



**DESY Summer Student Program 2012**  
**Automation of pixel detector module  
characterization**

Marco Sessa  
University of Bologna, Italy

Supervisors: Jan Hampe and Dr. Karsten Hansen  
FEC group

**Abstract**

The present pixel detector of the CMS experiment at CERN was designed for a maximum luminosity of  $1 \times 10^{34} \text{cm}^{-2} \text{s}^{-1}$ , a value that will be exceeded in LHC "Phase I", during which the instantaneous luminosity will reach the value of  $2 \times 10^{34} \text{cm}^{-2} \text{s}^{-1}$ . Consequently, the present system will not be able to sustain such extreme operating conditions, so an upgrade has been scheduled during the technical stop of 2016. During these 7 weeks, I worked in the electronics group of DESY and I focused my studies on the automation of dummy bare module tests and their characterization. These modules are used for evaluation of bump bonding technology that later will be used to connect the front-end readout chips to the silicon pixel sensors. I was involved in the development and debugging of two software programs, written respectively in MATLAB and LabVIEW language, to permit the automation of pixel detector module characterization.

## **Contents**

<b>1. CMS experiment</b>	<b>3</b>
<b>2. CMS pixel detector</b>	<b>5</b>
<b>3. Upgrade of the CMS pixel detector</b>	<b>7</b>
<b>4. Bump bonding technique</b>	<b>9</b>
<b>5. Software debugging</b>	<b>11</b>
5.1. MATLAB program . . . . .	11
5.2. LabVIEW program . . . . .	18
5.2.1. Scan list error . . . . .	21
5.2.2. Timeout error . . . . .	22
<b>6. Conclusions</b>	<b>24</b>
<b>7. Acknowledgments</b>	<b>24</b>
<b>A. MATLAB Code</b>	<b>25</b>

## 1. CMS experiment

CMS (Compact Muon Solenoid) is one of the four big experiments of CERN and its main goals are to explore physics at the TeV scale, the discovery of the Higgs boson and the research of physics evidences beyond the Standard Model, such as supersymmetry or extra dimensions.

CMS is a multipurpose detector (see figure 1); it contains subsystems which are designed to measure the energy and momentum of photons, electrons, muons and other products of the collisions. A particle emerging from the collision and travelling outwards will first encounter the tracking system, made of silicon pixels and silicon microstrip detectors. These accurately measure the positions of passing charged particles allowing to reconstruct their tracks (see figure 2).

The energy of the particles will be measured in the next layers of the detector, the so-called calorimeters. The first calorimeter layer is designed to measure the energy of electrons and photons with great precision. Since these particles interact electromagnetically, it is called electromagnetic calorimeter (ECAL).

Particles that interact by the strong force, hadrons, deposit most of their energy in the next layer, the hadron calorimeter (HCAL). The only particles able to penetrate beyond the HCAL are muons that will be detected by muon chambers.

Between hadron calorimeter and muon chambers is located the solenoid magnet which produces a high field of about 4 Tesla, about 100000 times stronger than that of the Earth, necessary for the particle momentum measurement [8].

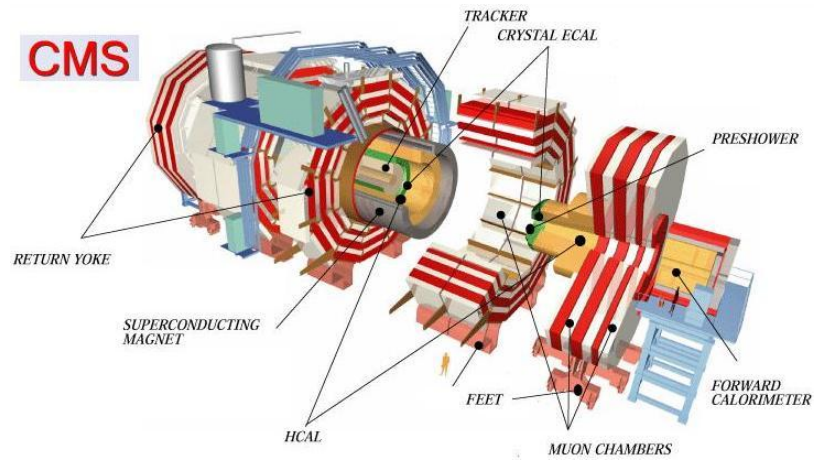


Figure 1: CMS detector.

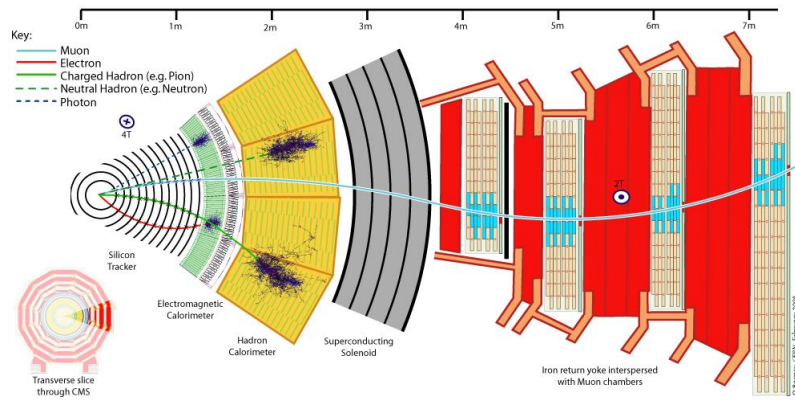


Figure 2: A slice of the CMS detector.

## 2. CMS pixel detector

The present CMS pixel detector consists of a central barrel and a pair of forward disks. The three 53 cm long barrel layers are located at mean radii of 4.3 cm, 7.2 cm and 11.0 cm. The two pairs of forward disks are placed 34.5 cm and 46.5 cm from the barrel center. The pixel detector is crucial for the detection of secondary vertices and plays an important role in the pattern recognition [3].

The detector has been assembled and tested at the Paul Scherrer Institute (PSI), Switzerland, in about two years, finishing at the beginning of 2008 [6]. The environment near the interaction point of LHC is characterized by an high flux of particles that leads to an extremely high radiation dose. Naturally, the efficiency and the resolution of the detector are expected to degrade slowly with irradiation [3].

The present barrel pixel detector is built of 768 segmented silicon sensor modules [6], mounted in three layers on a structure of thin-walled aluminum pipes and carbon fiber cross-pieces. Each of the cylindrical layers is formed by two half-shells.

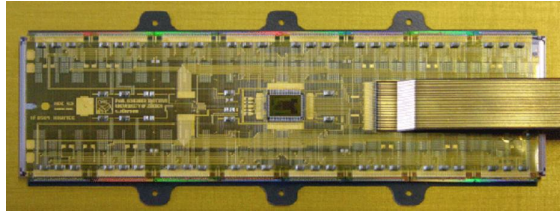


Figure 3: Module of CMS pixel detector [5].

There are two kinds of modules: full modules composed of 16 ROCs (Read Out Chips) and half modules built of 8 ROCs. In total, the barrel pixel detector contains 672 full and 96 half modules. The size of a single full module is  $66.6 \times 26 \text{ mm}^2$  and each ROC is segmented into  $52 \times 80$  pixels of size  $100 \times 150 \mu\text{m}^2$ , so that the total number of readout channels is about 48 million [6].

A completed full module weights 2.2 g plus up to 1.3 g for cables and consumes 2 W of power. The picture of a CMS barrel pixel detector module and its components are shown in the figure 3 and 4 respectively.

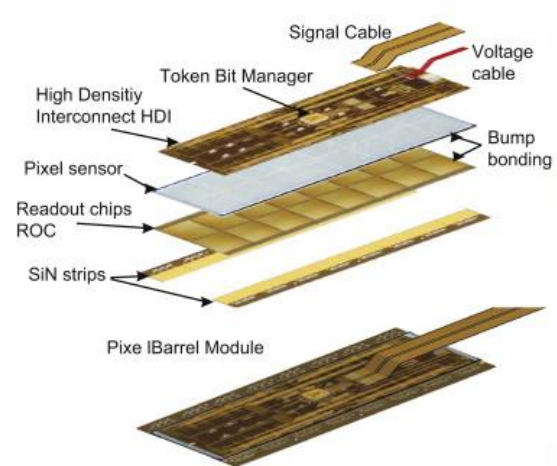


Figure 4: The components of a barrel pixel detector module [5].

### 3. Upgrade of the CMS pixel detector

The LHC 2012 run at a beam energy of 4 TeV has started, corresponding to a collision energy of 8 TeV, compared with the 7 TeV runs in 2010 and 2011. A technical stop is planned for 2016, before of the so called "Phase 1" run, which will start around 2017 and will continue past 2020. The expected center of mass energy will be 14 TeV and the peak luminosity will be  $2 \times 10^{34} \text{cm}^{-2} \text{s}^{-1}$ . With the increase of luminosity, the density of particles in the detector will be significantly greater. The number of particles that will cross the CMS pixel detector will be very huge, about 10 million of particles per square centimetre per seconds and the detector has to be able for at least six years to reconstruct all the particle tracks. The upgrade will replace the present pixel detector and will improve the CMS performance thanks to:

- a smaller beam pipe to improve impact parameter resolution and B-tagging;
- a fourth layer in the barrel detector and a third ring in the forward one for better track seeding efficiency and to improve the tracking;
- less passive material to reduce multiple scattering, photon conversion and nuclear interactions [5].

With the further layers, the present number of modules and pixels, respectively 768 and about 48 million, will be increased up to 1200 and about 80 million.

The new layout of the CMS pixel detector is shown in the figures 5 and 6.

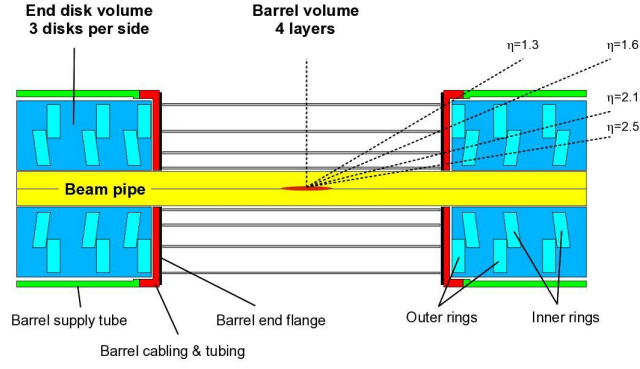


Figure 5: Schematic view of the four layers barrel pixel detector [7].

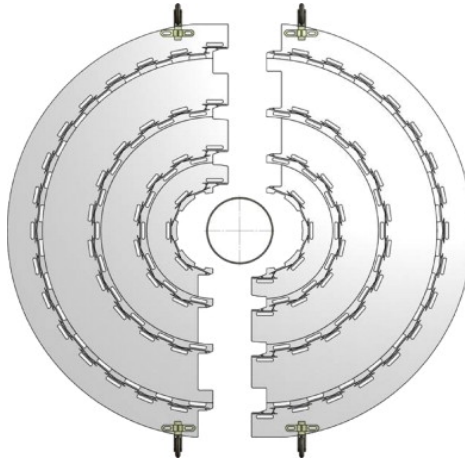


Figure 6: Drawing of the two barrel half-shells with four detector layers [3].

## 4. Bump bonding technique

During my work, I've tested dummy modules for CMS upgrade that were made by the PacTech German company using *SB<sup>2</sup>Jet* technique (see figure 7). This technique allows to place individually pre-formed balls over the sensor pads and re-flow solder them using a short laser pulse through a capillary [1]. This can be performed on full wafers or on individual sensors. The material used for solder bumping is a Sn-Ag-Cu alloy. The diameter of each solder ball is about  $40\mu m$  and the ball placement rate is 5 bumps/s.

The bump-bonding process can be divided into 3 steps:

- under-bump metal (UBM) deposition;
- solder sphere deposition;
- flip chip bonding with re-flow soldering followed by the bare-module electrical tests [1].

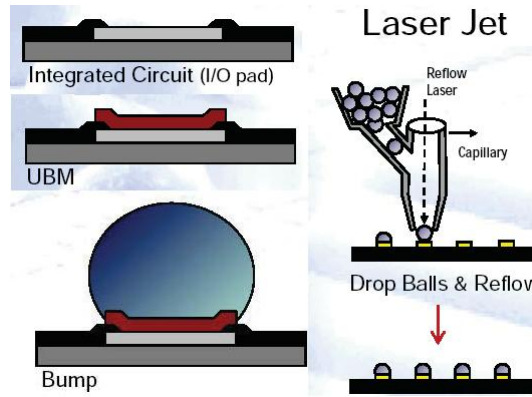


Figure 7: UBM and solder sphere deposition [5].

The final step, during which sensor chips are flip-chip bonded to the Read Out Chips, requires another machine with a precision better than  $3\mu m$ . This process is divided into two steps: flip-chip placement followed by re-flow soldering of the full module. Two fully equipped sensor-like structures were brought to DESY for ohmic testing of contact chains on a probe station. A probe card with 104 needles for 4-point contacting of 26 chains per ROC-like structure is used. A current is injected to a chain

of 160 bonds and the voltage drop measured. A single missing bump solder leads to a defective chain [1]. Only devices with less than 1% dead or noisy and without pixel masking defects in each chip are used in the following assembly steps [4].

The probe station and the dummy module, used during my work in the FEC laboratory, are shown in the figures 8 and 9, respectively.



Figure 8: Karl Suss PA300 Probe Station used for module characterization.

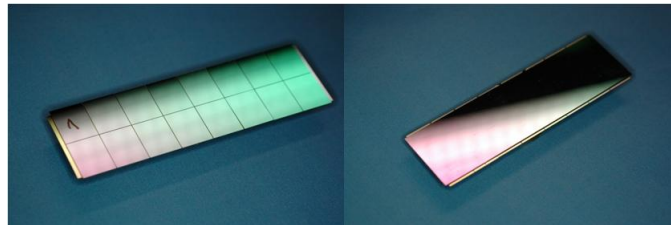


Figure 9: Top and bottom view of the CMS dummy module.

## 5. Software debugging

During my work in the electronics group (FEC) of DESY, I've been involved in the development and debugging of two programs, written respectively in MATLAB and LabVIEW language.

### 5.1. MATLAB program

The MATLAB program I've improved allows analysis of data acquired by 3706 KEITHLEY switch system/multimeter. Utilizing this program, I made an electrical characterization of two dummy modules each comprised of 16 ROCs and a sensor. The values of daisy chain resistances for one chip are shown in the following example output file.

```
LabVIEW Measurement
Writer_Version 2
Reader_Version 2
Separator Tab
Decimal_Separator .
Multi_Headings No
X_Columns No
Time_Pref Relative
Operator hampe
Date 2012/07/12
Time 10:16:12.26555
***End_of_Header***

Channels 1
Samples 9
Date 2012/07/12
Time 10:16:12.26555
X_Dimension Time
X0 0.000000000000000E+0
Delta_X 1.000000
***End_of_Header***
X_Value Untitled Comment
20.925791
20.964107
21.040000
21.185850
9.900000E+37
21.231814
20.984955
20.974770
20.961784
20.992110
20.932207
20.920063
20.885282
20.882481
9.900000E+37
21.179207
21.307538
21.184723
21.197463
21.242051
21.464587
23.361312
26.005269
28.750315
9.900000E+37
9.900000E+37
```

Each value corresponds to the resistance of a chain of the chip; it's easy to note the presence in this specific chip of four overflow values (that is four infinite resistance values). These values indicate that the respective

chains are open circuits. The reason could be a misalignment in theta, level and position, a missing solder bump or a defective pad, during the phase of assembly.

The values of chain resistances for all 32 analysed chip are shown in the figures 10 and 11.

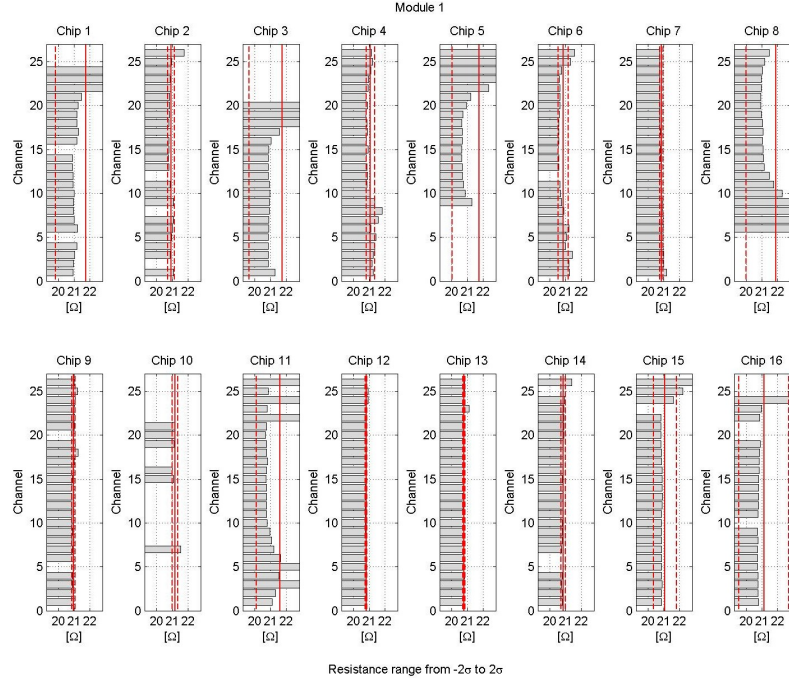


Figure 10: Graphical representation of module 1

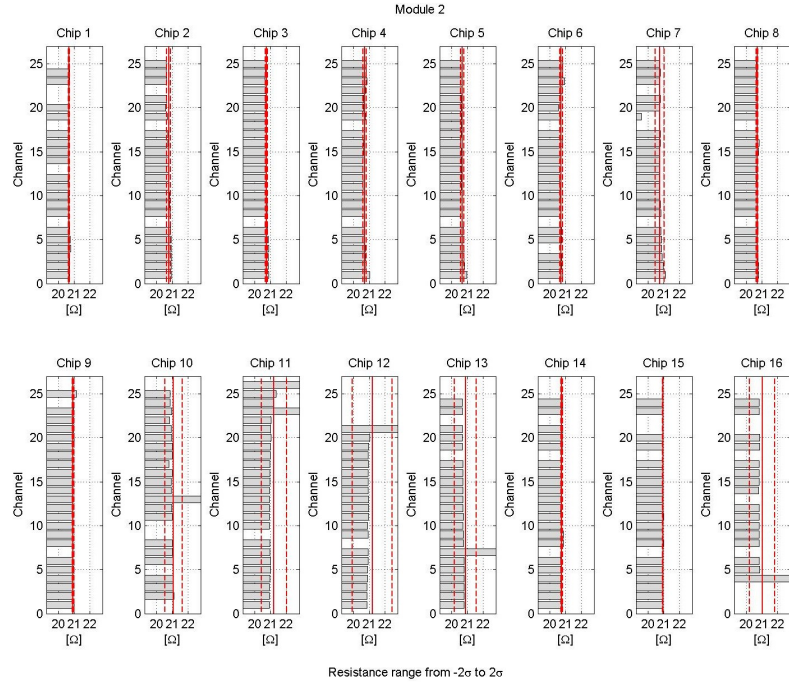


Figure 11: Graphical representation of module 2

The red dashed lines represent the negative and positive value of the first standard deviation of the specific chip. For better comparison, all x-axis cover the same range which is determined by the overall mean value and second standard deviation.

From these results it has been possible to reconstruct the histogram shown in the figure 12 and to determine the chip mean resistance, which value is  $21\Omega$ .

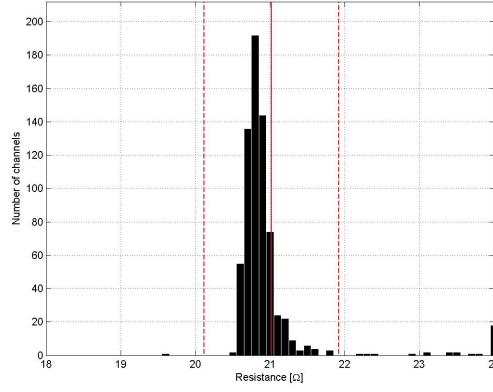


Figure 12: Chain resistance histogram

Moreover, it is possible to calculate the bump resistance mean value from the acquired data. To obtain it, the resistance of the interconnecting metal lines is subtracted (indeed it dominates the measurement: about  $18\Omega$  out of  $21\Omega$  per chain).

$$R_{bump} = \frac{R_{chain} - R_{AlSiCu}}{160}$$

In the figures 13 and 14, it is shown the bump resistance distribution and the mean bump resistance per chip respectively. The mean single bump resistance is about  $19\text{ m}\Omega$ .

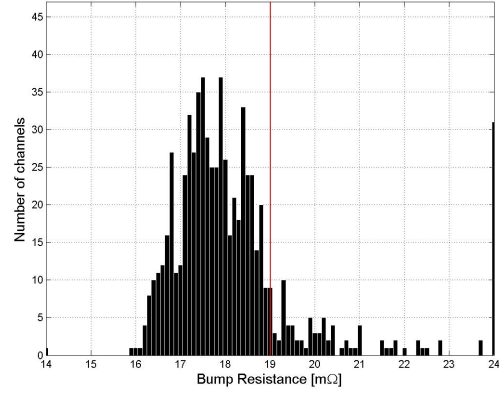


Figure 13: Bump resistance histogram.

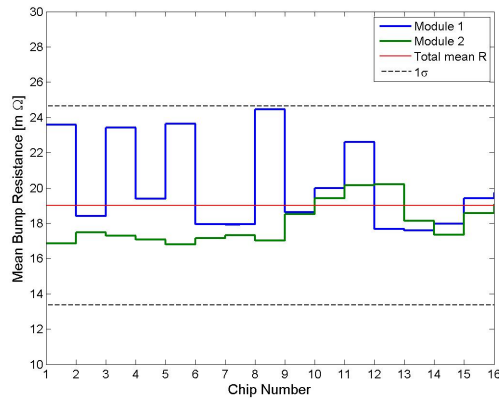


Figure 14: Mean bump resistance for each chip.

Figure 15 shows the number of defects for each chip and for both modules.

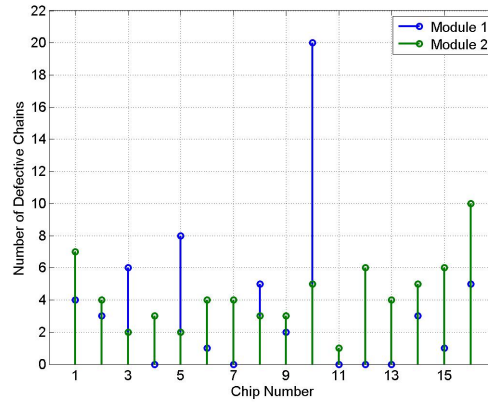


Figure 15: Defects per chip.

In the figure 16, it is shown the number of chips as function of number of defects per chip.

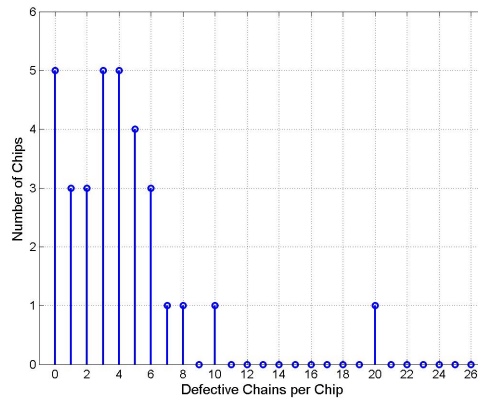


Figure 16: Number of chips with a certain number of defects.

This means that most of the chips have few defects and it is not common find a chip with a large number of error. We've found just one chip that presents 20 defective chains out of a total of 26.

Finally, the figure 17 shows the chains where is more probable to find a contact defect.

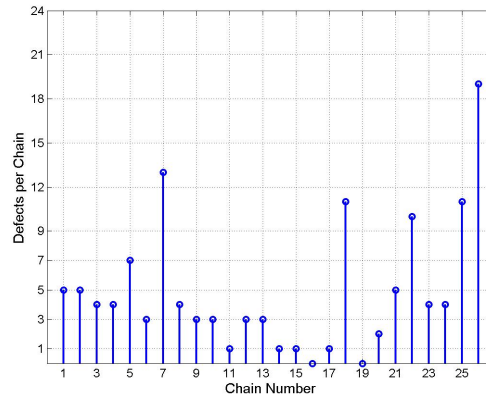


Figure 17: Number of defects for each chain

It's clear that most of the defects of production concern the left and right side of the chip. This is probably due to a misleveling during the flip chip placing.

## 5.2. LabVIEW program

The LabVIEW program allows the control and automation of the probe station (Suss Microtech PA 300 Probe Station, figure 19) for dummy module tests. It manages movement of the chuck for probing the ROC, data acquisition and storage. Automation is needed because CMS upgrade is coming and the number of module to test will be very high, so a chip per chip analysis will be impossible.

The developed program allows, once established the contact between the chip and the probe, the measurement of chain resistances. After the first measurement, the chuck go down and the probe separates from the chip. The chuck moves now towards the second chip, establishes a new contact, starts the measurement and repeats these steps until the last chip. When the sequence of measurements is finished, the chuck brings the module in the "unload position", from which it is possible to extract the chip from the probe station.

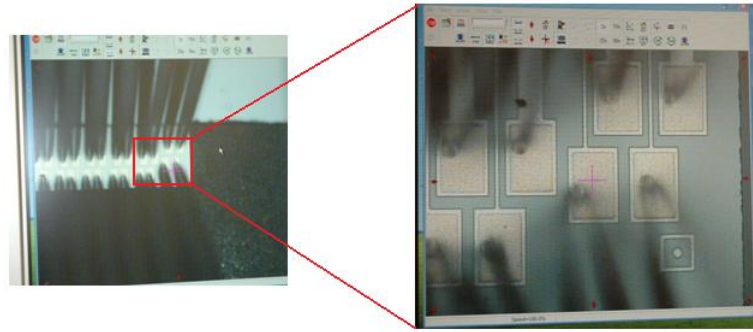


Figure 18: View of probe pins and contact pads.

It is possible make resistance measurements on all chips of a row, but, unfortunately, not in the other one because of asymmetry of the contact pads.

In the figure 19, it is shown the front panel of the LabVIEW program. In this Graphical User Interface we have the possibility to set some parameters, needed for the measurement. We can set:

- the VISA Resource Name, that identifies the Switch system/multimeter in the network;
- the kind of measurement to execute;
- the number of daisy chains to analyze;
- the number of times that we want to execute the measurements on the same chip;
- the name of the output files;
- the velocity percent of the motion of the chuck;
- the Number of Power Line Cycles ;
- the number of chips to analyze;
- the timeout value.

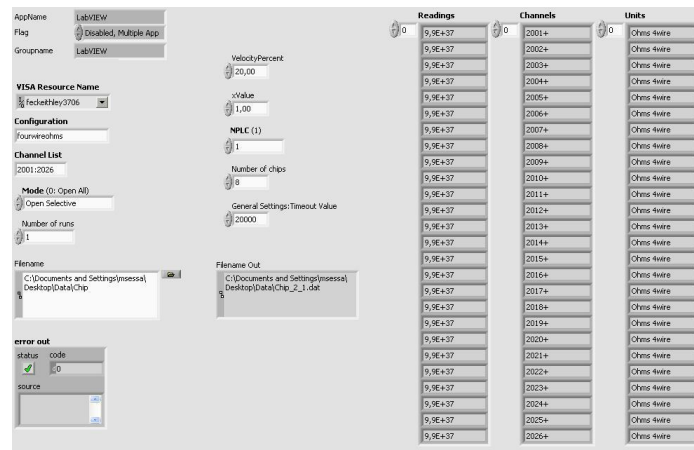


Figure 19: LabVIEW program front panel.

In the figure 20, it is shown the block diagram of the LabVIEW program. Here we can see the Karl Suss PA300 Probe Station drivers (orange icons) that allow the motion of the chuck for the analysis of each chip.

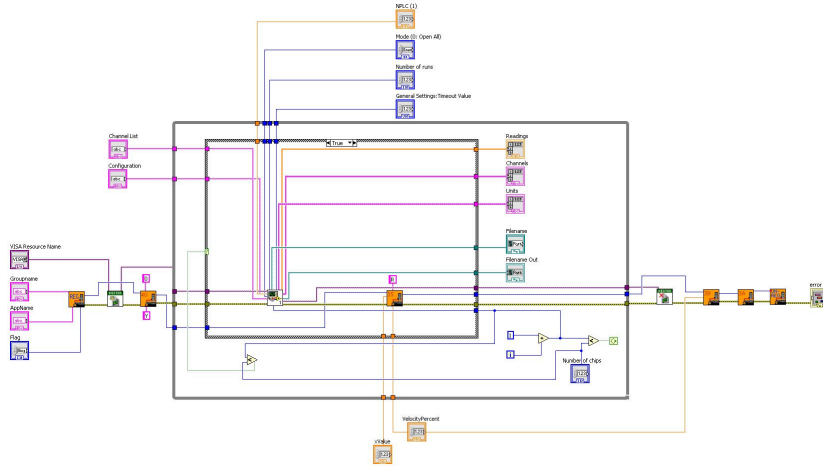


Figure 20: LabVIEW program block diagram.

To reduce the complexity of the program, part of it was inserted into a subprogram, called "SubVI", that it is illustrated in the figure 21. The SubVI contains the 3706 KEITHLEY switch system/multimeter drivers that allow the measurement of daisy chain resistances in each chip.

During my work, I removed two bugs in the existing LabVIEW program, changing the configuration of the 3706 KEITHLEY switch system/multimeter drivers.

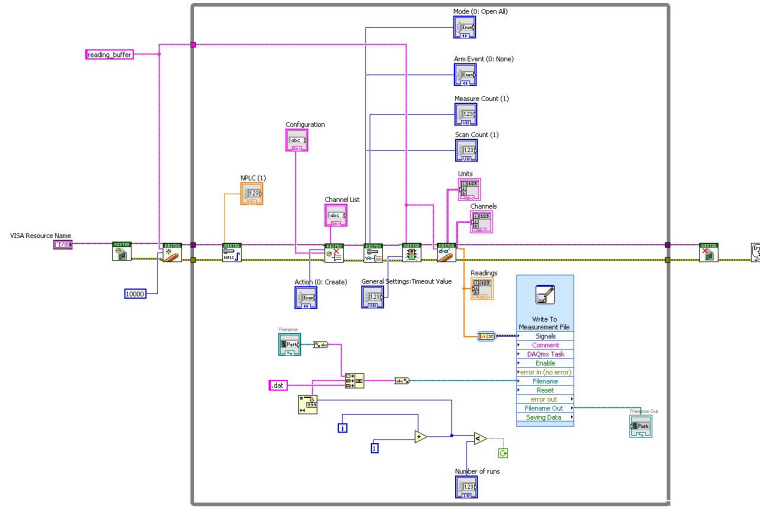


Figure 21: LabVIEW SubVI.

### 5.2.1. Scan list error

This first problem concerns the Read Measurement Buffer driver. Per chip, 26 channels are to be analyzed but, by the default driver configuration, 26 is interpreted like 2 on the laboratory computer. This is due to a conflict in the representation of real numbers in the operating system with LabVIEW. It is possible to switch a flag in the driver between true and false to change the decimal separator between comma and point. If true, that is the default, the decimal separator is the comma and this is shown in the figure 22. If we switch to false in place of true, the decimal separator is the point and in this case the Read Measurement Buffer driver works correctly.

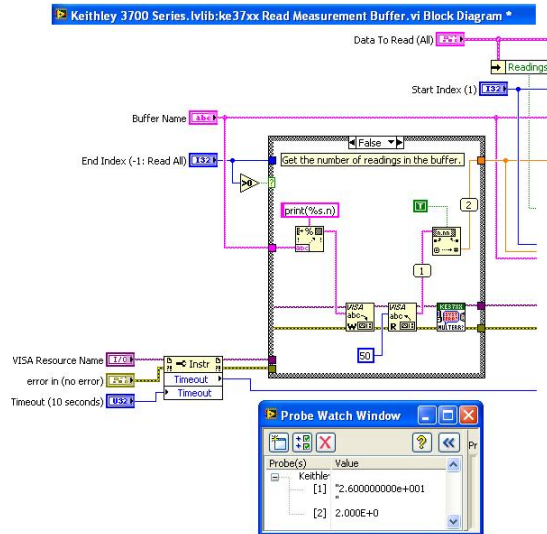


Figure 22: Read Measurement Buffer.vi Block Diagram.

### 5.2.2. Timeout error

This kind of error, shown in the figure 23, can occur when we want readout large amounts of data from the buffer and save them in the LabVIEW computer directory.

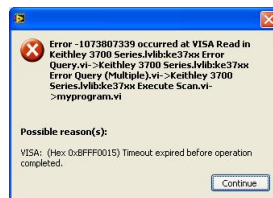


Figure 23: Timeout error message.

The communication time through the Ethernet connection is longer than default timeout. So I've included a further timeout function (see figure 24) and, in this way, the program waits for a period 10 times longer than default one.

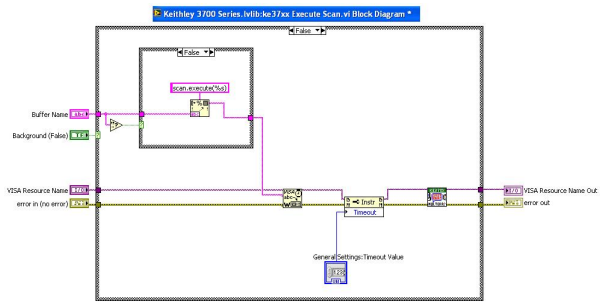


Figure 24: Execute Scan.vi Block Diagram.

## 6. Conclusions

My work in the FEC group of DESY concerned the improving and debugging of two programs, employed for CMS dummy module characterization. This kind of characterization is important for testing the quality of  $SB^2Jet$  technique.

The first program, written in MATLAB language, allows the daisy chain resistance data analysis and it has been improved such that the statistical analysis of any number of modules is possible. The MATLAB code provides an automated data import from a specific folder which is selected by user query as well as automated storage of the produced figures and plots in JPEG and EPS format.

The LabVIEW program allows the control of probe station chuck and the probing of each chip of a single row of dummy module. The daisy chains of the chips are automatically measured and the results stored. During my work, I've learned how to localize a bug inside the program and drivers and how to correct it. The LabVIEW program now works correctly and any other kind of error has been observed.

## 7. Acknowledgments

I would first like to thank my supervisor Jan Hampe for his availability, his patience and his good company. Thank you Jan! Of course, thanks to Karsten Hansen and to the whole team of FEC.

Thanks to Olaf Behnke, Doris Eckstein and Andrea Schrader for organizing the Summer Student Program and giving me the opportunity to spend these awesome two months in DESY and Hamburg.

Finally, thanks a lot to all Summer Students and in particular to all Italian guys for your company and for the time spent together.

## A. MATLAB Code

```
% Section 1
clear all, close all;

% Input data
datapath = input('Type the datapath: ');
modules = input('Type the number of modules: ');
touchdowns = input('Type the number of chip per module: ');

ModulePath='Modul ';

B = zeros(26,touchdowns*modules);
for i = 1:modules

    Path = [datapath, '\', ModulePath, num2str(i), '\'];

    % Import data and store it in a matrix. Each chip is represented by a
    % column.
    for k=1:touchdowns
        A = importdata(strcat(Path, 'Chip', num2str(k), '.dat'), ' ', 22);
        B(:,(i-1)*touchdowns + k) = A.data;
    end

end

% L is the matrix with value "0" where B has value above maximum (here 1000)
% and value "1" where B has value below maximum.
L=B(:,:)<1000;

% A_exc is the matrix of the data in which the infinite resistances
% (i.e. here > 1000 Ohms) becomes equal to 0.
A_exc=L.*B;

% A_sum is the column vector in which each element represents the sum of
% the resistances per chip (below 1000 ohm).
A_sum=diag(L'*B);

% T is the row vector in which each element represents the number of the
% operating channels
T=sum(L);

% Mean value per chip.
M_chip=A_sum'./T;

% Mean value per module.
for n=1:modules

    M_module(n)=sum(A_sum(1+(n-1)*touchdowns:n*touchdowns,1))/...
        sum(T(1,1+(n-1)*touchdowns:n*touchdowns));

end
```

```

% Mean value per all modules.
M_tot=sum(A_sum)/sum(T);

%%
% Section 2
% Calculation of the standard deviation.
% The infinite resistances have been excluded and do not disturb the result.

% Building the differences with the mean value.
for k = 1:(modules*touchdowns)

    V1(:,k)=L(:,k) * M_chip(k);
    V2(:,k)=L(:,k) * M_tot;

    dif_exc(:,k)=A_exc(:,k)-V1(:,k);
    dif_exc_tot(:,k)=A_exc(:,k)-V2(:,k);
end

dif_exc=dif_exc.^2;
dif_exc_tot=dif_exc_tot.^2;

% There are two definitions of the standard deviation.

% Definition 1.
%S = sqrt(sum(dif_exc) ./ (T));

% Definition 2 (used in Matlab by default).
S=sqrt(sum(dif_exc) ./ (T-1));
S_tot=sqrt(sum(sum(dif_exc_tot)) ./ (sum(T)-1));

% To plot the standard deviations into the bar graphs, the absolut values
% on y-axes are calculated.
S1P = M_chip + S;
S1N = M_chip - S;

S1P_tot = M_tot + S_tot;
S1N_tot = M_tot - S_tot;

S2P_tot = M_tot + 2*S_tot;
S2N_tot = M_tot - 2*S_tot;

S3P_tot = M_tot + 3*S_tot;
S3N_tot = M_tot - 3*S_tot;

% Resistance in Ohm per Daisy Chain due to AlSiCu connections neglecting
% the bump pads.
AlSiCu_path = 17.978;
R_bump = (M_tot - AlSiCu_path) / 160;

% Number of measured chains
finite_measures = sum(T);

% Number of open chains
infinite_measures = (modules*touchdowns*26) - finite_measures;

```

```

% Vector of mean value per chip
R_bump_chip = (M_chip - AlSiCu_path) ./ 160;

Measured_chains_in_percent = finite_measures / (modules*touchdowns*26)*100;
Measured_chains_in_percent_per_chip = (T ./ 26) * 100;

%%
% Section 3: plotting the resistance per bump chip wize.

% Coloured figures.

for i=1:modules

    figure;

    for k=1:touchdowns

        subplot(2,8,k)
        barh(A_exc(:,k+touchdowns*(i-1)), 'FaceColor', [0.85,0.85,0.85]);
        hold on;

        % axes labels
        title(['Chip ', num2str(k)])
        ylabel('Channel')
        xlabel(['\Omega'])
        axis([S2N_tot S2P_tot 0 27])
        grid on

        % Mean value line
        plot([M_chip(k+touchdowns*(i-1)) M_chip(k+touchdowns*(i-1))],...
            [0 27], 'Color', 'r', 'Linewidth', 1)

        % Standard deviation lines
        plot([S1N(k+touchdowns*(i-1)) S1N(k+touchdowns*(i-1))],[0 27],...
            'Linewidth',1,'Color','r','LineStyle','--')
        plot([S1P(k+touchdowns*(i-1)) S1P(k+touchdowns*(i-1))],[0 27],...
            'Linewidth',1,'Color','r','LineStyle','--')

    end

    %figure dimensions
    set(gcf,'PaperType','A4','PaperUnits','normalized',...
        'PaperOrientation','landscape','PaperPosition', [0 0 1 1])

    %placing the title
    text(-2.6, 68.7, ['Module ', num2str(i)]);
    text(-6.9, -6.6, 'Resistance range from -2\sigma to 2\sigma');

    print('-djpeg', ['Module_', num2str(i), '_c']);
    print('-depsc2', ['Module_', num2str(i), '_c']);
    hold off

    mkdir('Visualized module')
    movefile(['Module_', num2str(i), '_c.jpg'], 'Visualized module');
    movefile(['Module_', num2str(i), '_c.eps'], 'Visualized module');

```

```

end

% Black and white figures.

for i=1: modules

    figure;

    for k=1:touchdowns

        subplot(2,8,k)

        barh(A_exc(:,k+touchdowns*(i-1)), 'FaceColor', [0.85,0.85,0.85]);

        hold on;

        % axes labels
        title(['Chip ', num2str(k)])
        ylabel('Channel')
        xlabel('Resistance [\Omega]')
        axis([S2N_tot S2P_tot 0 27])
        grid on

        % Mean value line
        plot([M_chip(k+touchdowns*(i-1)) M_chip(k+touchdowns*(i-1))],...
            [0 26], 'Color', 'k', 'Linewidth', 1)

        % Standard deviation lines
        plot([S1N(k+touchdowns*(i-1)) S1N(k+touchdowns*(i-1))], [0 26],...
            'Linewidth',1,'Color','k','LineStyle','--')
        plot([S1P(k+touchdowns*(i-1)) S1P(k+touchdowns*(i-1))], [0 26],...
            'Linewidth',1,'Color','k','LineStyle','--')

    end

    set(gcf, 'PaperType', 'A4', 'PaperUnits', 'normalized', ...
        'PaperOrientation', 'landscape', 'PaperPosition', [0 0 1 1])

    text(-2.6, 68.7, ['Module ', num2str(i)]);
    text(-6.9, -6.6, 'Resistance range from -2\sigma to 2\sigma');

    print('-djpeg', ['Module_', num2str(i), '_bw']);
    print('-depsc2', ['Module_', num2str(i), '_bw']);
    hold off

    mkdir('Visualized module')
    movefile(['Module_', num2str(i), '_bw.jpg'], 'Visualized module');
    movefile(['Module_', num2str(i), '_bw.eps'], 'Visualized module');
    close;

end

```

```

%%
% Section 4: histogram of the resistances.

% x axis range goes from -3 sigma to 3 sigma.
limitx_pos = ceil(S3P_tot);
limitx_neg = floor(S3N_tot);

% Create the histogram and count the entries for each bin
% (the infinite valuse have been excluded).
hist(A_exc(A_exc > 0), limitx_neg:0.1:limitx_pos);
limit_y = max(hist(A_exc(A_exc > 0), limitx_neg:0.1:limitx_pos))+ 20;

grid on
hold on

% Mean value line.
plot([M_tot M_tot],[0 limit_y], 'Color', 'r', 'Linewidth', 1)

% Standard deviation lines.
plot([S1N_tot S1N_tot],[0 limit_y], 'Linewidth', 1,'Color', 'r',...
'LineStyle', '--')
plot([S1P_tot S1P_tot],[0 limit_y], 'Linewidth', 1,'Color', 'r',...
'LineStyle', '--')

% Axes labeling.
xlabel ('Resistance [ $\Omega$ '])
ylabel ('Number of channels')

% modifying the histogram appearence
% i.e. bars with white edge and black color instead of standard blue
h=findobj(gca, 'Type', 'patch');
set(h,'FaceColor','k','EdgeColor','w');

axis([limitx_neg limitx_pos 0 limit_y]);

hold off;

print('-depsc2', 'histogram_chain');
print('-djpeg', 'histogram_chain');

mkdir('Histograms');
movefile('histogram_chain.jpg', 'Histograms');
movefile('histogram_chain.eps', 'Histograms');

%%
% Section 5: module vision

for i=1:modules;

    figure;

```

```

z = 1;
subplot(1, 2, 1);
hold on;

% First row of chips
for chip = 1+touchdowns*(i-1):8+touchdowns*(i-1);

    % Take all channels per chip
    for channel = 1:26;

        % Colour selection

        % Within the 1. standard deviation: green.
        if (A_exc(channel, chip) < S1P_tot && A_exc(channel, chip) >...
            S1N_tot && A_exc(channel, chip) > 0)
            color = 'g';

        % Within the 2. standard deviation: yellow.
        elseif (A_exc(channel, chip) < S2P_tot && A_exc(channel, chip)...
            > S2N_tot && A_exc(channel, chip) > 0)
            color = 'y';

        % Within the 3. standard deviation: red.
        elseif (A_exc(channel, chip) < S3P_tot && A_exc(channel, chip)...
            > S3N_tot && A_exc(channel, chip) > 0)
            color = 'r';

        % Outside the 3. standard deviation: black.
        elseif (A_exc(channel, chip) > S3P_tot || A_exc(channel, chip)...
            < S3N_tot || A_exc(channel, chip) == 0)
            color = 'k';
        end

        % Plot a rectangle per channel and fill it with the appropriate color.
        fill([0 0.5 0.5 0],[0 0 1 1]-(z-2), color)
        z= z+1;

    end

    fill([0 0.5 0.5 0],[0 0 2 2]-(z-1), 'w')
    z=z+2;

end

ylim([-220 2]);

% axes labels:
set(gca, 'ytick', [-191; -167; -138; -109; -80; -52; -24; -2],...
    'YTickLabel',{'8'; '7'; '6'; '5'; '4'; '3'; '2'; '1'})
set(gca, 'xtick', []);

% Second row of chips
z = 1;

```

```

subplot(1, 2, 2);
hold on;

for chip = 9+touchdowns*(i-1):touchdowns+touchdowns*(i-1);

    % Take all channels per chip
    for channel = 1:26;

        % Colour selection

        % Within the 1. standard deviation: green.
        if (A_exc(channel, chip) < S1P_tot && A_exc(channel, chip) >...
            S1N_tot && A_exc(channel, chip) > 0)
            color = 'g';

        % Within the 2. standard deviation: yellow.
        elseif (A_exc(channel, chip) < S2P_tot && A_exc(channel, chip)...
            > S2N_tot && A_exc(channel, chip) > 0)
            color = 'y';

        % Within the 3. standard deviation: red.
        elseif (A_exc(channel, chip) < S3P_tot && A_exc(channel, chip)...
            > S3N_tot && A_exc(channel, chip) > 0)
            color = 'r';

        % Outside the 3. standard deviation: black.
        elseif (A_exc(channel, chip) > S3P_tot || A_exc(channel, chip)...
            < S3N_tot || A_exc(channel, chip) == 0)
            color = 'k';

        end

        % Plot a rectangle per channel and fill it with the appropriate color.
        fill([0 0.5 0.5 0],[0 0 1 1]-(z-2), color)
        z= z+1;

    end

    fill([0 0.5 0.5 0],[0 0 2 2]-(z-1), 'w')
    z= z+2;

end

ylim([-220 2]);

% axes labels:
set(gca,'ytick',[-191; -167; -138; -109; -80; -52; -24; -2],...
'YTickLabel',{'16'; '15'; '14'; '13'; '12'; '11'; '10'; '9'});
set(gca,'xtick',[]);

```

```

% Speichereinstellungen
set(gcf,'PaperType','A4','PaperUnits','normalized','PaperOrientation',...
'landscape','PaperPosition',[0 0 1 1])

%text(-5, -4, ['Module ', num2str(i)]);

print('-djpeg', ['Module_vision_', num2str(i)]);
print('-depsc2', ['Module_vision_', num2str(i)]);
hold off

mkdir('Visualized module')
movefile(['Module_vision_', num2str(i), '.jpg'], 'Visualized module');
movefile(['Module_vision_', num2str(i), '.eps'], 'Visualized module');
end

%%
% Section 6: conversion from chain resistance into bump resistance.

% AlSiCu_path is the variable for resistance in Ohm per Daisy Chain due to
% AlSiCu connections neglecting the bump pads.

% Total mean value in mOhm from all measures
% 160 is the number of bumps per chain
R_bump = (M_tot - AlSiCu_path) / 0.160;

% Matrix of bump resistance per chain (in mOhm)
R_chain_bump = (A_exc - (L*AlSiCu_path)) ./ 0.160;

% Mean value of bump resistance per chip
R_bump_per_chip = sum(R_chain_bump) ./ T;

for k = 1:(modules*touchdowns)
    G1(:,k)=L(:,k)*R_bump_per_chip(k);
    G2(:,k)=L(:,k)* R_bump;

    dif_exc(:,k)=R_chain_bump(:,k) - G1(:,k);
    dif_tot_exc(:,k)=R_chain_bump(:,k)-G2(:,k);
end

% Squared
dif_exc = dif_exc.^2;
dif_tot_exc = dif_tot_exc.^2;

% There are two definitions of the standard deviation.
% Definition 1
%S = sqrt(sum(dif_exc) ./ (T));

% Definition 2, which is used in Matlab by default
S = sqrt(sum(dif_exc) ./ (T-1));
S_tot = sqrt(sum(sum(dif_tot_exc)) ./ (sum(T)-1));

% To plot the standard deviations into the bar graphs the absolut values
% on y-axes are calculated

```

```

S1P = R_bump_per_chip + S;
S1N = R_bump_per_chip - S;

S1P_tot = R_bump + S_tot;
S1N_tot = R_bump - S_tot;

S2P_tot = R_bump + 2*S_tot;
S2N_tot = R_bump - 2*S_tot;

S3P_tot = R_bump + 3*S_tot;
S3N_tot = R_bump - 3*S_tot;

%%
% Section 7: histogram of bump resistances.

% x axes limit: from -1 sigma to 1 sigma.
limitx_neg = ceil(S1N_tot);
limitx_pos = floor(S1P_tot);

% Create the histogram
hist(R_chain_bump(R_chain_bump > 0),limitx_neg:0.1:limitx_pos);

% y axes upper limit
limit_y = max(hist(R_chain_bump(R_chain_bump > 0),limitx_neg:0.1:limitx_pos))+ 10;

grid on
hold on

axis([limitx_neg limitx_pos 0 limit_y])

% Mean value line
plot([R_bump R_bump],[0 limit_y], 'Color', 'r', 'Linewidth', 1)

% axes labeling
xlabel ('Bump Resistance [m\Omega]', 'FontSize', 14);
ylabel ('Number of channels', 'FontSize', 14);

% Blue bins and white edges
h = findobj(gca, 'Type', 'patch');
set(h, 'FaceColor', 'k', 'EdgeColor', 'w')
hold off;

print ('-depsc2', 'histogram_bump');
print ('-djpeg', 'histogram_bump');

mkdir('Histograms');
movefile('histogram_bump.jpg', 'Histograms');
movefile('histogram_bump.eps', 'Histograms');

%%
% Section 8: defective chains per chip

% Number of defect per chip
defect=ones(1,modules*touchdowns)*26 - T;

%Counting how many chips have the same number of defects

```

```

defect_count=zeros(2,27);
defect_count(1,:) = [0:1:26];

%Vector with number of chips with certain number of defects
for k = 1:(modules*touchdowns)

    defect_count(2,defect(k)+1) = defect_count(2,defect(k)+1) + 1;

end;

% Create figure 1
defects_per_chip=figure;

% Create axes
axes1=axes('Parent',defects_per_chip, 'XTick',1:2*touchdowns, 'YGrid',...
'on', 'XGrid','on','FontSize',14);

% x axes limit
xlim(axes1,[0 17]);

% y axes limit
limit_y=max(defect);
ylim(axes1,[0 limit_y+2]);

box(axes1,'on');
hold(axes1,'all');

% Create stem
for k = 1:modules
    stem(defect(((k-1)*touchdowns+1):k*touchdowns),'LineWidth',2,...
'DisplayName',sprintf('Module %i', k));
end

% Create xlabel
xlabel('Chip Number','FontSize',14);

% Create ylabel
ylabel('Number of Defective Chains','FontSize',14);

% Create legend
legend(axes1,'show');
hold off;
print ('-depsc2', 'defective_chains_chipwise');
print ('-djpeg', 'defective_chains_chipwise');

% Create figure 2.
defect_per_chip = figure;

% Create axes.
axes1 = axes('Parent',defect_per_chip,...
'XTick', [0:2:26], 'YGrid','on',...
'XGrid','on',...
'FontSize',12);

```

```

% Axis limits.
xlim(axes1,[-1 27]);

limit_y=max(defect_count(2,:));
ylim(axes1,[0 limit_y+1]);

box(axes1,'on');
hold(axes1,'all');

% Create stem.
stem(defect_count(1,:), defect_count(2,:), 'LineWidth',2, 'Parent', axes1, ...
'DisplayName', 'Modul data');

% Create xlabel.
xlabel('Defective Chains per Chip', 'FontSize',14);

% Create ylabel.
ylabel('Number of Chips', 'FontSize',14);

% Create the figure.
print ('-depsc2', 'number_of_chips_with_certain_defect_number');
print ('-djpeg', 'number_of_chips_with_certain_defect_number');

% Create a new folder and put into the files.
mkdir('Defect');
movefile('defective_chains_chipwise.jpg', 'Defect');
movefile('defective_chains_chipwise.eps', 'Defect');
movefile('number_of_chips_with_certain_defect_number.jpg', 'Defect');
movefile('number_of_chips_with_certain_defect_number.eps', 'Defect');
%%
% Section 9: bump resistance per chip.

% Coloured figure.

% Create figure.
Bump_resistance_per_chip = figure;

% Create axes.
axes1=axes('Parent',Bump_resistance_per_chip,'YGrid','off','XTick',...
1:1:touchdowns,'XGrid','off','FontSize',12);

% Uncomment the following line to preserve the X-limits of the axes.
xlim(axes1,[1 touchdowns]);
% Uncomment the following line to preserve the Y-limits of the axes.
ylim(axes1,[10 30]);

box(axes1,'on');
hold(axes1,'all');

% Create stairs.
for k=1:modules

stairs(R_bump_per_chip(((k-1)*touchdowns+1):k*touchdowns), 'LineWidth', 2,...
'Parent', axes1, 'DisplayName', sprintf('Module %i',k));

end

```

```

% Create plot.
plot([0 touchdowns], [R_bump R_bump], 'Parent', axes1, 'LineWidth', 1, ...
'DisplayName', 'Total mean R', 'Color', [1 0 0]);

% Standard deviation lines (lower and upper).
lsigma = plot([0 touchdowns], [S1N_tot S1N_tot], 'Parent', axes1, 'Color', ...
'k', 'LineWidth', 1, 'LineStyle', '--', 'DisplayName', '1\sigma');
usigma = plot([0 touchdowns], [S1P_tot S1P_tot], 'Parent', axes1, 'Color', ...
'k', 'LineWidth', 1, 'LineStyle', '--');

% Exclude line from legend.
set(get(get(usigma, 'Annotation'), 'LegendInformation'), 'IconDisplayStyle', ...
'off');

% Create xlabel.
xlabel ('Chip Number', 'FontSize', 14);

% Create ylabel.
ylabel ('Mean Bump Resistance [m \Omega]', 'FontSize', 14);

% Create legend.
legend(axes1, 'show');

print('-depsc2', 'resistance_chipwise');
print('-djpeg', 'resistance_chipwise');

% Create a new folder and put into the files.
mkdir('Bump resistance');
movefile('resistance_chipwise.jpg', 'Bump resistance');
movefile('resistance_chipwise.eps', 'Bump resistance');

% Black and white figure.

% Create figure.
Bump_resistance_per_chip = figure;

% Create axes.
axes1 = axes('Parent', Bump_resistance_per_chip, 'YGrid', 'off', 'XTick', ...
1:1:touchdowns, 'XGrid', 'off', 'FontSize', 12);

% Uncomment the following line to preserve the X-limits of the axes.
xlim(axes1, [1 touchdowns]);

% Uncomment the following line to preserve the Y-limits of the axes.
ylim(axes1, [10 30]);

box(axes1, 'on');
hold(axes1, 'all');

v_linestyle = {'--' '-' ':' '-.'};

```

```

% Create stairs.
for k = 1:modules

stairs(R_bump_per_chip(((k-1)*touchdowns+1):k*touchdowns), 'LineWidth',...
2, 'Parent', axes1, 'DisplayName', sprintf('Module %i',k), 'Color', 'k',...
'LineStyle', v_linestyle(mod((k-1),4)+1));

end

% Create plot.
plot([0 touchdowns], [R_bump R_bump], 'Parent', axes1, 'LineWidth', 1,...
'DisplayName', 'Total mean R', 'Color', [0 0 0]);

% Standard deviation lines (lower and upper).
lsigma = plot([0 touchdowns], [S1N_tot S1N_tot], 'Parent', axes1, 'Color',...
'k', 'Linewidth', 1, 'LineStyle', '--', 'DisplayName', '1\sigma');
usigma = plot([0 touchdowns], [S1P_tot S1P_tot], 'Parent', axes1, 'Color',...
'k', 'Linewidth', 1, 'LineStyle', '--');

% Exclude line from legend.
set(get(get(usigma, 'Annotation'), 'LegendInformation'), 'IconDisplayStyle',...
'off');

% Create xlabel.
xlabel ('Chip Number', 'FontSize', 14);

% Create ylabel.
ylabel ('Mean Bump Resistance [m \Omega]', 'FontSize', 14);

% Create legend
legend(axes1, 'show');

print('-depsc2', 'resistance_chipwise_bw');
print('-djpeg', 'resistance_chipwise_bw');
% close;

% Move files in the "Bump resistance" folder.
movefile('resistance_chipwise_bw.jpg', 'Bump resistance');
movefile('resistance_chipwise_bw.eps', 'Bump resistance');

%%
% Section 10: how often fails each chain per module.

% Vector with the sum of failures per chain in one module.

chain_failures = modules*touchdowns - sum(L');

% How many have max 2 defects
L_good_chains = chain_failures(:) < 3;
max_2 = sum(L_good_chains)

L_good_chains = chain_failures(:) < 2;
max_1 = sum(L_good_chains)

L_good_chains = chain_failures(:) < 1;
no_bad_chains = sum(L_good_chains)

```

```

% Create figure.
chain_failures_fig = figure;

% Create axes.
axes1 = axes('Parent',chain_failures_fig,'YGrid','on','XTick',1:2:26,...
'yTick',[1:2:10 12:3:32], 'XGrid','on','FontSize',12);

% Axis limit.
xlim(axes1,[0 27]);

limit_y=max(chain_failures);
ylim(axes1,[0 limit_y+5]);

box(axes1,'on');
hold(axes1,'all');

% Create stem.
stem(chain_failures,'LineWidth',2);

% Create xlabel.
xlabel('Chain Number','FontSize',14);

% Create ylabel.
ylabel('Defects per Chain','FontSize',14);
print('-depsc2', 'failures_chainwise');
print('-djpeg', 'failures_chainwise');

% Create a new folder and put into the files.
mkdir('Defect');
movefile('failures_chainwise.jpg', 'Defect');
movefile('failures_chainwise.eps', 'Defect');

```

## References

- [1] Daniel Pitzl, Jan Hampe, Karsten Hansen, The CMS DPIX Group, In-house bump bonding, A note to the CMS Upgrade Management Board (2012).
- [2] W. Erdmann, W. Bertl, R. Horisberger, H.C. Kastli, D. Kotlinski, S. Konig, U. Langenegger, B. Meier, V. Radicci, T. Rohe, A. Starodumov, S. Streuli, Upgrade plans for the CMS pixel barrel detector, Nuclear Instruments and Methods in Physics Research A (2009).
- [3] S. Konig, Ch. Hormann, R. Horisberger, S.Streuli, R. Weber, Assembly of the CMS pixel barrel modules, Nuclear Instruments and Methods in Physics Research A 565 (2006).
- [4] Daniel Pitzl, DESY, DESY Instrumentation Seminar 16.9.2001.
- [5] L. Caminada and A. Starodumov, Building and commissioning of the CMS pixel barrel detector, IOPscience (2009).
- [6] Carlotta Favaro, Universitat Zurich, The Upgrade of the CMS Pixel Detector, Florence, 12-15 October, 2010.
- [7] <http://cms.web.cern.ch/>