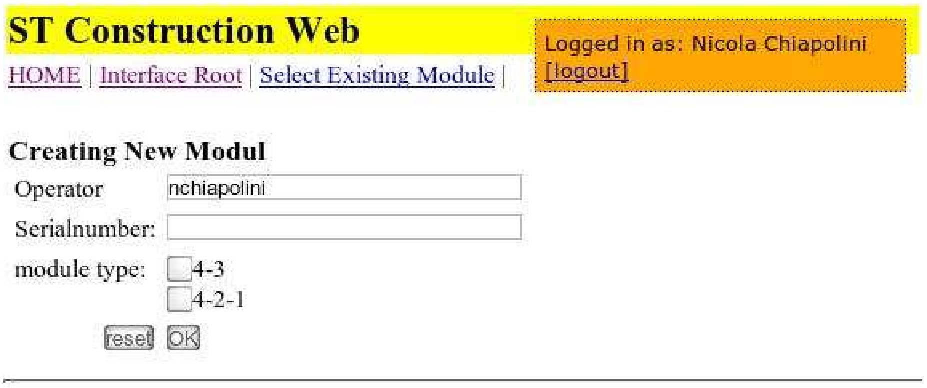
HOME | Interface Root |

Hybrids: Visual Inspection upload

Operator	nchiapolini
Serialnumber	
Hybrid Type	IT
Test date	2007-03-09
Comments:	

Figure 20: Visual Inspection for Hybrid

A.2 Module Assembly


ST Construction Web

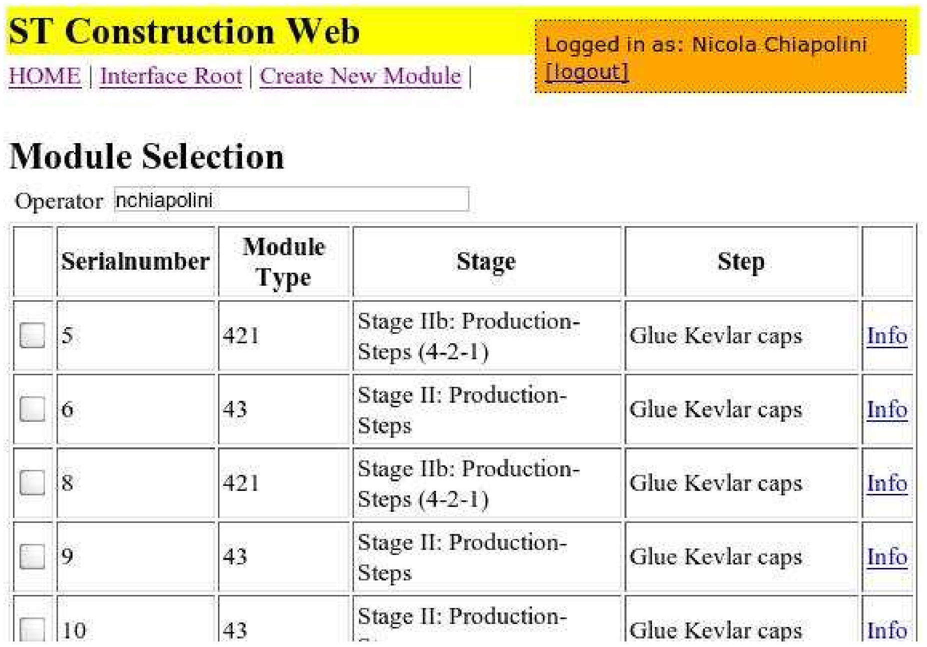
HOME | Interface Root | Select Existing Module |

Logged in as: Nicola Chiapolini [logout]

Creating New Modul

Operator	nchiapolini
Serialnumber:	
module type:	☐ 4-3 ☐ 4-2-1

Figure 21: Create New Module


ST Construction Web

HOME | Interface Root | Create New Module |

Logged in as: Nicola Chiapolini [logout]

Module Selection

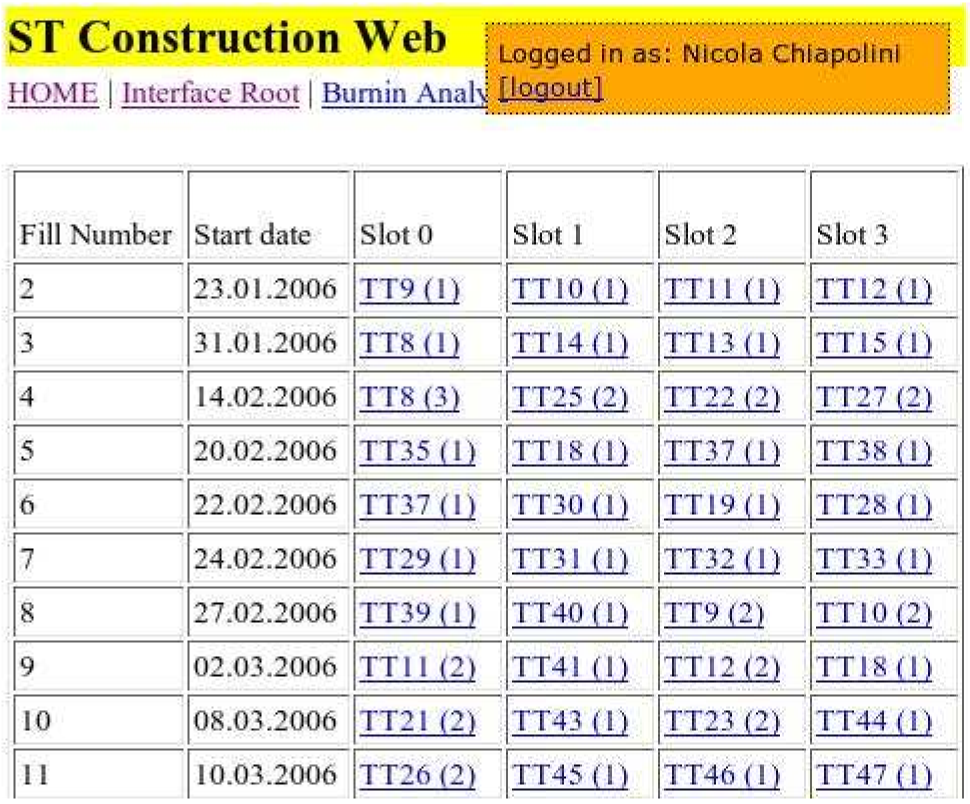
Operator nchiapolini

	Serialnumber	Module Type	Stage	Step	
☐	5	421	Stage IIb: Production-Steps (4-2-1)	Glue Kevlar caps	Info
☐	6	43	Stage II: Production-Steps	Glue Kevlar caps	Info
☐	8	421	Stage IIb: Production-Steps (4-2-1)	Glue Kevlar caps	Info
☐	9	43	Stage II: Production-Steps	Glue Kevlar caps	Info
☐	10	43	Stage II: Production-	Glue Kevlar caps	Info

Figure 22: Continue existing Module

A.3 Module Testing

A.3.1 Summary of the Fills



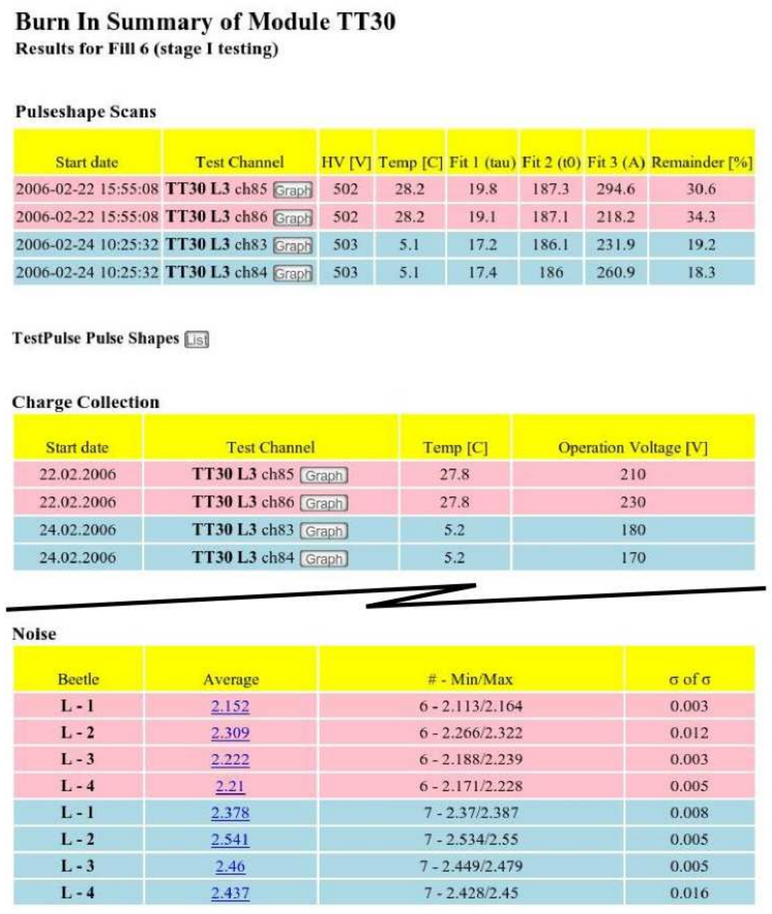
ST Construction Web

Logged in as: Nicola Chiapolini [logout]

HOME | Interface Root | Burnin Analy

Fill Number	Start date	Slot 0	Slot 1	Slot 2	Slot 3
2	23.01.2006	TT9 (1)	TT10 (1)	TT11 (1)	TT12 (1)
3	31.01.2006	TT8 (1)	TT14 (1)	TT13 (1)	TT15 (1)
4	14.02.2006	TT8 (3)	TT25 (2)	TT22 (2)	TT27 (2)
5	20.02.2006	TT35 (1)	TT18 (1)	TT37 (1)	TT38 (1)
6	22.02.2006	TT37 (1)	TT30 (1)	TT19 (1)	TT28 (1)
7	24.02.2006	TT29 (1)	TT31 (1)	TT32 (1)	TT33 (1)
8	27.02.2006	TT39 (1)	TT40 (1)	TT9 (2)	TT10 (2)
9	02.03.2006	TT11 (2)	TT41 (1)	TT12 (2)	TT18 (1)
10	08.03.2006	TT21 (2)	TT43 (1)	TT23 (2)	TT44 (1)
11	10.03.2006	TT26 (2)	TT45 (1)	TT46 (1)	TT47 (1)

Figure 23: Summary of the Fills



Burn In Summary of Module TT30

Results for Fill 6 (stage I testing)

Pulseshape Scans

Start date	Test Channel	HV [V]	Temp [C]	Fit 1 (tau)	Fit 2 (t0)	Fit 3 (A)	Remainder [%]
2006-02-22 15:55:08	**TT30 L3** ch85 Graph	502	28.2	19.8	187.3	294.6	30.6
2006-02-22 15:55:08	**TT30 L3** ch86 Graph	502	28.2	19.1	187.1	218.2	34.3
2006-02-24 10:25:32	**TT30 L3** ch83 Graph	503	5.1	17.2	186.1	231.9	19.2
2006-02-24 10:25:32	**TT30 L3** ch84 Graph	503	5.1	17.4	186	260.9	18.3

TestPulse Pulse Shapes List

Charge Collection

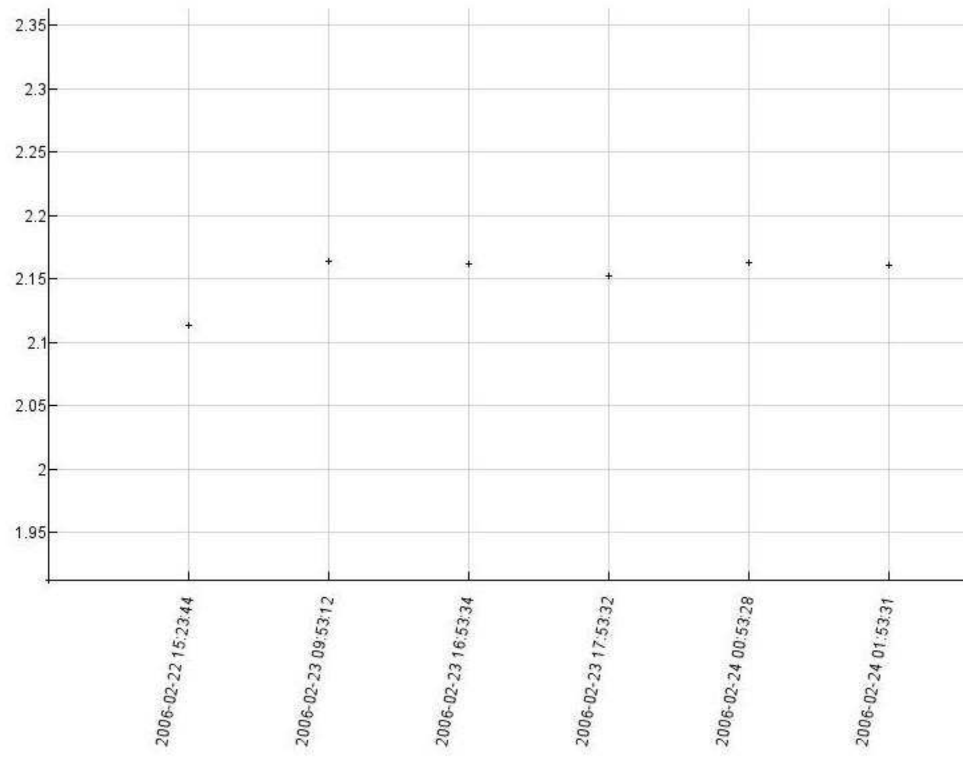
Start date	Test Channel	Temp [C]	Operation Voltage [V]
22.02.2006	**TT30 L3** ch85 Graph	27.8	210
22.02.2006	**TT30 L3** ch86 Graph	27.8	230
24.02.2006	**TT30 L3** ch83 Graph	5.2	180
24.02.2006	**TT30 L3** ch84 Graph	5.2	170

Noise

Beetle	Average	# - Min/Max	σ of σ
L - 1	2.152	6 - 2.113/2.164	0.003
L - 2	2.309	6 - 2.266/2.322	0.012
L - 3	2.222	6 - 2.188/2.239	0.003
L - 4	2.21	6 - 2.171/2.228	0.005
L - 1	2.378	7 - 2.37/2.387	0.008
L - 2	2.541	7 - 2.534/2.55	0.005
L - 3	2.46	7 - 2.449/2.479	0.005
L - 4	2.437	7 - 2.428/2.45	0.016

Figure 24: Summary of one Module

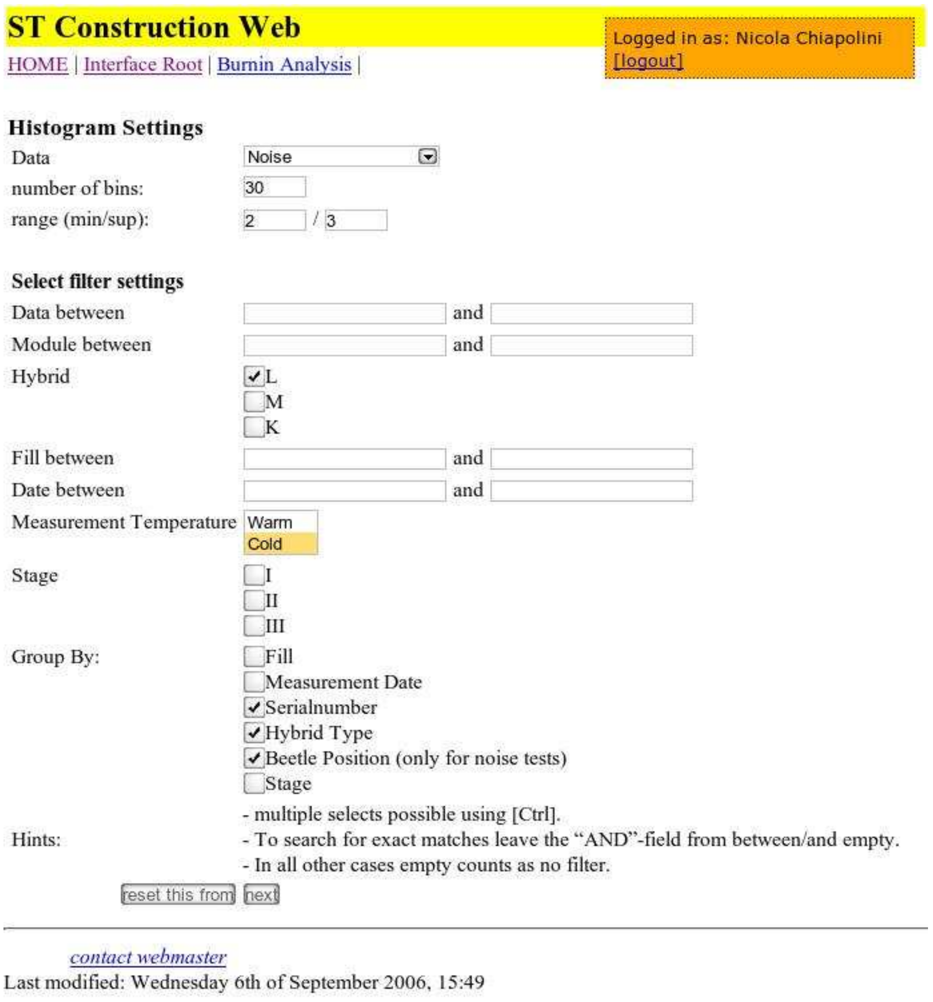
Noise Tests for TT30: L - 1 in fill 6 with temperature warm



Test Time	Temperature	Average of Channels	σ
2006-02-22 15:23:44	27.94	2.113 Channels Histo	0.087
2006-02-23 09:53:12	24.12	2.164 Channels Histo	0.088
2006-02-23 16:53:34	23.75	2.162 Channels Histo	0.091
2006-02-23 17:53:32	23.81	2.152 Channels Histo	0.084
2006-02-24 00:53:28	23.87	2.162 Channels Histo	0.092
2006-02-24 01:53:31	24	2.161 Channels Histo	0.091

Figure 25: Noise Details

A.3.2 Module Histogram



ST Construction Web

HOME | Interface Root | Burnin Analysis |

Logged in as: Nicola Chiapolini [logout]

Histogram Settings

Data	Noise
number of bins:	30
range (min/sup):	2 / 3

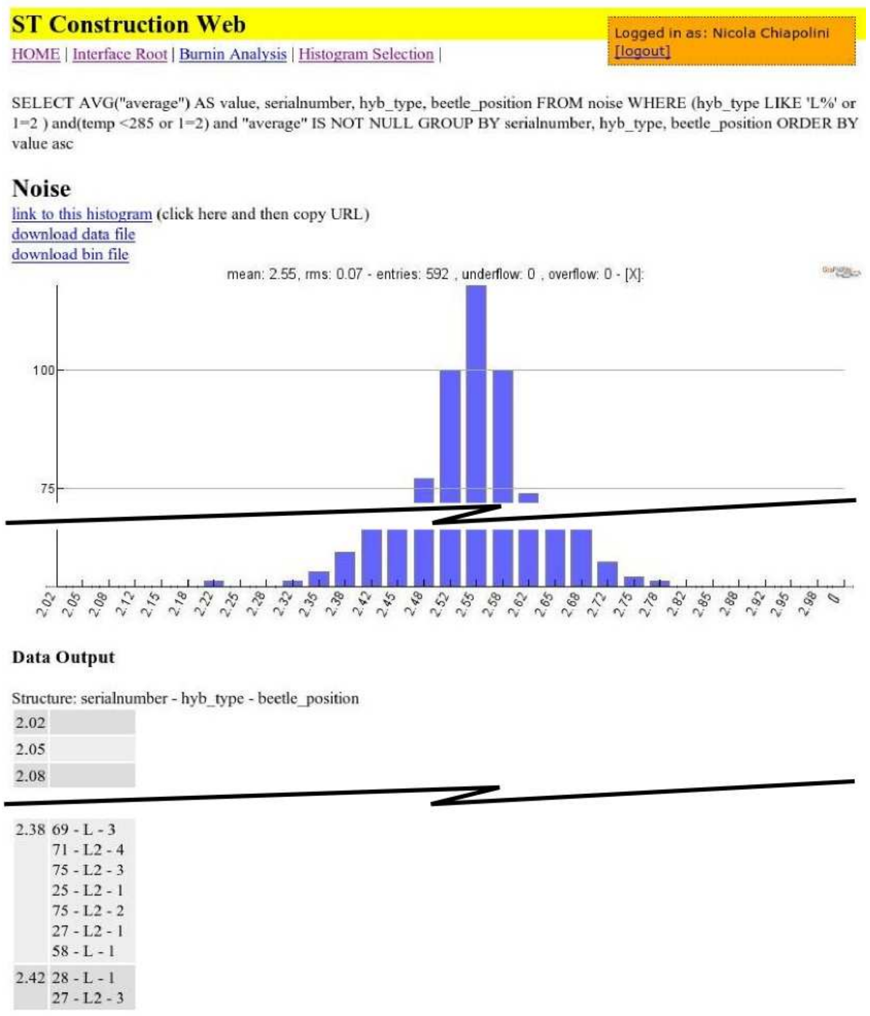
Select filter settings

Data between		and	
Module between		and	
Hybrid	[✓ L] M K		
Fill between		and	
Date between		and	
Measurement Temperature	Warm [✓ Cold]		
Stage	I II III		
Group By:	Fill, Measurement Date, [✓ Serialnumber], [✓ Hybrid Type], [✓ Beetle Position (only for noise tests)], Stage		

Hints:
- multiple selects possible using [Ctrl].
- To search for exact matches leave the "AND"-field from between/and empty.
- In all other cases empty counts as no filter.

Last modified: Wednesday 6th of September 2006, 15:49

Figure 26: Setup for Module Histogram


ST Construction Web
HOME | Interface Root | Burnin Analysis | Histogram Selection |
Logged in as: Nicola Chiapolini
[logout]
link to this histogram (click here and then copy URL)
download data file
download bin file
mean: 2.55, rms: 0.07 - entries: 592 , underflow: 0 , overflow: 0 - [X]:
100
75
2.02
2.05
2.08
2.12
2.15
2.18
2.22
2.25
2.28
2.32
2.35
2.38
2.42
2.45
2.48
2.52
2.55
2.58
2.62
2.65
2.68
2.72
2.75
2.78
2.82
2.85
2.88
2.92
2.95
2.98
0

```
SELECT AVG("average") AS value, serialnumber, hyb_type, beetle_position FROM noise WHERE (hyb_type LIKE 'L%' or
1=2 ) and(temp <285 or 1=2) and "average" IS NOT NULL GROUP BY serialnumber, hyb_type, beetle_position ORDER BY
value asc
```

Noise

Data Output

Structure: serialnumber - hyb_type - beetle_position

2.02	
2.05	
2.08	

2.38	69 - L - 3 71 - L2 - 4 75 - L2 - 3 25 - L2 - 1 75 - L2 - 2 27 - L2 - 1 58 - L - 1
2.42	28 - L - 1 27 - L2 - 3

Figure 27: Resulting Histogram