Report Semesterarbeit MORE-Web

Esteban Marín

2013-06-20

Abstract

This document describes an undergraduate project (Semesterarbeit) that is part of the physics studies at the ETH in Zurich, Switzerland. An introduction to physical research in particle physics is given. The need of particle accelerators and colliders for understanding the fundamental laws of nature is explained. A specific example is the CERN in Geneva, with an experiment called "CMS", which is of particular interest for this project.
Since parts of the "CMS" detector, namely the "CMS Pixel Detector", a silicon particle tracker for analyzing the momentum of particles, need to be upgraded, its components, the modules, need to be tested and qualified.
This Semesterarbeit is about the relaunch of a software "MoRe-Web" that analyzes test results and displays the qualification. The concept, as well as the implementation of the software are explained in detail. A step-by-step guide on how to use the software is given. All requirements for "MoRe-Web" could be met and the software is up and running, but will still be further improved in order to fit future changes in the tests.

Contents

1 Introduction

This document describes an undergraduate project (Semesterarbeit) that is part of the physics studies at the ETH in Zurich, Switzerland. Professor Wallny offered the opportunity to participate in a project called "CMS Pixel Detector". But the first question that arises: what is the need for writing this paper?

1.1 High Energy Physics

The goal of physics is to describe nature as accurately as possible, so over many years many people have worked on theories in order to achieve this goal as good as possible. Up to now the physics of microscopic particles is best described by the so called "Standard Model" (SM), which claims that all matter in the universe is made up of elementary particles and that interactions between those can be associated with other particles [1].

In order to verify the SM, the particles and its properties predicted have to be found and checked, which is when it comes to high energy particle physics. Many particles have already been found and the SM was found to accurately describe the particles known so far. In the mid-late twentieth century, a close connection between two of the four fundamental forces (interaction) predicted by the standard model was assumed. The SM describes those two forces with one theory. It states that well known interactions like electricity, magnetism, light and some types of radioactivity can be described by the concept of the so-called "electroweak force". A problem that arose concerning the interaction was that although all interaction particles were predicted to be massless, which fits the observations in the case of photons, the W and Z bosons were found to have a mass corresponding to about 100 times the mass of a proton.
In order to solve this discrepancy between theory and experiments, Robert Brout, François Englert and Peter Higgs proposed the Brout-Englert-Higgs mechanism, which "gives a mass to the W and Z when they interact with an invisible field, now called the "Higgs field", which pervades the universe. [...] The more a particle interacts with this field, the heavier it is.". [2]
As mentioned before, the SM associates a particle with the interaction between elementary particles, which in the case of the Higgs field is called "Higgs Boson". The higgs boson was a particle of particular interest, since it was a fundamental concept in the SM and the discovery of this particle was therefore crucial for the further verificiation of the SM. The higgs boson, among other particles, was is being looked for at CERN's Large Hadron Collider, a gigantic machine for discovering elementary particles and carrying out experiments with them. In 2012 on July 4, many physicists were really excited about an announcement made by the ATLAS and CMS experiments, which claimed the obervation of "new particle in the mass region around 126 GeV", consistent with the properties of the Higgs boson. This discovery was a huge progress in the verification of the SM, but also the beginning of even further research. This is where the "CMS Pixel Detector" project mentioned before comes in. It is a detector which is designed to detect and classify, among other particles, Higgs bosons. [2]

1.2 Tasks of the Semesterarbeit

The main task of this Semesterarbeit was to relaunch an application for qualifying the test results of parts of the CMS pixel detector.

2 CERN, LHC and CMS Pixel Detector

2.1 Cern

The two main question at CERN, the European Organization for Nuclear Research, can be summarized as follows: “What is the universe made of? How did it start?” [3].
With these two questions in mind, physicists and engineers are trying to analyze the fundamental structure of the universe. At CERN, they are provided with all the tools needed for this research. The “heart” of cern are the custom -built particle accelerators and detectors. The collisionning process of particles at velocities close to the speed of light allows physicists to study the structure of the particles and the fundamental laws of nature. The particles are boosted to high energies in the accelerators and are then made to collide with each other or with fixed targets. The results of these collisions are then observed and recorded in the detectors. The CERN was founded in 1954 and since then has helped particle physicists to develop theories and check the SM and therefore better understand the laws of nature.[3]

2.2 LHC

One part of CERN is the Large Hadron Collider (LHC), which, as its name implies, is a large particle accelerator and collider consisting of a “27-kilometre ring of superconducting magnets with a number of accelerating structures to boost the energy of the particles along the way”. Inside the LHC, two high-energy (close to the speed of light) particle beams travelling in opposite directions in two separate vacuum tubes are forced to collide at a specific point. In order to keep the particles in the ring, their path is deviated using strong magnetic fields generated by superconducting electromagnets. “The particles are so tiny that the task of making them collide is akin to firing two needles 10 kilometres apart with such precision that they meet halfway!”.[4]

2.3 CMS

As already outlined in section 1, the Compact Muon Solenoid (CMS) is a particle detector forming part of the LHC. The search for the Higgs boson is among the main purposes, although it is designed to be a general purpose experiment and fit a wide range of physical research topics. It has the same scientific goals as the ATLAS experiment at CERN, but uses a different approach concerning the technical solutions, and is therefore expected to independently provide similar results as the ATLAS, despite the different technical implementation. A huge solenoid magnet provides a magnetic field of about 4 Tesla. The CMS consists of several different parts at different distances from the beam in the center of the detector.[5] A sectional view of the CMS is shown in figure 1

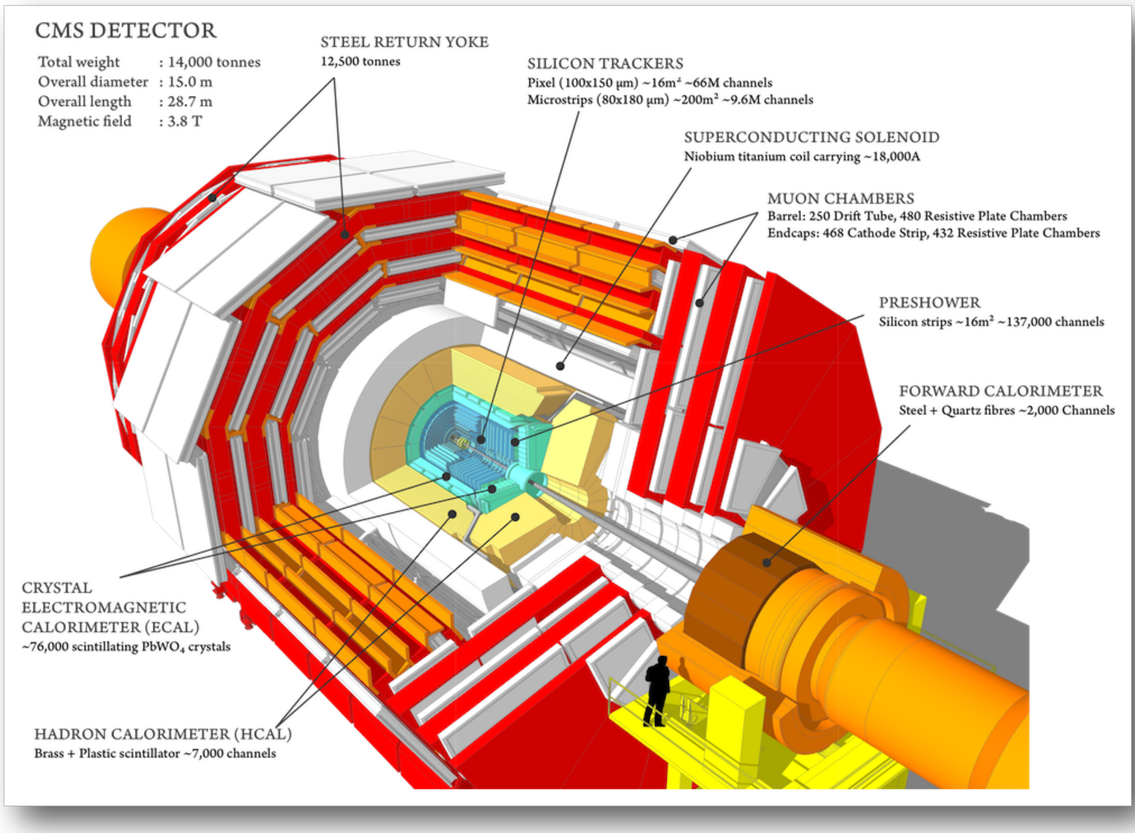

CMS DETECTOR
Total weight : 14,000 tonnes
Overall diameter : 15.0 m
Overall length : 28.7 m
Magnetic field : 3.8 T
STEEL RETURN YOKE
12,500 tonnes
SILICON TRACKERS
Pixel (100x150 μm) ~16m² ~66M channels
Microstrips (80x180 μm) ~200m² ~9.6M channels
SUPERCONDUCTING SOLENOID
Niobium titanium coil carrying ~18,000A
MUON CHAMBERS
Barrel: 250 Drift Tube, 480 Resistive Plate Chambers
Endcaps: 468 Cathode Strip, 432 Resistive Plate Chambers
PRESHOWER
Silicon strips ~16m² ~137,000 channels
FORWARD CALORIMETER
Steel + Quartz fibres ~2,000 Channels
CRYSTAL ELECTROMAGNETIC CALORIMETER (ECAL)
~76,000 scintillating PbWO₄ crystals
HADRON CALORIMETER (HCAL)
Brass + Plastic scintillator ~7,000 channels

Figure 1: Sectional view of the CMS Detector [6]

2.3.1 CMS Pixel Detector

The innermost layer of the CMS is the so-called “silicon tracker”, a tracker detector that measures the momentum of the particles passing by tracking their path in a magnetic field using several key points to measure the position and calculate the resulting momentum. At the very core of the detector, the silicon pixels absorb energy from passing particles. This signal is then amplified and recorded together with the position information of the particle. [7] The silicon tracker (CMS Pixel Detector) is divided up into pixel barrel modules that are arranged in three “pixel barrels” with symmetry axis parallel to the particle beam.

Pixel Barrel Modules

In the following parts of this document, the Pixel Barrel Modules will simply be called “Modules”. Each Module contains 66560 sensor pixels that are connected to a dedicated pixel unit cell (PUC) on one of the 16 readout chips (ROCs) by indium bumps and arranged in 52 columns and 80 rows per ROC. [9] For a detailed description of the structure of the modules see [9].

3 Module Qualification

3.1 Upgrade of CMS Pixel Detector

In 2014, an upgrade of the CMS pixel detector will be implemented as a part of a series of upgrades of the LHC. The full detector is expected to be ready by the end of 2016. [8] In order to include only modules that are working as expected, every single module of the pixel detector has to undergo a module qualification that calculates an overall grade according to the test results gained by performing several functionality tests using a custom software packages called “psi46expert” and “elComandante” and a general purpose calculation framework “ROOT” that was developed at CERN. [9]

3.2 Tests

There are several different tests that concern either the entire module or the individual ROCs. The set of all module and ROC tests is called “Module Fulltest” and is performed for each module individually and three times at two different temperatures.

3.2.1 Bump Bonding Test

An example of a ROC test is the bump bonding test. Its purpose is to test the quality of the bump bonding between the silicon sensor and the PUC. A scheme of the relevant parts for this test is given in figure 2. In the test routine, a calibration signal can “either be injected directly to the preamplifier (using switch 1 in figure 2) or to a pad on the ROC surface (using switch 2 in figure 2)”. A hit is simulated by injecting the signal to the pad, inducing a charge in the silicon and therefore generating a “hit” signal. Unfortunately, the expected result is not perfect, that is if the bump quality is poor or not connecting at all there might still be a hit signal, which is probably due to a so-called “cross-talk” phenomenon. The cross-talk describes a parasitic coupling between the calibration voltage line and the preamplifier (see figure 9). The algorithm that was developed to circumvent this problem is described as follows.

- **Vcal Threshold** Determination of the threshold value of the calibration voltage that generates a hit although switch 1 (figure 9) is open with switch 2 closed.
- **Parasitic cross-talk** Determination of the threshold value of the calibration voltage that generates a hit although both switches are open due to the parasitic capacitance.
- **Difference** The difference of the two threshold values is found to be very close to zero for a missing bump bond and unequal to zero for a working bump bond. The difference can therefore be used to determine the quality of the bump bonding.

[9] The result of a bump bonding test summarized for all ROCs on the module can be seen in figure 9 in the section labeled “Bump Bonding Problems”. The differences between the two threshold values for the pixels of a selected chip are summarized in figure 11 in the section labeled “Bump Bonding”. Since the overview of the bump bonding problems in figure 9 shows no defective pixel, all differences in the bump bonding test of the chip are expected to be non-zero, which is indeed the case.

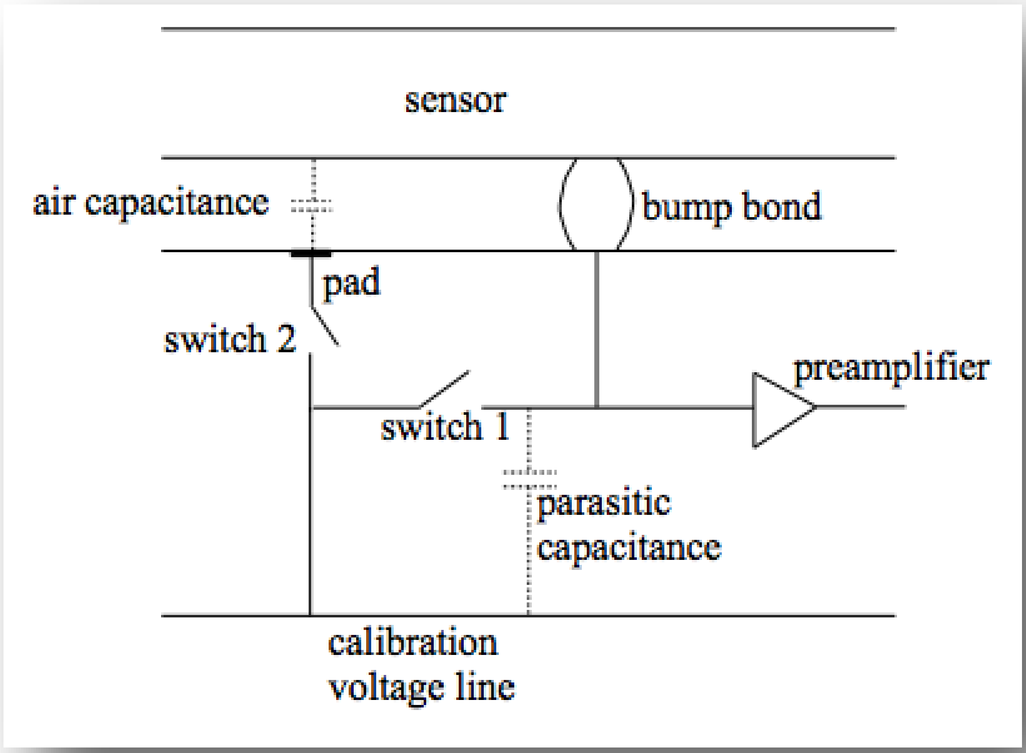


Figure 2: Scheme of relevant parts for bump bonding test [9]

3.3 Grading

The grade of a module is calculated by comparing specific values of the test results with predefined parameters. Grade A and B allow a module to be installed in the actual detector, although only those with grade A are located at the first layers near to the beam. Modules with grade C are not used.

4 Web Interface and Analysis Scripts

4.1 Introduction

Before the beginning of this semesterarbeit, an existing application was used to display all module qualification test results. As mentioned before, the main part of this semesterarbeit is to relaunch the module qualification application and adapt it to current needs, e.g. compatibility with new digital modules, more functionality, better user interface, etc.

4.2 Methodology

As a first step, a basic understanding of the CMS Pixel Detector (see section 2 and the module qualification tests (see section 3) was needed in order to analyze the requirements for the web application. In the following sections, the web interface is referred to as "**MoRe-Web**" (CMS Pixel Detector Module Qualification Result Web Interface). The general procedure was then defined to be as follows:

- Analysis of existing application
- Definition of Requirements
- Establishment of a concept for the new MoRe-Web
- Implemenation of MoRe-Web
- Testing

During the entire process, a repeated quality check is needed. It analyses the following aspects:

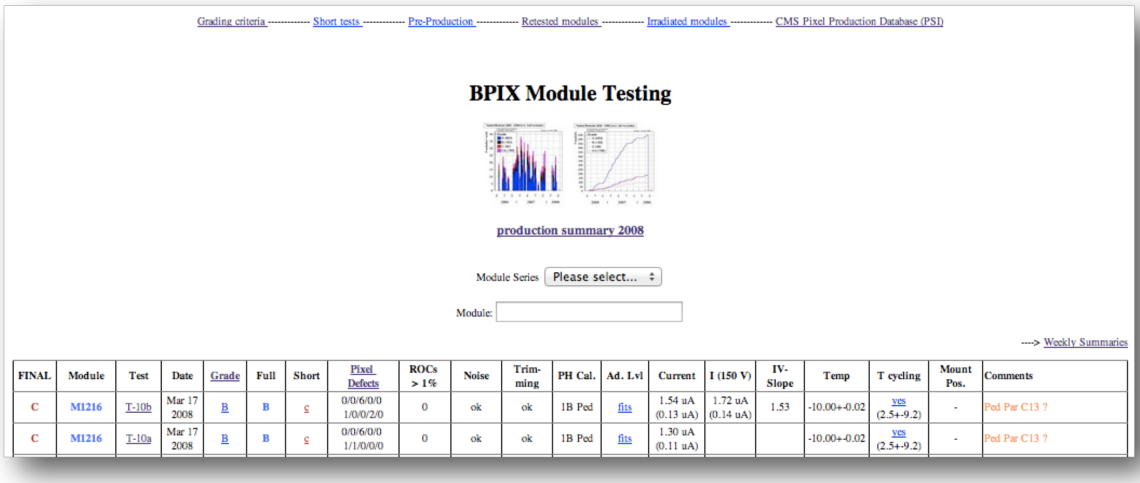
- Progress
- Meeting of requirements
- Unexpected problems
- Further improvements

4.3 Analysis of Existing Application

The starting point of the module qualification test results is `http://cmspixel.phys.ethz.ch` [10].

4.3.1 BPIX Module Testing

At `/moduleTests/moduleDB/prodTable.php` an application called "BPIX Module Testing" is located. The basic purpose of this script is to display an overview table for all module qualification tests and detail pages with further information for each tested module.


Grading criteria ------------ Short tests ------------ Pre-Production ------------ Retested modules ------------ Irradiated modules ------------ CMS Pixel Production Database (PSI)

BPIX Module Testing

production summary 2008

Module Series: Please select...

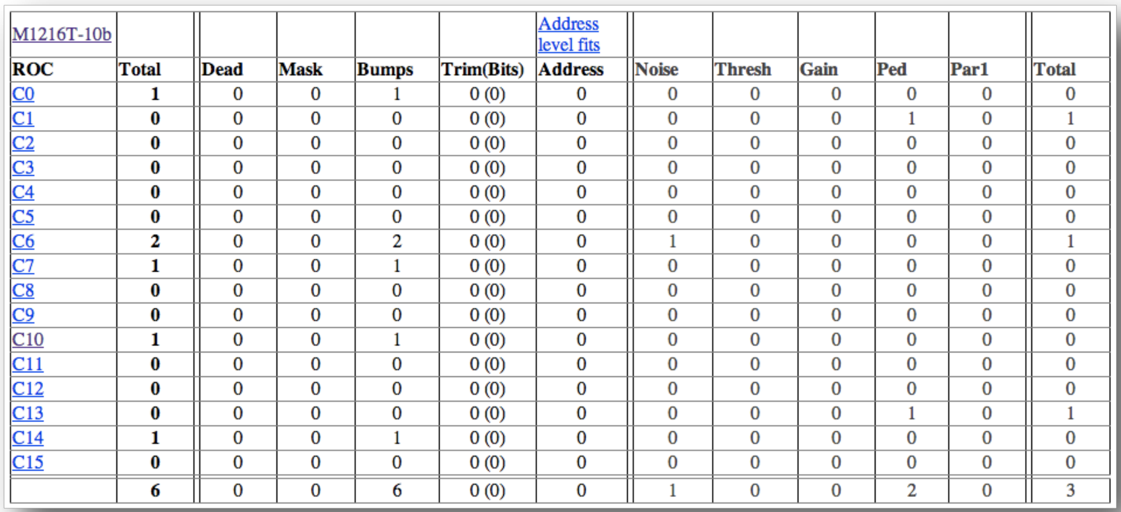
Module:

----> Weekly Summaries

FINAL	Module	Test	Date	Grade	Full	Short	Pixel Defects	ROCs > 1%	Noise	Trim-ming	PH Cal.	Ad. Lvl	Current	I (150 V)	IV-Slope	Temp	T cycling	Mount Pos.	Comments
C	M1216	T-10b	Mar 17 2008	B	B	c	0/0/6/0/0 1/0/0/2/0	0	ok	ok	1B Ped	fits	1.54 uA (0.13 uA)	1.72 uA (0.14 uA)	1.53	-10.00+-0.02	yes (2.5+-9.2)	-	Ped Par C13 ?
C	M1216	T-10a	Mar 17 2008	B	B	c	0/0/6/0/0 1/1/0/0/0	0	ok	ok	1B Ped	fits	1.30 uA (0.11 uA)			-10.00+-0.02	yes (2.5+-9.2)	-	Ped Par C13 ?

Figure 3: BPIX Module Testing Overview [10]

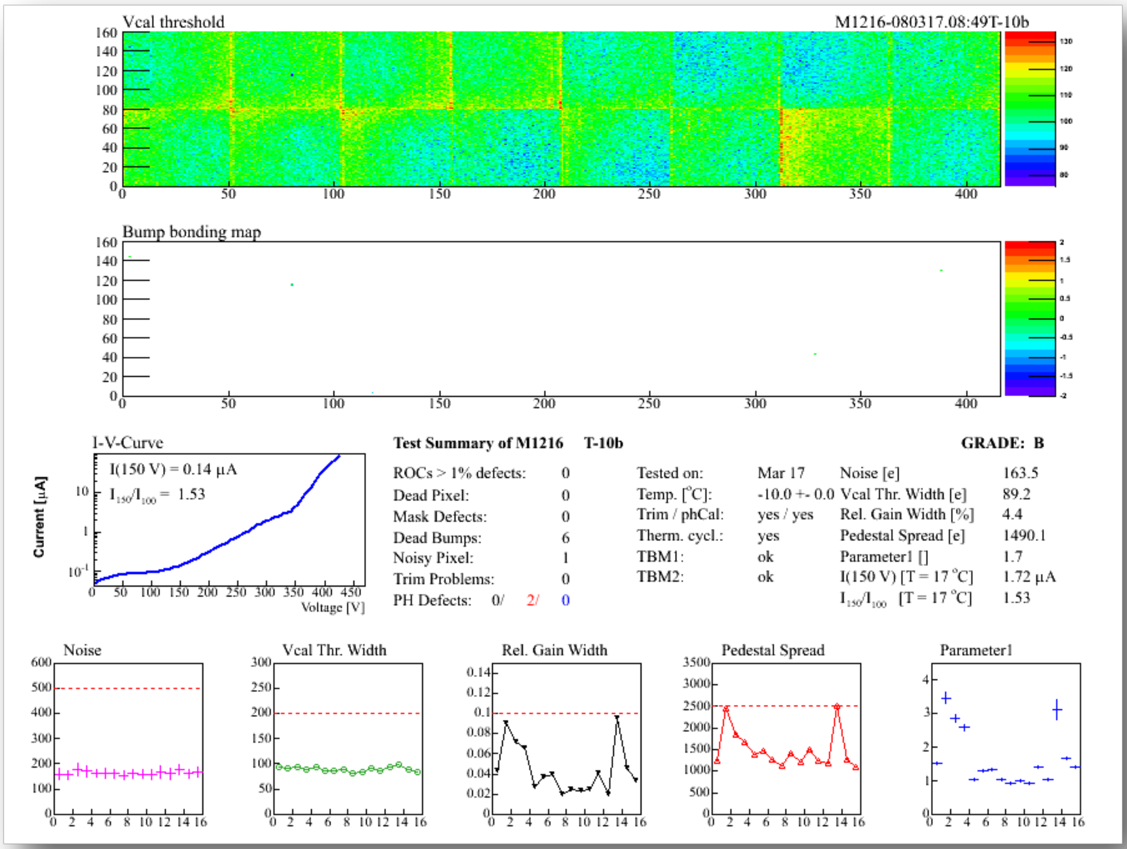
Frontend The frontend offers an overview list of all module qualification tests as can be seen in figure 3. By clicking on a module test, an overview as seen in figure 4 is presented. A link in the upper left corner leads to an overview graph for the module test as shown in figure 5. Also, there is a link horizontally centered which leads to an address levels test graph as shown in figure 6. By clicking a specific ROC identifier, the corresponding test results show up as given in figure 7.



M1216T-10b						Address level fits						
ROC	**Total**	**Dead**	**Mask**	**Bumps**	**Trim(Bits)**	**Address**	**Noise**	**Thresh**	**Gain**	**Ped**	**Par1**	**Total**
C0	**1**	0	0	1	0 (0)	0	0	0	0	0	0	0
C1	**0**	0	0	0	0 (0)	0	0	0	0	1	0	1
C2	**0**	0	0	0	0 (0)	0	0	0	0	0	0	0
C3	**0**	0	0	0	0 (0)	0	0	0	0	0	0	0
C4	**0**	0	0	0	0 (0)	0	0	0	0	0	0	0
C5	**0**	0	0	0	0 (0)	0	0	0	0	0	0	0
C6	**2**	0	0	2	0 (0)	0	1	0	0	0	0	1
C7	**1**	0	0	1	0 (0)	0	0	0	0	0	0	0
C8	**0**	0	0	0	0 (0)	0	0	0	0	0	0	0
C9	**0**	0	0	0	0 (0)	0	0	0	0	0	0	0
C10	**1**	0	0	1	0 (0)	0	0	0	0	0	0	0
C11	**0**	0	0	0	0 (0)	0	0	0	0	0	0	0
C12	**0**	0	0	0	0 (0)	0	0	0	0	0	0	0
C13	**0**	0	0	0	0 (0)	0	0	0	0	1	0	1
C14	**1**	0	0	1	0 (0)	0	0	0	0	0	0	0
C15	**0**	0	0	0	0 (0)	0	0	0	0	0	0	0
	6	0	0	6	0 (0)	0	1	0	0	2	0	3

Figure 4: BPIX Module Testing ROC Overview [10]

Source The script located at `moduleTests/moduleDB/prodTable.php` generates the overview table seen in figure 5. It basically scans the `moduleTests/moduleDB/` and compares parts of the file or folder names to some parameters (e.g. series or search string). It parses the `summaryTest.txt` file of a module test and retrieves information such as test date, module number, grading, etc. from it.



Test Summary of M1216 T-10b **GRADE: B**

ROCs > 1% defects:	0	Tested on:	Mar 17	Noise [e]	163.5
Dead Pixel:	0	Temp. [°C]:	-10.0 +- 0.0	Vcal Thr. Width [e]	89.2
Mask Defects:	0	Trim / phCal:	yes / yes	Rel. Gain Width [%]	4.4
Dead Bumps:	6	Therm. cycl.:	yes	Pedestal Spread [e]	1490.1
Noisy Pixel:	1	TBM1:	ok	Parameter1 []	1.7
Trim Problems:	0	TBM2:	ok	I(150 V) [T = 17 °C]	1.72 µA
PH Defects:	0/ 2/ 0			I_{150}/I_{100} [T = 17 °C]	1.53

Figure 5: BPIX Module Testing Module Test Overview [10]

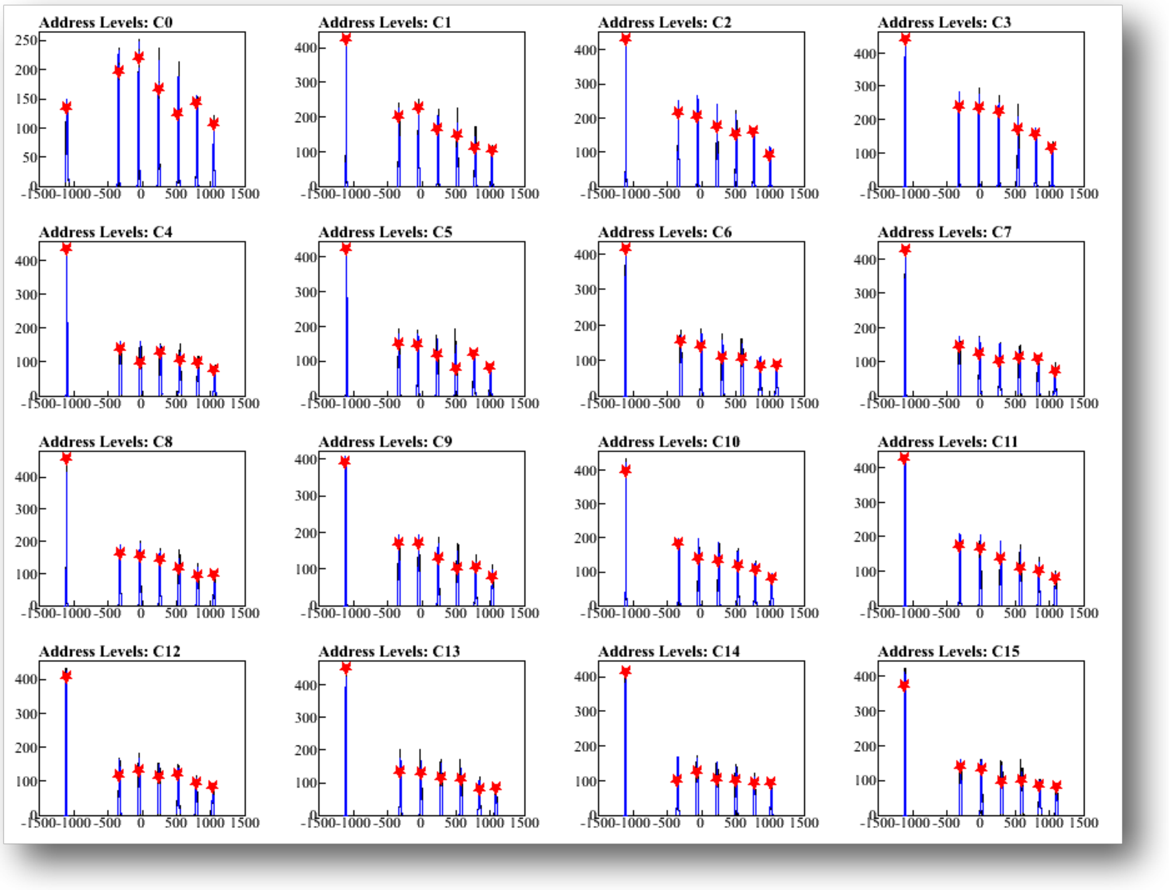
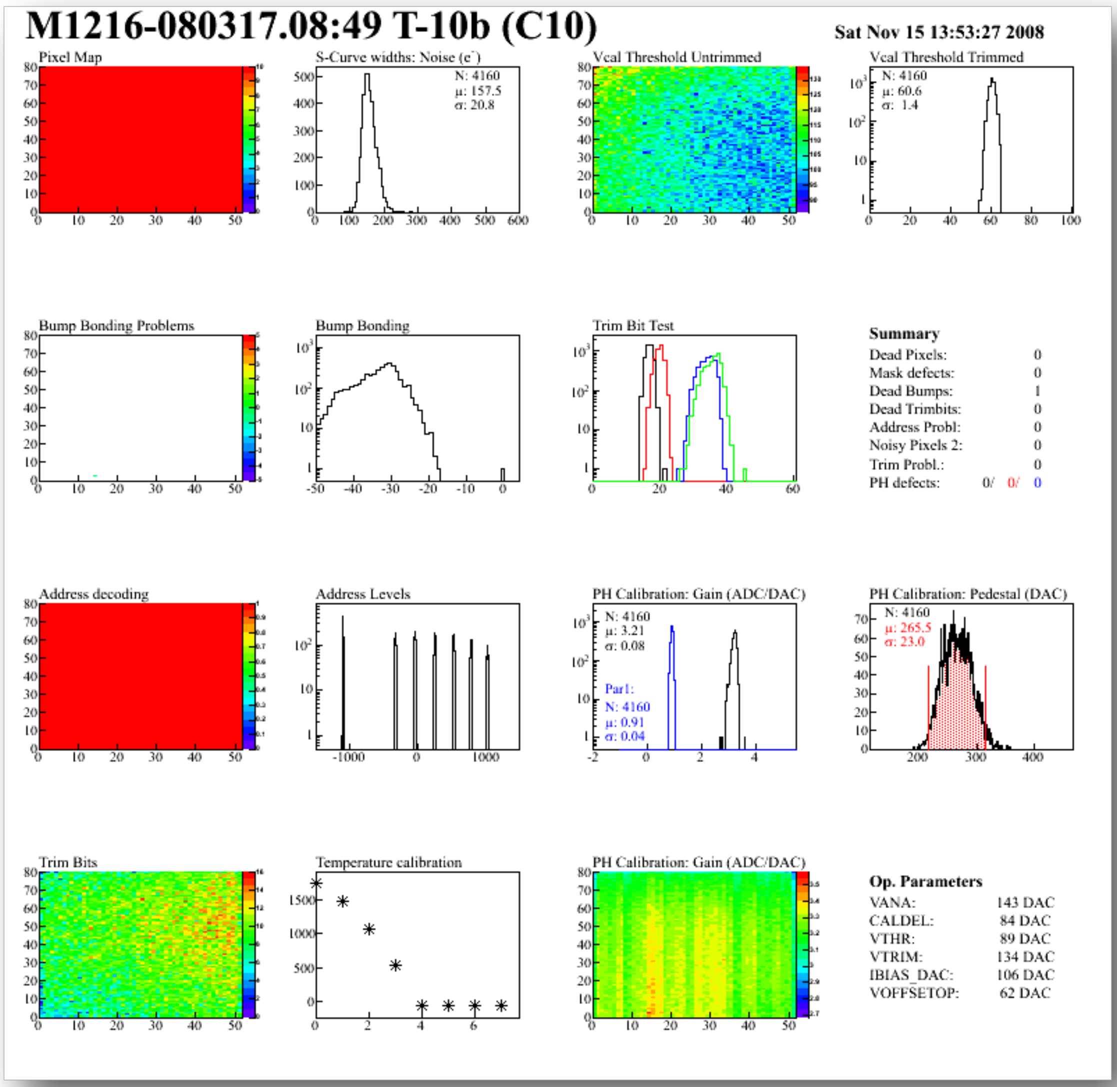


Figure 6: BPIX Module Testing Module Address Levels [10]



Summary

Dead Pixels:	0
Mask defects:	0
Dead Bumps:	1
Dead Trimbits:	0
Address Probl:	0
Noisy Pixels 2:	0
Trim Probl.:	0
PH defects:	0/ 0/ 0

Op. Parameters

VANA:	143 DAC
CALDEL:	84 DAC
VTHR:	89 DAC
VTRIM:	134 DAC
IBIAS_DAC:	106 DAC
VOFFSETOP:	62 DAC

Figure 7: BPIX Module Testing ROC Test [10]

4.3.2 Analysis Scripts

The analysis scripts that process the testing data an generate graphs etc. are located in `/home/peller/sdvlp/elcommandante/trunk/fitting` on the lion.ethz.ch server.

Module Summary Page The module summary page in figure 5 is generated by the script `moduleSummaryPage.C` Some important functions are listed below:

- `moduleSummary`
  Generates an image file for the module summary page. Generates the subplots using other functions in the file. Reads the data from ROOT-files. Saves the module summary as a .ps and .png file. Also generates the address level fits as .ps and .png files.
- `addVcalThreshold`
  Adds a Vcal Threshold diagram for a single chip to the Vcal Threshold overview.
- `makePlot`
  Draws the subplots “Noise”,“Vcal Thr. Width”,“Rel. Gain Width”,“Pedestal Spread”,“Parameter1”.(See fitNames, nfit variables in moduleSummary())
- `grading`
  Calculates grading for a module.
- `qualification`
  Calculates the number of dead pixels, etc. needed for grading.
- `fitPeaks`
  Finds peaks for chips in subplots and calculates some statistical values.

The code is difficult to understand, since its structure is not obvious and it lacks documentation and comments.

Chip Summary Page The chip summary page in figure 7 is generated by the script `chipSummaryPage.C` Some important functions are listed below:

- `chipSummary`
  Generates an image file for the chip summary page. Generates the subplots using other functions in the file. Reads the data from ROOT-files. Saves the module summary as a .ps and .png file.
- `analyse`
  Generates plots like “ADC Measurement for ROC”, “ADC Calibration for ROC”, etc.
- `readCriteria`
  Reads data like “NOISE B”,“TRIMMING B”, etc. from a file.

The code is difficult to understand, since its structure is not obvious and it lacks documentation and comments.

4.4 Requirements

4.4.1 General

In general, MoRe-Web should be an easy-to-use interface for accessing information. It should be accessible by any client device and therefore be built on open web technologies. Furthermore, it should be optimized for display on mobile devices such as smart phones or table computers. All data should be stored in such a way that it can easily be accessed and processed by other software. It also is important that the code is easy to understand and maintain. Furthermore, the software should be extensible and flexible, in order to meet requirements that might come up in the future.

4.4.2 Overview

The MoRe-Web should provide an overview list of all tested modules. There might be several tests per module. By clicking a module, the module overview together with a list of the ROCs should be displayed, similar to the existing application (see figure 4).

4.4.3 Module Test Overview

The overview page should offer information similar to figure 5

4.4.4 Module Test Details

In general, the test details should be available separately and offline, that is even without connection to MoRe-Web. The test detail page displays several plots for the test, which should be zoomable. The information available should be similar to the one in figure 7.

4.5 Concept

The project is divided into two parts. First, the analysis scripts which are the connection point between the raw testing data and further processing. And second the web interface MoRe-Web.

4.5.1 Analysis Scripts

The analysis scripts should take data from ROOT-files and generate the following objects for both the module summary and the chip summary in the given directory structure:

- plots of test results
- HTML-file for summary pages which includes all plots and information and is accessible offline, that is it should be viewable directly from the file structure

Furthermore, the analysis scripts should insert the parsed data to a database system. For this particular case the use of sqlite is recommended, since no database server is needed and the database is directly stored in a file. Already existing entries will be replaced if the module numbers match. Since this database file is also available offline, it is used to generated an offline overview page.

4.5.2 MoRe-Web

The web interface is basically just a dynamic output of the overview page generated by the analysis scripts. The individual module tests consist of the offline HTML-files generated by the analysis scripts.

4.6 Implementation

The software can be found in the repository located at `https://svn.cern.ch/reps/moreweb` .

4.6.1 Analysis Scripts

The analysis scripts themselves are based on the following two abstract python classes, located in `AbstractClasses/`:

- `GeneralTestResult` This class is the base for the abstracted test classes for the module summary page, the chip summary page and the address levels.
  It provides functions for populating the data from raw result files, etc. For analysing a single module, a test result object is created out of the GeneralTestResult class, which in turn contains sub test result objects that again may store sub test result objects and so on. The hierarchy is shown in figure 8.

  The actual test result data is structured in the following way:

  – `Attributes`
    Hash table with attributes for the test, e.g. test date, tested object ID, module version, etc.

  – `ResultData`

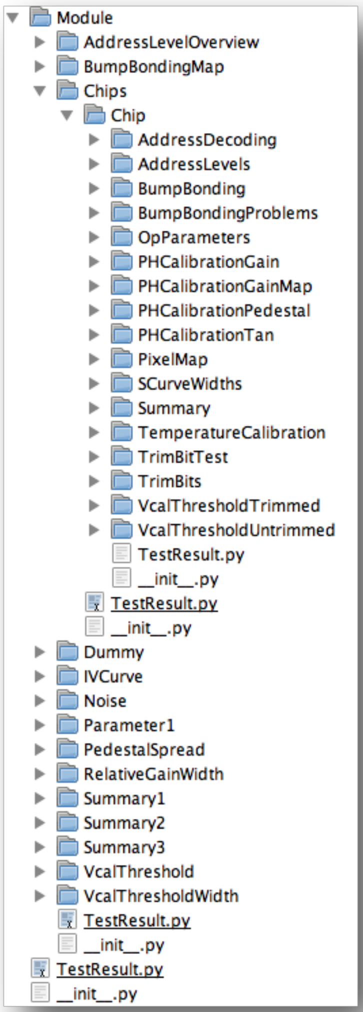
    ∗ `KeyValueDictPairs`
      Hash table that stores hash tables for unique keys containing a value, a unit and a label. For example a key can be “TestTemperature”, and the correspoding hash table stores -10 as the value, °C as the unit and “Test Temperature” as the label.

    ∗ `Plot`
      Hash table that stores a ROOT-Object used for the plot, a caption, a path to the generated image file and a file format.

    ∗ `SubTestResults`
      Hash table that stores sub test results, indexed by keys. For example, the module test result contains the IV-Curve as a sub test result.

    ∗ `Table`
      Hash table with three lists for header, body and footer data in a table.

  All the above data can be accessed by any test result object (therefore also by sub test results) since each test result object contains a reference to its parent test result object and the sub test results are indexed by a unique key.
  Some important functions implemented in the class are listed below:



```
Module
  AddressLevelOverview
  BumpBondingMap
  Chips
    Chip
      AddressDecoding
      AddressLevels
      BumpBonding
      BumpBondingProblems
      OpParameters
      PHCalibrationGain
      PHCalibrationGainMap
      PHCalibrationPedestal
      PHCalibrationTan
      PixelMap
      SCurveWidths
      Summary
      TemperatureCalibration
      TrimBitTest
      TrimBits
      VcalThresholdTrimmed
      VcalThresholdUntrimmed
      TestResult.py
      __init__.py
    TestResult.py
    __init__.py
  Dummy
  IVCurve
  Noise
  Parameter1
  PedestalSpread
  RelativeGainWidth
  Summary1
  Summary2
  Summary3
  VcalThreshold
  VcalThresholdWidth
  TestResult.py
  __init__.py
TestResult.py
__init__.py
```

Figure 8: Class Hierarchy for the inherited GeneralTestResult classes

– `PopulateAllData`
  This function invokes the data population for the rest results and all sub test results. It needs to be called for the topmost test result instance only.

– `CustomInit`
  Can be implemented for each test result class individually to set the name, some attributes and other variables before populating data.

– `GetUniqueID`
  Returns a unique id for a ROOT-object in order to prevent memory problems.

– `GetPlotFileName`
  Returns the full path to the corresponding plot file for the current test result.

– `PopulateResultData`
  Implements how the data is populated for the current test result. That is, it can populate attributes, generate a plot file, store key value pairs and a data table.

– `WriteToDatabase`

This function invokes the data upload to the database for the rest results and all sub test results. It needs to be called for the topmost test result instance only. Could be extended for storing all sub test results including all attributes, plots, key value pairs etc. (currently not needed, only module test result are written to database as a single row entry)

  - `CustomWriteToDatabase`
    Implements how the data is uploaded to the database for each test result.

  Moreover, it implements several functions for generating the HTML-output, explained in detail in section 4.6.2 For most test result classes, it is sufficient to implement the functions `CustomInit` and `PopulateResultData`.

- `TestResultEnvironment` This class provides general functions and variables that are needed by all test results, such as database connectivity and general parameters. The actual test result data is structured in the following way:

  - `GradingParameters`
    Hash table with grading parameters used to calculate the final grade of a module. The grading parameters are read form a configuration file.
  - `Configuration`
    Hash table containing several configuration parameters.

4.6.2 HTML-Output

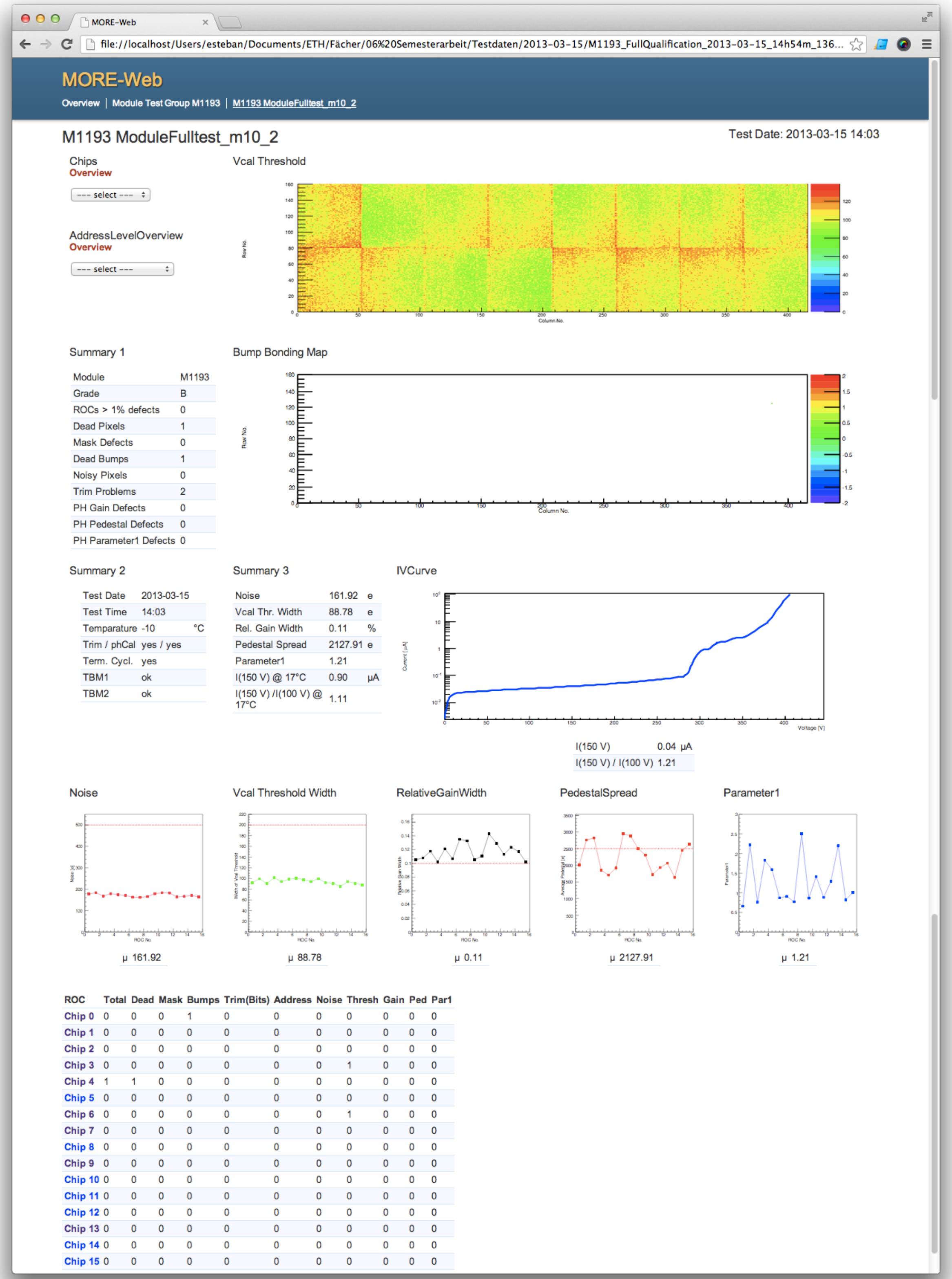
Test Result HTML File As mentioned before, every test result object based on the class `GeneralTestResult` creates an output HTML-file for directly opening the test result data, such as plots, values, etc. It contains a function called `GenerateDataFileHTML` which generates the output HTML file with the result data HTML obtained by calling `GenerateResultDataHTML` for the test result data and the sub test results. The functionality in `GenerateResultDataHTML` could also be directly implemented in `GenerateDataFileHTML`, but then it would not be possible to reuse it for sub test results that are displayed on the same page. For example, the module overview page contains a data table with all ROCs, but also several subtests, which are displayed on the same page. As of functionality, there are some key features worth mentioning:

- **Navigation** Since there are references to parent and sub test results in each test result instance, a relative navigation between the test results can be implemented without connection to a database at runtime.

- **Style** The styling of the HTML document is done using CSS. The layout adapts to different display sizes by dynamically adapting the number of columns shown to the screen width. Therefore it is also possible to browse the files on a mobile device.

- **Template** The HTML-files are generated using template files by replacing markers with actual content. Hence the layout can be changed without changing the code. For this, the templating class of the TYPO3-Project (`www.typo3.org`) was ported to python and can be found at `AbstractClasses/Helper/HTMLParser.py`

The specific test result HTML-file for a module fulltest is shown in figure 9. Figure 10 shows the same page, but with zoomed IV-Curve. An example for a chip test result HTML-file is given in figure 11.



M1193 ModuleFulltest_m10_2

Test Date: 2013-03-15 14:03

Summary 1

Module	M1193
Grade	B
ROCs > 1% defects	0
Dead Pixels	1
Mask Defects	0
Dead Bumps	1
Noisy Pixels	0
Trim Problems	2
PH Gain Defects	0
PH Pedestal Defects	0
PH Parameter1 Defects	0

Summary 2

Test Date	2013-03-15	
Test Time	14:03	
Temparature	-10	°C
Trim / phCal	yes / yes	
Term. Cycl.	yes	
TBM1	ok	
TBM2	ok	

Summary 3

Noise	161.92	e
Vcal Thr. Width	88.78	e
Rel. Gain Width	0.11	%
Pedestal Spread	2127.91	e
Parameter1	1.21	
I(150 V) @ 17°C	0.90	µA
I(150 V) /I(100 V) @ 17°C	1.11	

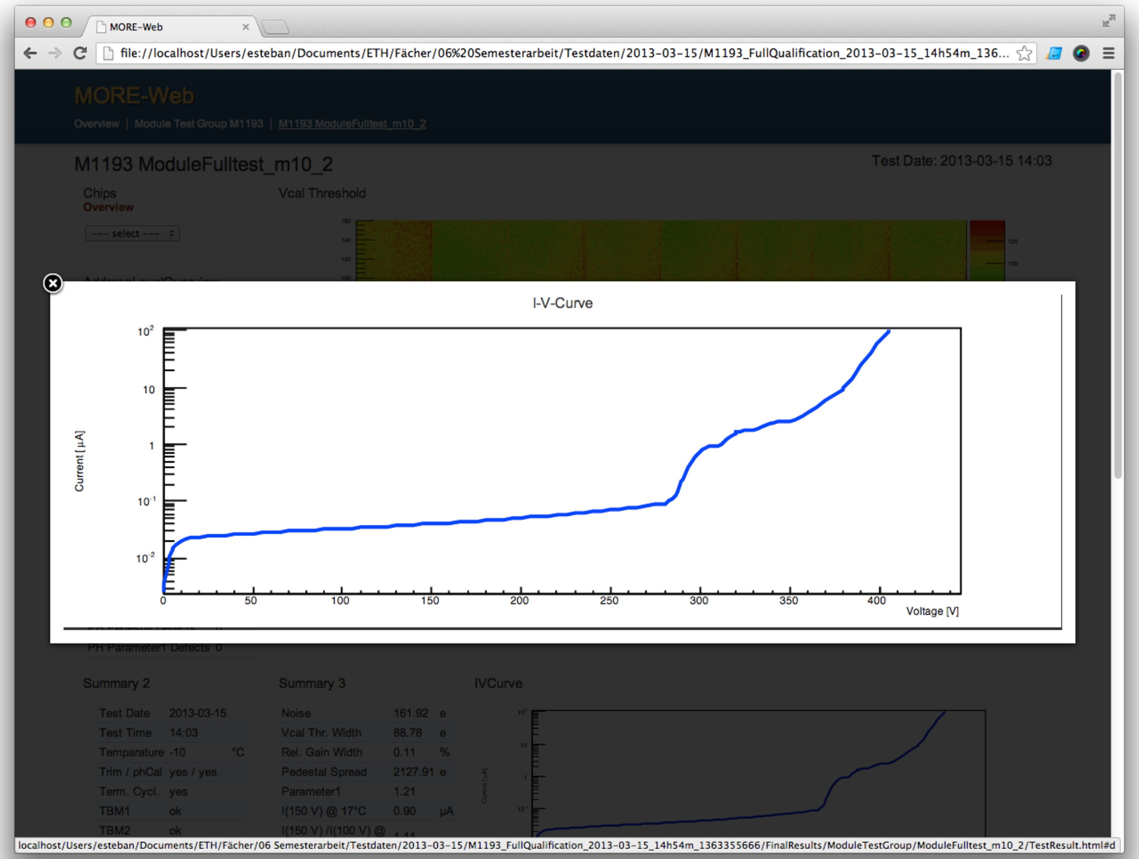
I(150 V)	0.04 µA
I(150 V) / I(100 V)	1.21

µ 161.92 µ 88.78 µ 0.11 µ 2127.91 µ 1.21

ROC	Total	Dead	Mask	Bumps	Trim(Bits)	Address	Noise	Thresh	Gain	Ped	Par1
Chip 0	0	0	0	1	0	0	0	0	0	0	0
Chip 1	0	0	0	0	0	0	0	0	0	0	0
Chip 2	0	0	0	0	0	0	0	0	0	0	0
Chip 3	0	0	0	0	0	0	0	1	0	0	0
Chip 4	1	1	0	0	0	0	0	0	0	0	0
Chip 5	0	0	0	0	0	0	0	0	0	0	0
Chip 6	0	0	0	0	0	0	0	1	0	0	0
Chip 7	0	0	0	0	0	0	0	0	0	0	0
Chip 8	0	0	0	0	0	0	0	0	0	0	0
Chip 9	0	0	0	0	0	0	0	0	0	0	0
Chip 10	0	0	0	0	0	0	0	0	0	0	0
Chip 11	0	0	0	0	0	0	0	0	0	0	0
Chip 12	0	0	0	0	0	0	0	0	0	0	0
Chip 13	0	0	0	0	0	0	0	0	0	0	0
Chip 14	0	0	0	0	0	0	0	0	0	0	0
Chip 15	0	0	0	0	0	0	0	0	0	0	0

Figure 9: Module Fulltest at $-10\,^{\circ}\mathrm{C}$

Overview HTML File Using the same styling, layout and templating as in the test result HTML-files, an overview page is generated using the class `ModuleResultOverview`, which reads the data from either the local database or the global database.


MORE-Web
file://localhost/Users/esteban/Documents/ETH/Fächer/06%20Semesterarbeit/Testdaten/2013-03-15/M1193_FullQualification_2013-03-15_14h54m_136...
MORE-Web
Overview | Module Test Group M1193 | M1193 ModuleFulltest_m10_2
Chips
Overview
--- select ---
Vcal Threshold
I-V-Curve
Current [µA]
Voltage [V]
10^2
10
1
10^{-1}
10^{-2}
0
50
100
150
200
250
300
350
400
IVCurve
localhost/Users/esteban/Documents/ETH/Fächer/06 Semesterarbeit/Testdaten/2013-03-15/M1193_FullQualification_2013-03-15_14h54m_1363355666/FinalResults/ModuleTestGroup/ModuleFulltest_m10_2/TestResult.html#d

M1193 ModuleFulltest_m10_2

Test Date: 2013-03-15 14:03

PH Parameter1 Defects 0

Summary 2

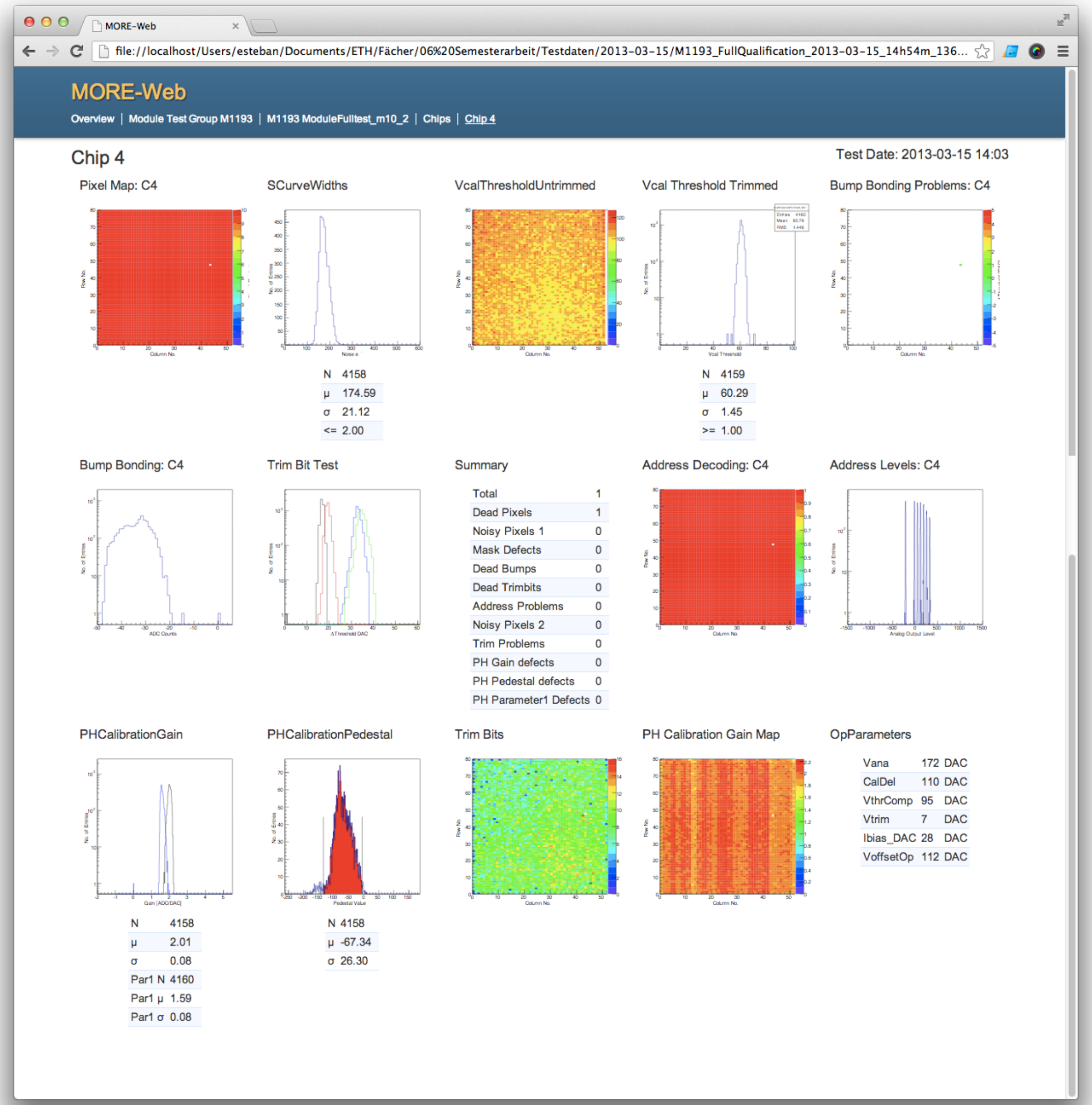
Test Date	2013-03-15	
Test Time	14:03	
Temparature	-10	°C
Trim / phCal	yes / yes	
Term. Cycl.	yes	
TBM1	ok	
TBM2	ok	

Summary 3

Noise	161.92	e
Vcal Thr. Width	88.78	e
Rel. Gain Width	0.11	%
Pedestal Spread	2127.91	e
Parameter1	1.21	
I(150 V) @ 17°C	0.90	µA
I(150 V) /I(100 V) @	1.11	

Figure 10: Module Fulltest at −10 °C with zoomed IV-Curve

An example is given in figure 12.


MORE-Web
file://localhost/Users/esteban/Documents/ETH/Fächer/06%20Semesterarbeit/Testdaten/2013-03-15/M1193_FullQualification_2013-03-15_14h54m_136...
Overview | Module Test Group M1193 | M1193 ModuleFulltest_m10_2 | Chips | Chip 4

Chip 4

Test Date: 2013-03-15 14:03

Pixel Map: C4

SCurveWidths

N	4158
μ	174.59
σ	21.12
<=	2.00

VcalThresholdUntrimmed

Vcal Threshold Trimmed

N	4159
μ	60.29
σ	1.45
>=	1.00

Bump Bonding Problems: C4

Bump Bonding: C4

Trim Bit Test

Summary

Total	1
Dead Pixels	1
Noisy Pixels 1	0
Mask Defects	0
Dead Bumps	0
Dead Trimbits	0
Address Problems	0
Noisy Pixels 2	0
Trim Problems	0
PH Gain defects	0
PH Pedestal defects	0
PH Parameter1 Defects	0

Address Decoding: C4

Address Levels: C4

PHCalibrationGain

N	4158
μ	2.01
σ	0.08
Par1 N	4160
Par1 μ	1.59
Par1 σ	0.08

PHCalibrationPedestal

N	4158
μ	-67.34
σ	26.30

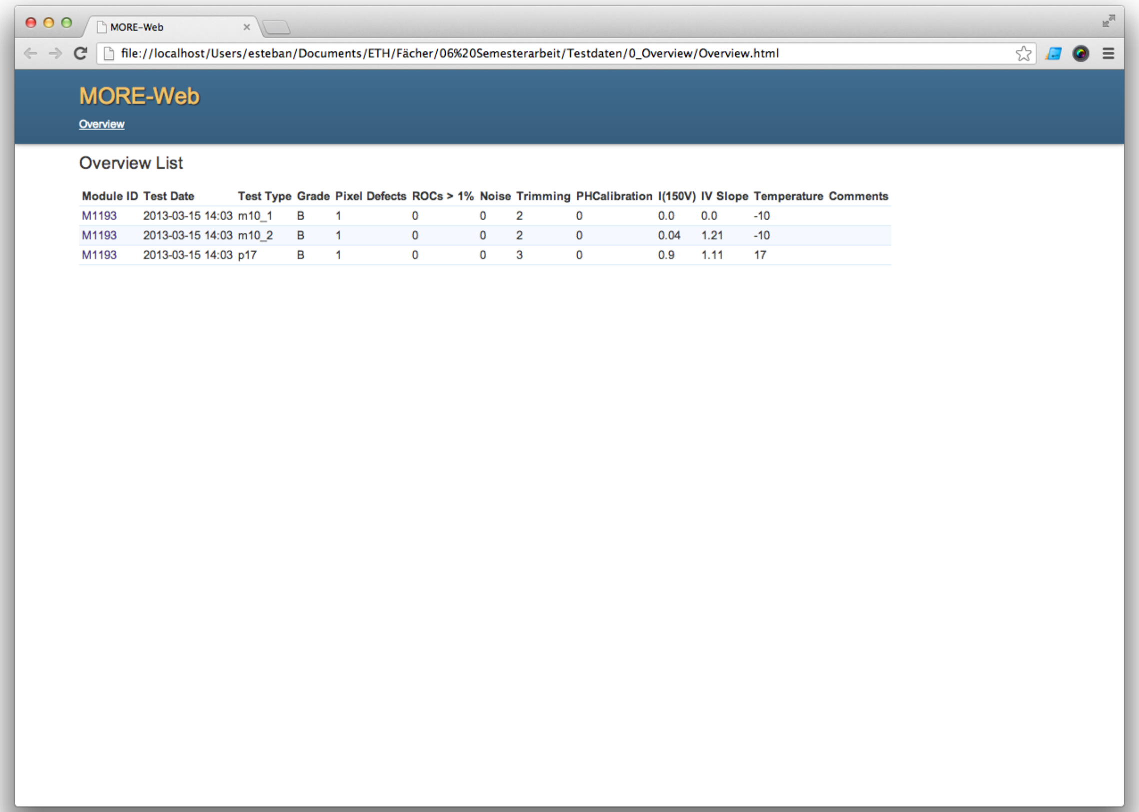
Trim Bits

PH Calibration Gain Map

OpParameters

Vana	172	DAC
CalDel	110	DAC
VthrComp	95	DAC
Vtrim	7	DAC
Ibias_DAC	28	DAC
VoffsetOp	112	DAC

Figure 11: Chip Test Results



MORE-Web

file://localhost/Users/esteban/Documents/ETH/Fächer/06%20Semesterarbeit/Testdaten/0_Overview/Overview.html

MORE-Web

Overview

Overview List

Module ID	Test Date	Test Type	Grade	Pixel Defects	ROCs > 1%	Noise	Trimming	PHCalibration	I(150V)	IV Slope	Temperature	Comments
M1193	2013-03-15 14:03	m10_1	B	1	0	0	2	0	0.0	0.0	-10	
M1193	2013-03-15 14:03	m10_2	B	1	0	0	2	0	0.04	1.21	-10	
M1193	2013-03-15 14:03	p17	B	1	0	0	3	0	0.9	1.11	17	

Figure 12: Overview page with list of module test results

4.7 Controller

A module qualification session is instantiated using the `Controller.py` program. It will instantiate the module test result object for all folders found in the `TestResultDirectory` variable defined in the configuration and all necessary sub test result objects from the classes found in `TestResultClasses/CMSPixel/ModuleTestGroup/`, provide it with a test result environment object containing data relevant for the qualification session. It also reads the configuration and sets all relevant paths.

4.7.1 System Requirements

The following requirements have to be met in order for the Controller.py to run correctly:

1. Python installed, minimum version: 2.6
2. Python packages:
   (a) ROOT (`http://root.cern.ch/drupal/content/pyroot`)

4.7.2 Step-By-Step Guide

Follow this step-by-step guide in order to perform module qualifications and generate the result output.

1. Check the minimum requirements as described in section 4.7.1
2. Open the command line on your UNIX system(Mac, Linux, etc.)
3. Change to the directory where the analysis scripts are located, i.e. where the `Controller.py` is stored, by typing `cd PathToAnalysisScripts`, where you replace `PathToAnalysisScripts` by the actual path to the analysis scripts.
4. Adjust the configuration to your current qualification session by editing the configuration files found in the `Configuration/` subfolder. The following lists describes the content of each configuration file:
   - **GradingParameters.cfg** Contains all the grading parameters. Usually needs no changes.
   - **ModuleInformation.cfg** Contains module information such as the module version, etc. for the type of modules analysed in the current qualification session.
   - **Paths.cfg** Stores the paths to the test result directory where all test results you want to qualify in this session are stored. Moreover, the path to the overview page is stored. This directory also contains the local `Overview.sqlite` sqlite database that stores all qualification results. The same database can be used for any number of qualification sessions, even when the test results in each session are stored in different folders, and therefore the resulting overview page contains all module qualification results.
   - **SystemConfiguration.cfg** Several system parameters such as the default image format, database connection parameters, etc. can be changed in this file.

5. Type the command `python Controller.py` to run the qualification session. For each module, it should display
   “Working on: ’TestDate’: ’1363355666’, ’TestType’: ’FullQualification’, ’ModuleID’: ’M1193’ –”. The current step is shown as either “Populating Data” or “Generating Final Output”. If other messages occur, it is recommended to analyse their meaning, as they are most probably indicating some problems that might have occured during the qualification process.

6. If no error messages appeared, all files should have been generated and the overview page can be found in the folder specified in the configuration. Each module test result folder now contains a folder called `FinalResults`, which you can browse to access the test result files of any sub test. **Important Notice:** The HTML-files can be opened with any web browser, but for best results, make sure you use an up-to-date version of one of the applications recommended in table 6. Please note that Internet Explorer lower than version 10 is unable to display SVG-graphics. If needed, the output file format can be changed to “png”, with the drawback of losing the scalability of vector graphics. Also, on some systems the gzip-compression of SVG-files might cause problems, in which case the configuration in **SystemConfiguration.cfg** can be changed to disable the compression.

Application	Minimum Version
Mozilla Firefox	4
Google Chrome	18
Safari	3
Opera	9
Internet Explorer	10

Table 1: Browser recommendations for opening the HTML-files

5 Conclusion

This project gives insight in several parts of physical research work. It is interesting what effort needs to be done in order to understand the tiniest particles in the universe using gigantic machines. What is most astonishing is the endurance that physicists show in order to gain a better understanding of nature.

The result of this Semesterarbeit is very satisfying. The software "MoRe-Web" is working properly and is already used in the productive module qualification. All requirements could be met and the users of the software can easily adapt the code to their needs.
The software was implemented in a way that allows its use also for other projects than the CMS pixel detector module qualification where plots and information of several objects have to be analysed and displayed.

5.1 Outlook

Since all important features have been implemented, there are currently no open tasks. However, the tests are expected to change and increase in number, since they have not all been adapted to the upgrade of the CMS pixel detector. Code changes should be possible without much effort, since the software was designed to be modular and extensible.

References

[1] W.N. Cottingham, D.A. Greenwood, *An Introduction to the Standard Model of Particle Physics*, 2nd edition, Cambridge University Press, 2007.

[2] CERN, *The search for the Higgs boson* [Online], 2013-06-19, `http://home.web.cern.ch/about/physics/search-higgs-boson`

[3] CERN, *About CERN* [Online], 2013-06-19, `http://home.web.cern.ch/about`

[4] CERN, *The Large Hadron Collider* [Online], 2013-06-19, `http://home.web.cern.ch/about/accelerators/large-hadron-collider`

[5] CERN, *CMS* [Online], 2013-06-19, `http://home.web.cern.ch/about/experiments/cms`

[6] CERN, *Sectional view of CMS* [Online Image], 2013-06-19, `https://cms-docdb.cern.ch/cgi-bin/PublicDocDB/RetrieveFile?docid=11514&version=1&filename=cms_120918_03.png`

[7] CERN, *CMS Tracker Detector* [Online], 2013-06-19, `http://cms.web.cern.ch/news/tracker-detector`

[8] CERN, *CMS Tracker Detector Upgrade* [Online], 2013-06-20, `http://cms.web.cern.ch/news/cms-prepares-pixel-and-hcal-upgrades`

[9] P. Trüb, *CMS Pixel Module Qualification and Monte-Carlo Study of H*, Ph.D. thesis, Diss. ETH No. 17985 (2008).

[10] CMS pixel group, *BPIX Module Testing* [Online], 2013-06-20, `http://cmspixel.phys.ethz.ch/moduleTests/moduleDB/prodTable.php`