Setting up an LHCb TT-Detector Test Stand

Gabriel Landolt

July 26, 2009

Bachelor Thesis

Institute of Physics University of Zurich

under supervision of

Olaf Steinkamp and Achim Vollhardt

LHCb is one of the four experiments at the Large Hadron Collider (LHC) and is designed to measure CP-violation and rare decays in b-physics. Part of its tracking system is the silicon micro-strip detector Tracker Turicensis (TT). The task of this thesis was to set up a test stand for a detector module, employing the same hardware and software as at the TT-station. The test stand will allow to examine specific hardware problems and to perform readout tests.

Acknowledgments

This thesis was only possible with the aid and contribution of the members of the LHCb group at the university of Zurich.

I want to express my gratitude to my tutors *Olaf Steinkamp* and *Achim Vollhardt* for their time and sharing their knowlegde.

A special thank goes to the competent helpers *Angela Buechler*, *Stefan Steiner*, *Mark Tobin*, *Roland Bernet* and to the group of Prof. *Hans-Werner Fink*, who made the SEM-picture of the wire bonds.

Further I'd like to thank Prof. *Ulrich Straumann*, who gave me the opportunity to do this thesis in his group.

Thanks everyone, who helped with words and deeds.

Contents

1 Introduction **4**

2 Assembly of the Test Stand **5**
2.1 Detector and Front-End Electronics 6
2.2 Data Processing and Aquisition 7
2.2.1 Digitizer Board 7
2.2.2 TELL1 8
2.3 Slow and fast control and triggering 8
2.3.1 Control Board 8
2.3.2 Specs Master and Slave 9
2.3.3 Readout Supervisor (*Odin*) 10

3 Installing the Test Stand **11**
3.1 PVSS – DIM 11
3.2 Communication Tests with Control Board via SPECS 12
3.3 Setting up of Front-End Electronics Software 12
3.4 Slow Control of Odin 13
3.5 Installing TELL1 14
3.6 Calibration of Hybrid Temperature Sensors 14
3.7 Adjusting the Digitizer Sampling Time 15

4 Operating the Test Stand **17**

5 First noise measurements **18**
5.1 Raw data 18
5.2 Known Issues 19

6 Broken Bond Problem **21**
6.1 Prototype Half-Module 22
6.2 Half-Module TT111 22

7 Control Board-regulator temperatures **27**

A Commissioning the Test Stand **31**
A.1 How To Control Front-End Electronics 31
A.2 How To Send Triggers 33
A.3 How To Start Data Aquisition 36
A.3.1 How To Open Data in ROOT 37

B Installation of New Hardware **38**

B.1 Instantiation Of New Hardware Types 38
B.2 Add Hardware To Finite State Machine 38

C List of PVSS Panels 40

1 Introduction

The LHCb detector at CERN's LHC is designed to measure observables in B-meson decays, which might go beyond the standard model. It searches also for rare decays of B_s.

Part of the tracking system is the TT silicon strip detector. For its detector modules a test stand was set up at the university of Zurich, in order to be able to examine problems that might occur at the TT-detector in a small, standalone experimental setup.

In Section 2 the assembly of the hardware components of the test stand are described. Section 3 shall give an overview of the setting up and adaption of the controlling software.

Section 4 summarizes, how the test stand is started up and a readout is performed. In Appendix A a detailed step-by-step instruction manual is given.

First practical applications of the test stand are described in the last three sections. In Section 5 the result of first noise measurements is presented. The results of cooling tests on the voltage regulators for the readout electronics can be found in Section 7. Finally Section 6 is dedicated to investigations concerning broken wire bonds that were found on some of the detector modules.

2 Assembly of the Test Stand

In the following the setup of the test stand and its hardware components is described.

The components of the test stand can be divided into four groups: the detector modules, including the silicon sensors and its attached front-end electronics; the Triggering and Fast Control (TFC), providing clocking and triggering; the Slow Control or Experiment Control System (ECS), interfacing human with hardware control and configuration; and the Data Acquisition (DAQ) system, which digitizes, processes and stores the data.

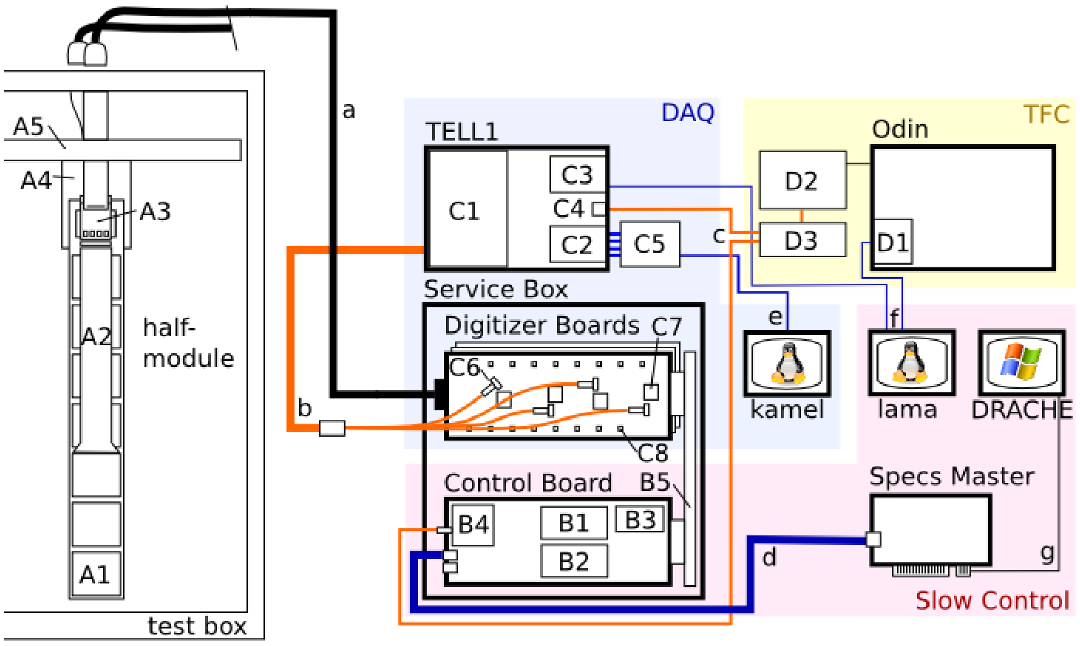


Figure 2.1: Experimental setup of the test stand, omitting power supplies.

Fig. 2.1 shows a scheme of the setup, the numbers in the following refer to the figure.

A: Detector Module and Test Box
A1: Silicon sensor, A2: Kapton interconnect cable, A3: front-end Hybrid with Beetle chips, A4: balcony, A5: heatsink.

B: Slow Control (Control Board, Specs Master)
B1: Specs slave 1, B2: Specs slave 2, B3: voltage regulators with heatsink, B4: TTCrq, B6: backplane.

C: Data Acquisition (TELL1, Digitizer Boards)
C1: Two O-RxCards, C2: GBE-card, C3: CCPC, C4: TTCrx, C5: GBE-switch.
C6: VCSEL, C7: GOL chip, C8: ADC chip.

D: TFC System (Odin)
D1: CCPC, D2: TTCtx, D3: TTCoc.

Transmission Lines
`a`: Analogue Beetle data (el.), `b`: digitized data (opt.), `c`: digital clock and trigger information (opt.), `d`: SPECS-bus (el.), `e`: Gigabit Ethernet, `f`: Ethernet, `g`: PCI-bus.

2.1 Detector and Front-End Electronics

The Silicon Tracker at the LHCb is based on silicon micro-strip detectors. For a so-called *half-module* seven nearly square silicon sensors (`A1`) are mounted on common support rails. The sensors are either split in two groups of 4 and 3 or in three groups of 4, 2 and 1 sensor(s). These groups are called *sectors*. The strips of the outer-most sector are directly electrically connected to the *Hybrid* (`A3`), carrying the readout electronics, the strips of the inner one(s) via a *kapton interconnect cable* (`A2`). In the detector two half-modules are connected to a full-module. [1, 2]

For the test stand as for the TT-station the (half-)modules are mounted inside a light-tight test box providing controllable temperature and humidity conditions. The half-module ist suspended on a cooling *balcony* (`A4`), which on its part is attached to a heatsink (`A5`) equiped with a cooling coil connected to a chiller.

Each readout sector has 512 strips which are bonded via a *Pitchadapter* to the Hybrid on which four *Beetle readout chips* are placed. The Pitchadapter reduces the sensor strip pitch to the pitch of the Beetle input. The Beetles integrate the charge signals for each channel into a shape-tunable peak signal. The charge signal then is analoguely sampled every 25 ns and stored in a ring buffer whose latency is max. 160 sampling intervals. When a trigger-accept forces a readout – knowing the fixed latency between buffering and trigger – the position of the desired event in the ring buffer can be calculated. Each Beetle multiplexes 128 channels onto four output ports with 32 analogue values of 25 ns duration each. Preceding each data packet four binary header bits are sent to the *Digitizer Boards*, containing among others the pipeline column number (see [3] p. 16). The physical transmission line is a roughly 5m-long shielded twisted-pair cable (`a`), it consists out of 16 analogue data balanced lines, the power supply lines and the slow and fast control signal balanced lines. [3, 2]

Signal and Noise The silicon sensors are basically spatially extended diodes sharing a bulk (512 p-type strips on a n-type bulk). By applying a reverse bias voltage ($\simeq$ 500V) the depletion zone is extended to the full sensor thickness. Ionizing radiation and particles produce electron-hole pairs, which are accelerated in the electric field of the depletion zone, yielding a current in reverse direction.

In order to decouple the readout circuit from the leakage currents through the bulk, the readout circuit is AC-coupled by adding an insulating SiO_2 layer between p-strip and grounded metal contact. The positive bias voltage is applied to the backplane of the sensor. (See fig. 2.2.) [4, 5, 2]

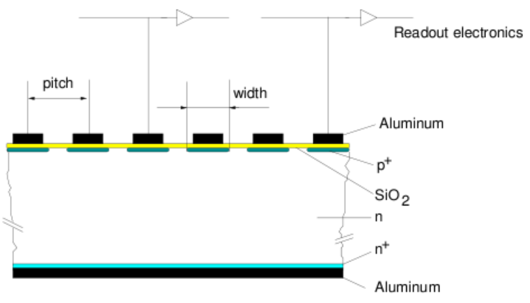


Figure 2.2: Scheme of a silicon strip sensor, out of [4].

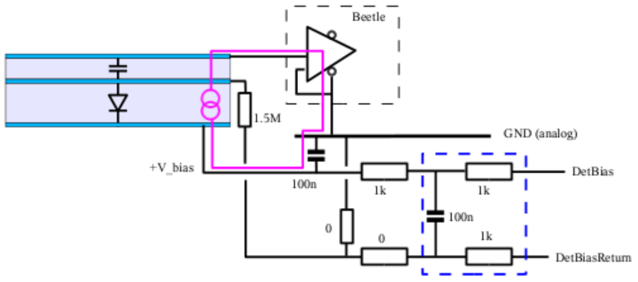


Figure 2.3: Input circuit of a silicon detector, out of [5]

The n-bulk is connected to the bias voltage with a low-pass filter reducing the noise introduced via the bias supply line. In fig. 2.3 the input circuit is drawn in pink. [5]

The discrete nature of the charge carriers leads to fluctuations of the current – noise, one distinguishes between thermal noise and shot noise. The noise is mainly generated by the *charge* integrating amplifier, and is essentially dependent on the diode capacitance. The equivalent noise charge in the order of $ENC = 500e^{-} + 50e^{-}/\mathrm{pF}\cdot C$. [2, 4]

2.2 Data Processing and Aquisition

2.2.1 Digitizer Board

The 5m-long twisted-pair cable leads to the *Service Box*. The Service Box houses the *Digitizer Board* and the *Control Board*, the latter being responsible for the slow control of the Digitizer Board and the Beetles.

On the Digitizer Board the analogue signal is first amplified by *Line Receivers*. Then, three of the header bits and the 32 signalpulses are sampled by Analogue-Digital-Converters (ADCs) (`C8`) at a frequency of 40 MHz and a digitization depth of signed 8 bit. Afterwards the digitized data is serialized and encoded for optical transmission by *Gigabit Optical Link Serializer*s (*GOL*s) (`C7`) [6], which modulate the outputs of laser-diodes (*Vertical-Cavity Surface-Emitting Laser (VCSEL)*) (`C6`). There is one GOL per Beetle chip. The optical data of 12 GOLs (3 Digitizer Boards) is coupled into 12 single fibers, which are combined to a bundle in a ribbon fiber (`b`). (At LHCb the optical fibre allows high-speed long distance transmission of the data out of the pit to the counting house located at a distance of about 100m behind the radiation shielding.) [2]

An additional analogue-digital-converter, the *DCUF* on on the Digitizer Board also allows readout of ta *PT1000* temperature sensor on the connected Hybrid.

2.2.2 TELL1

At the test stand the only ribbon fiber is connected to one of the *O-RxCards* (`C1`) directly mounted on the *Trigger Electronic and Level-1* (*TELL1*) *Board*, which de-serializes and restores the original 32 bits and preceding header format. [7]

The data are then passed to four *Preprocessing-FPGA*s (*PP-FPGA*s) on the TELL1. In normal operation at LHCb, these process the data, doing pedestal subtraction, common mode rejection, clustering/zero suppression. As it is desired for the test stand to have non-zero suppressed data, those processes are omitted. Following the PP-FPGAs, the *SyncLink-FPGA* mainly perform Ethernet and IP framing before transmission of the data with the *Gigabit Ethernet card* (*GBE-card*) (`C2`) via the *GBE-switch* (`C5`). The data (`e`) is the finally received by a GBE-card in the linux PC *kamel*, which stores the data to its hard drive. [7, 2, 8]

The control of the TELL1 Board is done via a *Credit Card PC* (*CCPC*) (`C3`) mounted on the Board, providing standard Ethernet connection (`f`). Standard ssh-protocol provides access to the CCPC allowing firmware flashing, configuration and commissioning of the data aquistion.

2.3 Slow and fast control and triggering

The slow and fast control commands are collected by the *Control Board* and forwarded to the Digitizer Boards and Beetles. The *Odin* Board is the supervisor of the fast-control triggering system.

2.3.1 Control Board

There are two main tasks for the Control Board. On the one hand, it distributes the signals for fast control of the Beetle chips via the Service Box *backplane* (`B6`) and the Digitizer Boards. The *TTCrq mezzanine card* (`B4`) mounted on the Control Board receives among others the clock and the Level 0-Trigger via an optical fibre from Odin (shortcutting the *LHC Timing, Trigger and Control* (*TTC*) distribution system used at CERN). The TTCrq card carries a *TTCrx* chip, delivering the clock together with control and synchronisation information, and the *Quartz Crystal Based Phase-Locked Loop* (*QPLL*) for jitter filtering of the clock signal. Two deskewed reference clocks at LHC bunch-crossing frequency (40.08 MHz) are provided – one serving as the Beetle clock (*Clock40Des1*), the other as clock for the ADCs and GOLs (*Clock40Des2*). In addition after phase adaption by the *Delay25 chip* L0-Trigger, Beetle resetting signal and the testpulse are forwarded as well (see 2.4). [9, 10, 11]

On the other hand, the Control Board hosts two mezzanine cards, *SPECS slave 1* and *2* (`B1,B2`). Via the internal I2C bus they control the already mentioned TTCrq, the De-

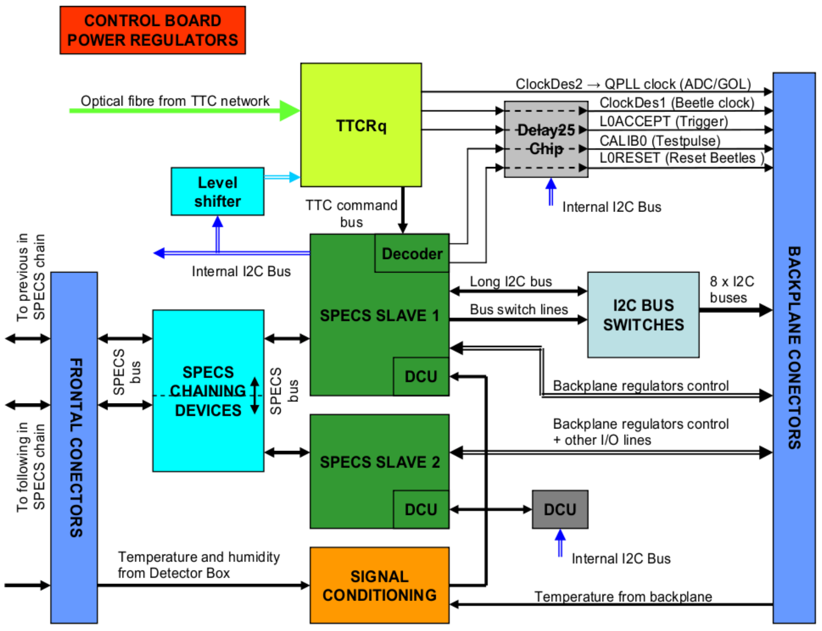


Figure 2.4: Control Board block diagram. [9]

lay25 chip, and a *DCU* (*Detector Control Unit*, radiation-hard ADC with I2C-interface) that is located on the Control Board and is used, among others, for humidity readout in the detector box. Additionally, the SPECS Slaves carry DCUs for the readout of the Service Box backplane- and Control Board regulator-temperatures, monitor regulator-voltages (voltage regulators (B3)) and for the readout of the temperature sensors in the detector box. The readout of the humidity and temperature sensors in the detector box is not used in the test stand. The SPECS Slave 1 controls via I2C the Digitizer Board electronics namely the GOLs, the DCUFs and the Beetle chips. [9]

Further the Control Board controls the powering of the Digitizer Boards and Beetles. [9]

2.3.2 Specs Master and Slave

Using an LVDS serial link, physically transmitted by an Ethernet cable, the SPECS slaves are chained via the serial SPECS bus (d) to the *SPECS master*. Four masters and one *internal SPECS slave* are hosted on the SPECS master board, which is implemented as a PCI-board and at the test stand is plugged in the PCI bus (g) of the PC called *DRACHE*. [12]

2.3.3 Readout Supervisor (Odin)

At LHC*b*, the *Timing and Fast Control* (*TFC*) system drives the LHC*b* readout by distributing timing, trigger and control information to the front-end electronics, whereas Odin supervises the processes. Odin processes the L0 decisions from the *L0-Decision Unit* (*L0DU*) (which itself collects the triggers from the L0 trigger processors namely the Calorimeter Muon processors an the Pile-Up System, and combines all signatures into one decision per crossing [13]) and redistributes them via the TTC system to the front-end electronics. [14]

Odin also provides several means for auto-triggering which do not need the L0DU. These triggers were implemented for calibration and testing: auxiliary external trigger, random trigger, periodic trigger, calibration trigger, timing trigger and more. For the test stand those triggers and the internal clock of Odin are used. The clock is multiplexed with the triggers and other command signals like reset or calibration signals and send optically (`c`) to the Control Board and the TELL1. The conversion into the optical signal is done by the *TTCtx Transmitter* (`D2`), after the optical signal is multiplied by the optical splitter *TTCoc* (`D3`). The configuration and control of the triggering is done on the linux PC *lama*, the physical connection is based on standard Ethernet (`f`) connecting lama and Odin's CCPC (`D1`). [14]

3 Installing the Test Stand

This chapter gives an overview of the setting up of the test stand components, with special emphasis on its control software.

3.1 PVSS – DIM

PVSS (Prozessvisualisierungs- und Steuerungs-System) is a commercial *SCADA* (Supervisory Control And Data Acquisition) tool, that allows object-oriented process visualization and controlling. It is used for the control of all LHC experiments.

The controlling of hardware in PVSS is not directly done via drivers but is provided by the DIM (Distributed Information Management) system. It is based on the mechanism of server–client communication: a (hardware-)*DIM-server publishes* a service, which can be *subscribed* by *DIM-clients*, while the *DIM name server* provides a list of available services. The clients send commands to and receive service data from the server. In PVSS a *data point* is equivalent to a DIM-service. When the server updates (status/data) the service, the data is written into the datapoint, while writing to a datapoint will send a DIM-command to the server. [15, 16]

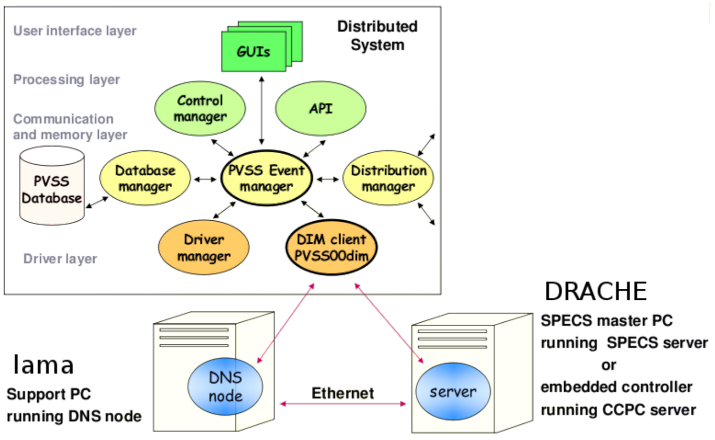


Figure 3.1: PVSS with DIM-server/clients.

Two DIM-servers are used: on the one hand the (DIM-)`SPECS-server` running on the PC hosting the SPECS-master (ie. DRACHE[1]). On the other hand the DIM-servers running on the credit card PC's of ODIN (`TFC_server`) and TELL1 (`ccserv`[2]). The LHC*b* *PVSS Framework* component `fwDIM` provide the DIM-client `PVSS00dim` and also control panels adapted to the particular hardware.

[1]see `http://lhcb-online.web.cern.ch/lhcb-online/ECS/PVSS_SPECS/default.html` for installation procedure

[2]Not running in the test stand because for the TELL1 at the test stand no PVSS-control is intended

3.2 Communication Tests with Control Board via SPECS

As mentioned in the preceding section the control of the hardware is done by the `SPECS-server`. It is a program written in C for WINDOWS and Linux and both executables and source files are part of the `fwSpecs`-component. The libraries `SpecsUser` and `SpecsLib` provide functions and constant definitions to use the Specs functionalities. In order to test if the Specs-drivers[3] were working properly, a couple of C-programs were written using these libraries – testing the Master-Slave communication, the communication between the Slaves and the devices on the Control Board (such as TTCrq, DCU, DCUF, the latter two provide readout of the Control Board's and Hybrid's temperature sensors respectively) and the writing of the registers (controlling for instance the powering of the partitions).

It was found, that on Linux although writing worked properly, each process involving reading – be it direct from a Slave's register or over the I2C bus – led to severe communication errors (SPECS-error code 0x74), which was not to overcome. Therefore the PVSS-project was set up on the WINDOWS boot (PC DRACHE), where such problems were not observed.

3.3 Setting up of Front-End Electronics Software

The used PVSS installation is: `PVSS 3.6 SP1 Windows` with the `Cumulative Patch for 3.6 SP1`. Many components of the LHC*b* PVSS Framework are used as well, albeit some of them with adaptions. Of special interest is the `fwInstallation Tool` which provides a panel for easy component installation and gives a list of the installed components. The component `fwDim` contains beyond others the `PVSS00dim` manager[4] - acting as DIM-client for different needs. The *SPECS-server* `SpecsServer.exe` is part of `fwSpecs` and can be found in the `bin/` folder of the project (see A.1), moreover the `fwSpecs`-component provides all the libraries and panels necessary to control the front-end electronics.

The main task was to generate the required *data point type*s, make instantiations and configure them properly. For each required hardware-type a new PVSS-panel was created (a full list can be found in App. C), containing in each case at least two buttons: one for creating the hardware-type and one for applying the correct configuration to one of its instances. The instantiation is done in the `HW Tool` panel of the `fwHw` framework component (see App. B.1). Finally the device is added to the *Finite State Machine* (*FSM*) using the Device Editor & Navigator (see App. B.2).

The naming of the instances demands special attention: the configuration is implemented

[3]installation instructions: `https://lhcb.lal.in2p3.fr/Specs/SpecsUserDoc/`

[4]For every client the DNS-node had to be specified: In the PVSS console the manager argument `-dim_dns_node lama.physik.uzh.ch` was added

for the hardware composition of LHC*b*. Matching an entire branch of LHC*b*'s DAQ/LV-hierarchy[5] with the assembly of the test stand significantly reduces the effort required to adapt the software.

Some hardware components such as the Hybrids[6] need to configured by an additional, variable configuration so-called *recipes*. For those, additional buttons were implemented on the particular panels. (See tab. C.1 in the Appendix.)

In fig. 3.2 the completely configured instances of the hardware types are listed, the illustration is taken out of the `Hw Tool` of the PVSS project. With this tree of configured hardware it is possible to operate one arbitrary three-sector half-module (operation of a two-sector half-module is also feasible). More than one half-module can currently not be controlled, for that it would be necessary to generate more hardware instantiations (see App. B).

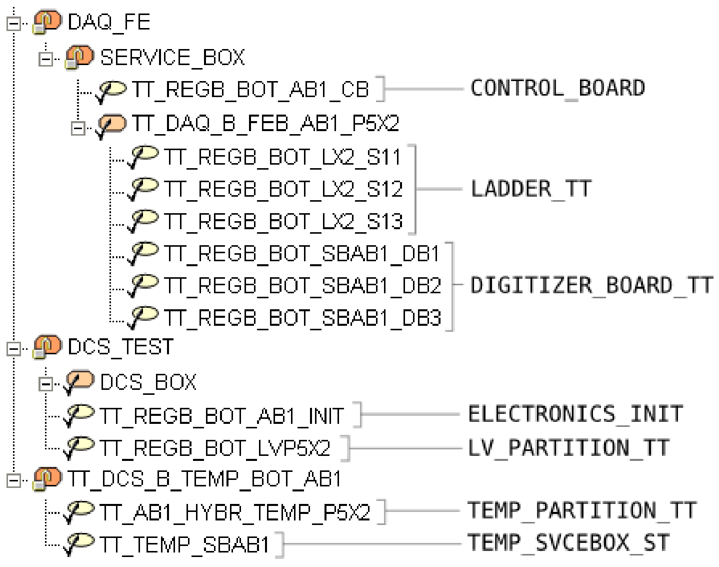


Figure 3.2: Tree of instantations (left) of the hard-ware types (right) at the teststand. *dark orange oval with lock*: control unit objects, *bright orange oval*: objects, *yellow oval*: devices.

3.4 Slow Control of Odin

The used Odin Board is a preliminary release and relies on an earlier version of the PVSS-project, which had been installed for the former burn-in test stand. Therefore the project software for the Odin was not brought up-to-date, but left untouched.

[5]complete TT-partitioning tables can be found here: `https://twiki.cern.ch/twiki/bin/view/LHCb/LHCbST`
[6]In the PVSS-project the term *Ladder* is used to name a Hybrid or a readout sector.

3.5 Installing TELL1

The software of the TELL1 was updated to the newest version which is done on the ssh-accessible TELL1 CCPC. The TELL1 CCPC server software was updated by executing the following in a CCPC console:

```
>> chroot /data/ccpc_SLC4/root bash -c "yum -y install tell1lib"
```

which installed the packets `tell1lib-3.1.-1.i386.rpm` and `exclulib-1.1.0-1.i386.rpm`.

The newest TELL1 FPGA firmware of version v3.1.3 (official version v3.1 produced errors while programming FPGA's) was installed as well:

```
>> EPC16Handling -e -p 1 tell1_ST_v3.1.3.pof
```

The Gigabit Ethernet card was installed in a new linux PC, namely kamel. The card uses non-standard Gigabit Ethernet protocol, making adaptions in the driver necessary, which is only possible on a linux operating system.

In order to run the data aquisition (see sect. A.3) a bunch of parameters have to be set in a configuration file (`/home/cc/ST5.v26_TTTELL01_labUZH.cfg` on CCPC of TELL1). The most important parameters, that had to be adjusted, are the IP and MAC addresses of the destination (ie. kamel) and source (ie. TELL1), the subdetector ID (`Detector ID: 2-ST`), disabling of zero-suppression (`zerosuppress_enable: 0`) and disabling the non-used optical links (`ST link 0-11 disable`).

3.6 Calibration of Hybrid Temperature Sensors

It was necessary to calibrate the readout of the PT1000-sensors (see fig. 3.4) on each Hybrid. The ADC-value for 0°C was determined using a 1000Ω-resistor in place of the PT1000 sensor. A second ADC – temperature pair was made by measuring the air temperature with an external thermometer while the readout electronics and cooling were switched off and the setup was in thermal equilibrium with the environment (which could be verified considering the flatness of the temperature – time curve). The two parameters of the linear calibration curve were directly written in registers `TT_AB1_HYBR_TEMP_P5X2_TEMPERATUEn#.CURVE_SLOPE` and `···n#.CURVE_OFFSET` of the Control Board instance (see fig. 3.2) using the PVSS-panel `Hw Tool` (`#` stands for the Digitizer Board-number). (See in fig. 3.3 the Hybrid temperature as a function of time for calibration with electronics switch off around `12:15:00 PM`.)

During calibration of the temperature sensors it was discovered, that the measured ADC-value was dependent on the number of powered Digitizer Boards. This was due to a known problem of the particular prototype backplane used in this setup, which had too

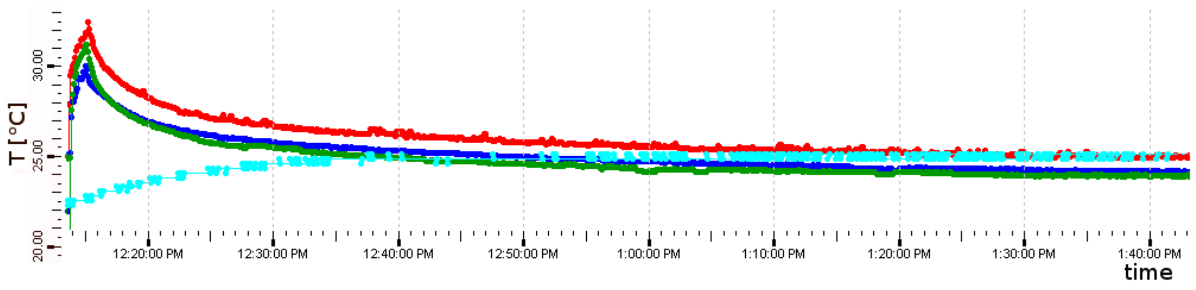


Figure 3.3: Temperature evolution as a function of time. *Blue*, *green*: calibrated L- and K-Hybrid sensors. *Red*: uncalibrated M-Hybrid sensor. *Cyan*: regulator sensor temperature.

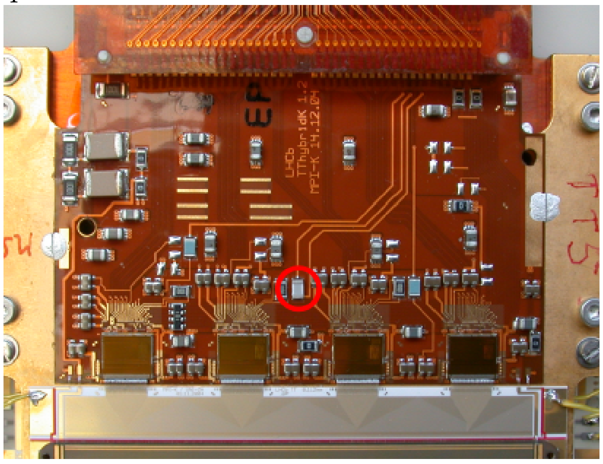


Figure 3.4: A Hybrid with PT1000 temperature sensor marked with the red circle.

small traces for the supply currents, yielding to voltage drops. The problem had already been fixed for the 2.5V-line, but not for the 5V-line. Therefore the affected 5V-lines were bridged by soldering cables with a cross-section about five times larger.

3.7 Adjusting the Digitizer Sampling Time

In order to adjust the ADC sampling time with respect to the Beetle output signal, the fine delay of TTCrq for the Clock40Des2 is to be adjusted.

The delay range is $K \cdot 104$ps with $K \in [0, 239]$. The optimal delay was found by scanning over K by changing the value[7] in the register `TTCrxFineDelay2` of the Control Board instance via the `Hw Tool`. By connecting the optical output of the Digitizer Board to a digital-analogue-converter box, which converts digital optical signal to analogue electrical output, the number of digitized header bits could be observed on an oscilloscope.

[7]The nm-value in `TTCrxFineDelay2` is related to K via conversion table in the App. of [11].

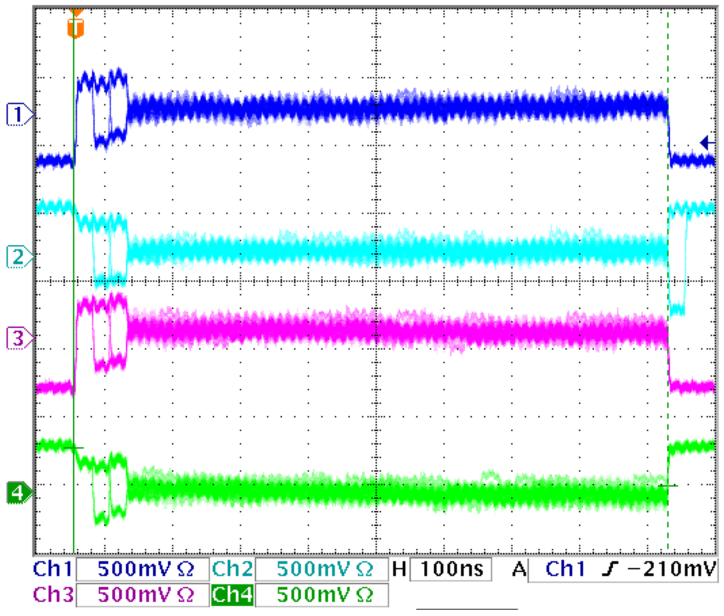


Figure 3.5: Analogue conversion of Digitizer Board output of four output ports of a Beetle, imaged with an oscilloscope.

For correct timing three[8] high level header bits can be seen in front of the analogue channel data (see fig. 3.5), whereas for wrong delays only two or a blurred mixture of both. (The oscilloscope image 3.5 shows also several artefacts of the employed digital-analogue-convert box: the bit after the data in channel 2 (cyan), different baselines for different channels outside of the actual 35 cycles long data block and a ripply overlay.)

It was decided to set the K-value to 0 ($nm = 0x0$E), this value was hardcoded by setting the variable `CLOCK_DES2_FINE_DELAY2` in the PVSS-library `STCommonLibs/stConstantLib.ctl`.

[8]The first header bit is dropped.

4 Operating the Test Stand

In this section an overview is given over the software that has to be used in order to start up the test stand.

Because of the server-client based hardware communication of PVSS (see section 3.1) the main task is to start up the particular hardware servers. The required DNS server is automatically running on start-up of lama.

Front-end electronics The front-end electronics – in this context including the Hybrids, the Digitizer Boards and Control Board – is controlled by the PVSS on DRACHE. First, the SPECS-DIM-server (`SpecsServer.exe`) has to be started up on DRACHE in a command prompt, next the PVSS-project can be initialized. It is then necessary to subscribe all of the hardware types at the server using the `Hw Tool`. Finally one can startup the Finite State Machine in order to initialize the hardware and observe its operating states. See App. A.1 for details.

Triggering First, the DIM-server (`TFC_server`) on the ssh-accessible credit card pc on the Odin Board has to be started. The PVSS-project for controlling Odin is installed on lama, only the panel `TFC_top` has to be started. In the appearing *TFC Startup Panel* the network configurations of Odin have to be set. After configuration, the panel *TFC Local Run Control* opens, which allows the selection and configuration of the triggers. See App. A.2 for details.

Data Aquisition For the TELL1, no DIM-server is needed, because no PVSS-control for it is used in the test stand. The data aquisition is started by ssh-connecting to the TELL1's credit card pc and executing `daq_tell1` with the desired configuration file. The TELL1 will send the data upon a trigger to kamel. See App. A.3 for details.

On kamel the program *writeEventsToBinary* needs to be running, which uses the libraries of the LHC*b* *Event Builder*. It receives all data sent by the TELL1 and writes the events in a binary file. The raw data file is processed by an analysis program called *GetData* which performs a series of algorithms in the *GAUDI framework* [17]. It decodes the ADC-values per strip from the raw data and passes it to the TELL1 emulator, which performs a pedestal subtraction and executes a linear common mode subtraction algorithm. Finally the noise is calculated and the histograms are saved in a root-file. See App. A.3.1 for details.

5 First noise measurements

In order to test the interplay of Odin, the TELL1-Board and the readout software, measurements of detector noise were performed. Due to the much smaller noise level of a detector channel which has a broken bond it is also possible to detect the affected channels. (See Sec. 6.)

Noise The measurements were performed without reverse bias voltage applied to the sensors. The difference in noise between broken and safe channels is bigger, because the depletion zone is reduced to a thin layer – leading to a larger capacitance and thus more noise.

Header Cross-Talk As described in Sec. 2.2.1 for each Beetle output port the analogue data from 32 detector channels is preceded by a header of 4 bits of binary data, which are encoded with a signal level that is significantly higher than the detector noise data. It was observed that there is a cross-talk from these header data to the first few detector channels. These channels show an increased noise.

Non-Zero Suppression As it was desired to store the raw detector data, the TELL1 was configured such that no zero-suppression was performed. Consequently the data processing steps on the TELL1 are omitted. Pedestal subtraction and common mode rejection can be emulated after storing the data on the aquisition pc *kamel*.

Timing As described in Sec. 2.3.1 the two clocks for the Beetle chips on one hand and for ADC and GOL chips on the other hand, Clock40Des1 and Clock40Des2 respectively, are provided by the TTCrq on the Control Board. In order to optimize the ADC sampling point, the two clocks can be independently shifted with the *fine delay registers.* [9]
During readout tests it was found that due to a specific problem of the employed backplane, a clock phase twist led to a shift of half a clock cycle length for some of the Digitizer Board slots. The timing was adjusted for the Digitizer Boards in the non-affected slots by setting the fine delay register of Clock40Des2, with the intention of substituting the backplane later on. See Sec. 3.7 for details.

5.1 Raw data

In the following, the raw non-zero suppressed data from 10000 triggers is displayed for the three Hybrids of half-module TT111. The labels K,M and L for Hybrids refer to their physical length (*k*urz, *m*ittel and *l*ang). They are responsible for the readout of the single-, double- and four-sensor sector of the half-module.

Broken Bonds In fig. 5.1 (a) raw noise (rms) as a function of channelnumber for the K-Hybrid is drawn. The 4-fold structure corresponding to the readout by the four Beetle chip is clearly visible. There are 8×8 low noise channels on the first two Beetles, due to broken Bonds. This was confirmed with a microscopic analysis of the Bond wires (see Sec. 6). Fig. 5.1 (b) shows for the same Hybrid the raw noise of the first 38 channels. The effect of the header cross-talk can be seen on the 0th and 32nd channel, taking into account that they have broken Bonds.

Fig. 5.2 (a) displays the raw noise for each channel of the M-Hybrid. Also here, every forth channel on the first two Beetles shows reduced noise – with the exception of channels 248 and 252.

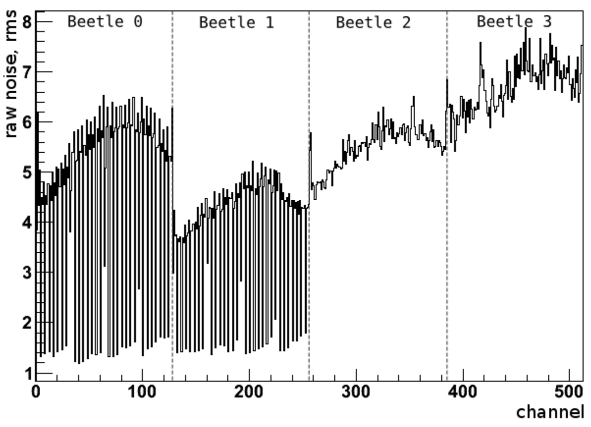
Header Cross-Talk Fig. 5.2 (b) shows the channel noise for the L-Hybrid after common mode (CM) subtraction. A single dead channel can be identified at position 70. The pronounced header cross-talk increases the noise not only on the first channel of a group of 32 channels (corresponding to a Beetle output port), but also on the second.

The linear CM suppression algorithm essentially performs a linear fit to the ADC-values of each set of 32 channels and subtracts the fitted line from the data. Hits are then identified based on an rms-threshold for the 32 channels and set to zero. Subsequently a second linear CM subtraction is performed, after which the final hit detection is performed. [8] This algorithm generally decreases the noise level. However, for channels with broken bonds or header cross-talk the effect is contrary. The main contribution to the CM noise is due to pick-up on the silicon strips. Channels with broken bonds do not have this pickup and therefore the CM algorithm peforms a wrong correction on them. Thus the noise on these channels increases. Header cross-talk affects only single channels and is emphazised by the CM suppression.

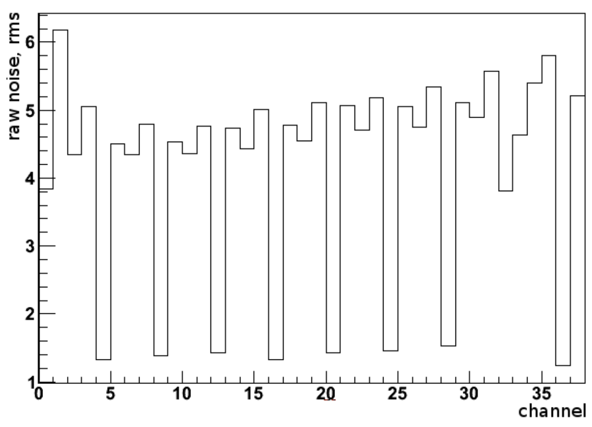
5.2 Known Issues

One optical output channel of one Digitizer Board has too weak light intensity. This is due to a known problem during the board assembly, which led to misaligned laser diodes. The corresponding Digitizer Board has in the meantime been replaced.

Furthermore it was observed that sometimes the digitized output from four Beetle output ports show an temporally alternating mixture of constant base line and of true noise data. It was presumed that a loose contact somewhere in the analogue data chain produces this feature. This problem is to be checked and corrected.

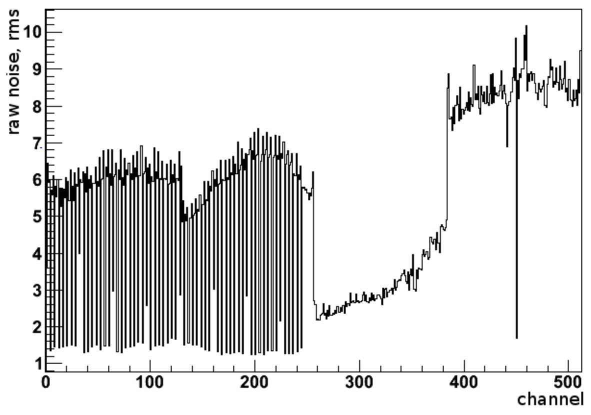


(a)

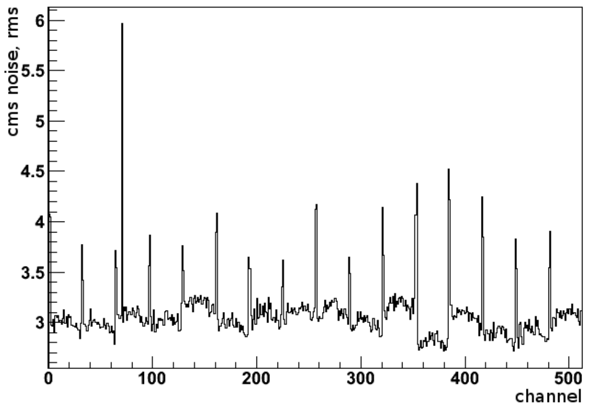


(b)

Figure 5.1: (a) Raw noise (rms) vs. channelnumber for K-Hybrid. (b) Zoom of raw noise (rms) of K-Hybrid from channel 0–37.



(a)



(b)

Figure 5.2: (a) Raw noise (rms) vs. channelnumber for M-Hybrid. (b) Cms noise (rms) for L-Hybrid for channels 0–511.

6 Broken Bond Problem

During operation of the TT station at LHC*b* it was observed that particular channels of certain modules started to show reduced noise. It turned out, that for the affected channels the wire bonds making the electrical connection between the Beetle-chips and the Pitchadapter was interrupted. Due to the density of input pads on the Beetle chips, the connections are made by 4-fold staggered rows of bond wires. Microscopic examinations show that for the affected channels the bond wire is broken close to the bond foot, either on the Beetle or the pitchadapter side. The broken bond wires are part of the innermost row and hence are the shortest wires with the most pronounced curvature. Additionally it was observed that the process of braking starts at one side of the Hybrid (Beetle 0- or Beetle 3-side) and gradually propagates to the other side.

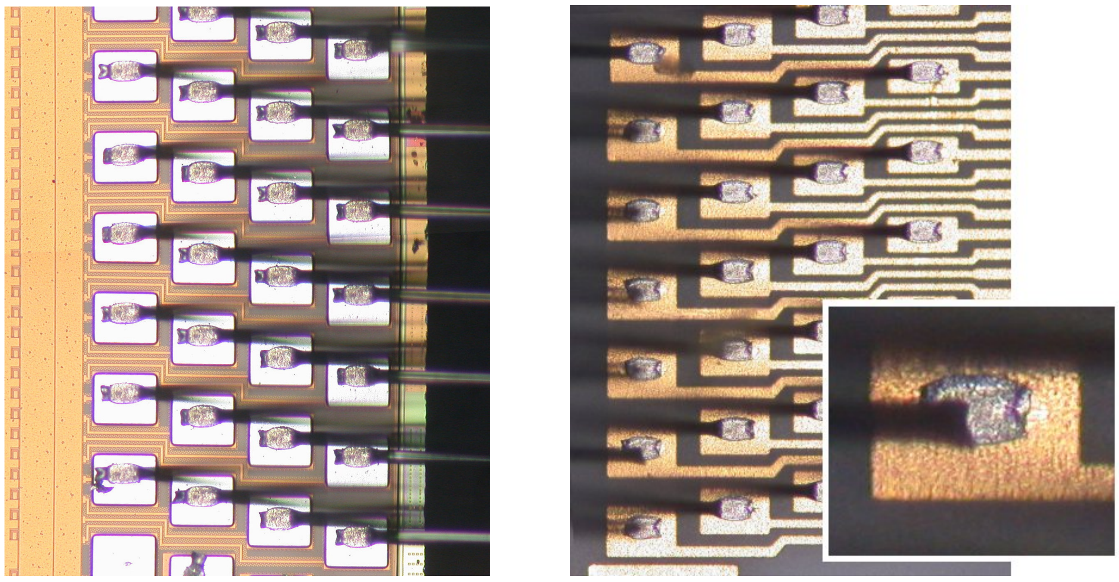

(a) (b)

Figure 6.1: First bonds of Beetle 0 of K-Hybrid of TT111: Beetle side (a), Pitchadapter side (b) with a zoom on a broken bond. Every viewable bond of the innermost row is broken on Pitchadapter side, exept for the 7th on the upper edge of the pictures. Photographies by *S. Steiner*.

Even though all modules were extensively tested before installation in similar conditions at the former burn-in test stand, the problem only occured after the installation at LHC*b*. An attempt was made to reproduce the effect in the test stand, to understand which operating conditions led to the breaking of bond wires. First tests were made on a protoype half-module due to its availability. Later on, a module, which showed to have defect bonds, (half-module TT111 with Hybrids K 109, M 153, L 244) was taken out of the TT-station, and was examined in the test stand.

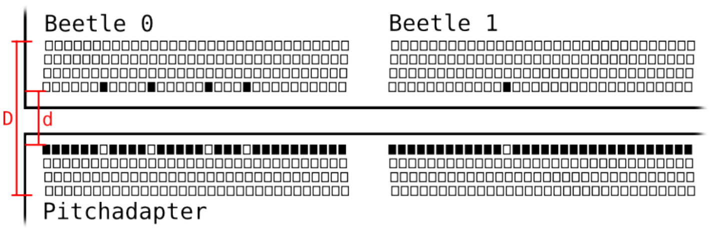
6.1 Prototype Half-Module

During the commissioning of the test stand a two-sector prototype half-module was mounted in the test box in order to be able to set up the Beetle chips and test the operating Control and Digitizer Boards.

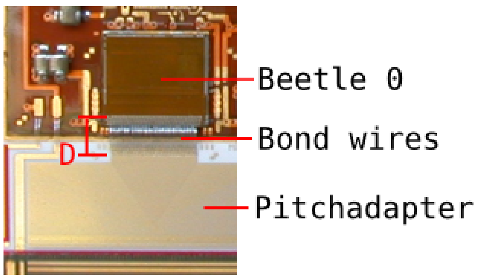
Despite the many times of operating the half-module, with and without active cooling applied, none of the observable bonds of the M-Hybrid did break over the time of the test.

6.2 Half-Module TT111

A potential cause for breaking of wire bonds are mechanical stress due to thermal extent change or vibrations. Also relative movement of the Pitchadapter and the carrier of the Beetle chips could cause a disruption of the bonds. Such relative movement is in principle possible since adhesive tape (3M$^{\text{TM}}$ 467MP) with non-hardening glue was used to mount the Hybrid and the Pitchadapter onto the copper carrier plate. Elevated temperatures may soften the glue. Forces such as due to different coefficients of expansion of the carrier, Hybrid, heatsink etc. or due to the tension of the Kapton interconnect cable could result in relative motions. As this cable was expected to contract in a dry environment the Detector Box was flushed with dry compressed-air. The latter also prevent dewing at low temperatures. See (a) in table 6.1.



(a)



(b)

Figure 6.2: (a) Scheme of the bonds of K-Hybrid on TT111 leading to Beetle 0 and 1. Broken bonds are filled black. The bonds of Beetle 2 and 3 are intact. Additionally the 3 HV-bonds on Beetle 0 side are broken aswell. Furthermore the distance d and D between the first innermost and outermost bondpads respectively of Beetle 0 is drawn. (b) Zoom in on Beetle 0 with a sketch of distance D.

Experiment In a first step it was tried to produce such relative movements by operating the half-module without any external cooling at all.

In a second step the Balcony was cooled to $-20°C$, presuming that this would lead to further contraction of the Kapton cable and to more brittleness of the bonds. About a third of the time the box was flushed with nitrogen, otherwise with compressed-air. (b)

It was also suspected that not temperature (changes) nor contraction of Kapton cable but gravitational force acting on the Kapton cable and the pitchadapter could over extended periods of time result in a relative motion. Therefore the Ladder was left untouched and switched off for about one month. (c)

Finally the Ladder was operated for again about a month with a moderate cooling at 15°C, trying to reproduce the environment at LHC*b*. (d)

Tab. 6.1 gives an overview over the test runs.

Two sets of distinctive spots on the Beetle- and Pitchadapter-side were chosen in order to monitor such displacements: On the one hand the distance D between the outermost pads (illustrated in fig. 6.2) was measured. This allowed measurements with a *surveyor's level* right after the opening of the test box with approximately the same humidity and temperature conditions as during operation. But the accuracy of the surveyor's level was low. On the other hand the distance d between the innermost bond pads was measured, which allowed higher resolved measurements under a microscope but made it necessary to remove the half-module from the test box.

	cooling [°C]	drying	Time [d]	equilibrium temp. [°C] K/M/L	D [mm] B0/B3	d [mm] B0/B3
					2.50/2.40	1.211/1.141
(a)	–	N_2	1.5	40.1/38.5/39.3	2.51/2.37	1.200/1.146$^{*)}$
(b)	$-20°C$	N_2/air	6	1.11/1.95/-1.05	2.47/2.29	1.207/1.145
(c)	–$^{**)}$	–$^{**)}$	40	–$^{**)}$	2.47/2.41	1.195/1.132
(d)	15°C	air	36	26.5/25.8/26.5	–	1.199/1.139

Table 6.1: Test runs, itemized is the kind of cooling and drying, the equilibrium temperature of Hybrids K/M/L, the duration of the run and the measured distances D and d. $^{*)}$ measured after a day with N_2-flushing without Beetle running. $^{**)}$ Beetles not running.

Results After each test run the K-Hybrid was checked using the microscope for additional broken bonds. A rupture of further bonds was not observed, after any of the tests.

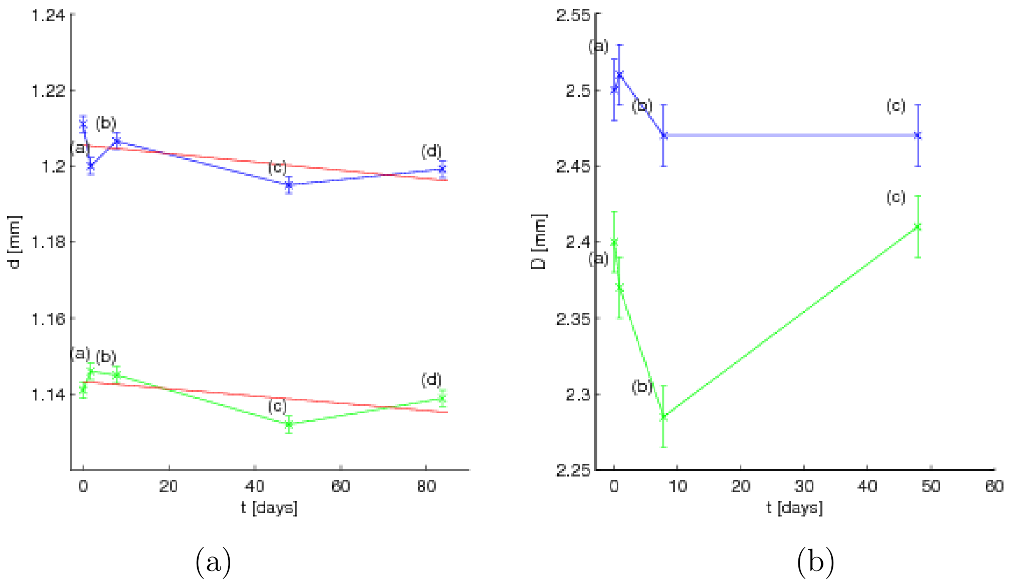


Figure 6.3: (a) time evolution of distance d measured with microscope; (b) evolution of D measured with surveyor's level. Blue: Beetle0-side, green: Beetle3-side

In fig. 6.3 the distances D and d, as defined above, are plotted as a function of time elapsed. Within the measurement precision, no displacement between the Pitchadapter and Hybrid over time was observed. The large outlier in D on Beetle 3-side was not observed at the same time in the corresponding measurement of d.

All values d measured under the microscope are within a range that is smaller than 15 μm. A linear fit results in a slope that produces a change smaller than 9 μm over the whole measurement period of about 80 days. (For comparision: It has been estimated that the distance has to be increased by about 80 μm in order to fully stretch the shortest bond-wires).

To estimate the error on d, a set of measurements was performed alternatingly by two persons, realigning the half-module under the microscope in between any two measurements. As those measurements were done in a time of the order of an hour, systematic errors, which might occur due to the large period over which the measurements (a)–(d) were performed, are not taken into account.

The surveyor's level did not provide a high enough accuracy, to make a statement about any shifting.

Discussion Over a period of about three months operating the detector with and without cooling, it was neither possible to reproduce a further bond breaking on the K-Hybrid nor a measurable displacement of the Pitchadapter.

During the search for defect bonds it was found that the bond feet are rather wide, which can be a sign for *overbonding* (too large pressing force during bonding). (see fig. 6.1 compare to fig. 6.5). Consequently the bond-heel is quite thin, as can be seen from the sharp-edged ruptured bond-wire in fig. 6.1. Furthermore, the bond wire loop heights of the innermost row seem to be rather low (see fig. 6.4 (a)). Both criterions may increase the liability to stress.

It is currently the subject of discussion, whether the bond heels have *initial cracks* due to the bonding process, which are a known phenomenon of wire bonding and increase the probability of bond failure. These initial cracks at the bond heel (see fig. 6.5) are suspected to arise due to many causes, such as the ultrasonic vibration during bonding [18], due to heel fatigue because of the back and forth motion of the wire during the bending process [19] or a combination of both [18]. It is also mentionend, that sharp bonding tools, destructive motion of the bonding tool during or after its lift off or excessive bond deformation may leed to these cracks [19]. Temperature cycling is said to be able to enlarge those cracks [19]. And there seems to be a strong dependence on the wire loop geometry for heel fatigue failures [20].

Without a comparison of loop geometry and initial crack appeareance between affected and unaffected Beetles at LHC*b*, no conclusion about the origin and no predection of possible future events may be given.

A Scanning Electron Microscope (SEM) picture of a broken bond is shown in fig. 6.4 (b). Fig. 6.5 shows an initial heel crack of an unbroken bond whose neighboring bonds of the same row are broken.

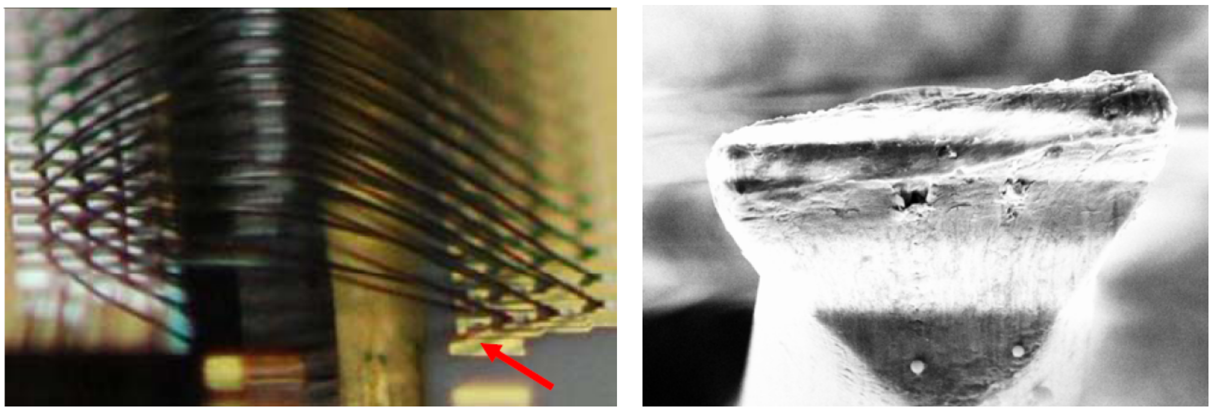

(a) (b)

Figure 6.4: (a) Side view of first Beeltle 0 bond-wires of K-Hybrid of TT111 (Pitchadapter is on right side); (b) SEM-picture of first broken bond on Hybrid TT134 K 124 on Pitchadapterside.

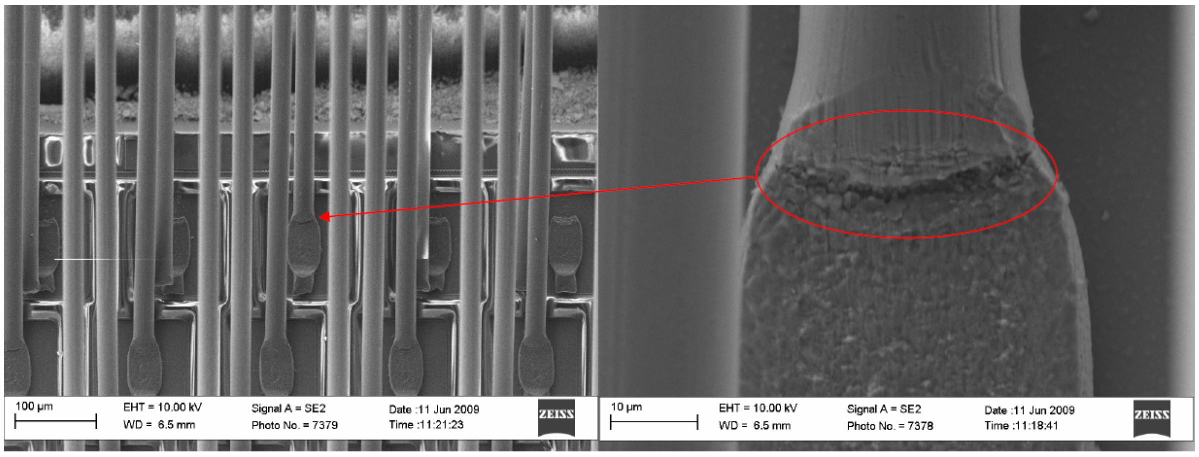


Figure 6.5: SEM-picture of initial heel crack on Hybrid TT134 M 175.

7 Control Board-regulator temperatures

The voltages needed for the electronics on the Control Board are provided by three voltage regulators. Due to their power dissipation, an active cooling is implemented. A water-cooled brass heatsink is mounted on top of the regulators. At LHCb the water supply lines, which are made out of flexible plastic, showed discoloration due to additives in the cooling water. It was discussed to replace the installed heatsinks, therefore tests with different heatsinks using the Control Board temperature readout of the test stand were performed.

Three voltage levels (2.5V, 3.3V, 5V) and therefore three voltage regulators are needed in order to power the devices on the Control Board. The 5V- and the 3.3V-regulator were fed via the Service Box backplane by an 8V power supply and the 2.5V-regulator by an 5V power supply. In order to reduce the heating up of the 3.3V voltage regulator, it was seperated from the 8V supply and soldered to the 5V power supply. (See fig. 7.1.)

The temperature sensor on the back of the voltage regulators was readout for four different cooling scenarios: no heatsink, aluminium heatsink, original brass heatsink without water flow and original setup with water cooling. For each test two digitzer boards were plugged in the service box, reading out as well the corresponding Hybrid temperature. The time it took to reach temperature equilibrium varied for the different heatsinks according to their masses, heat capacities and surface properties.

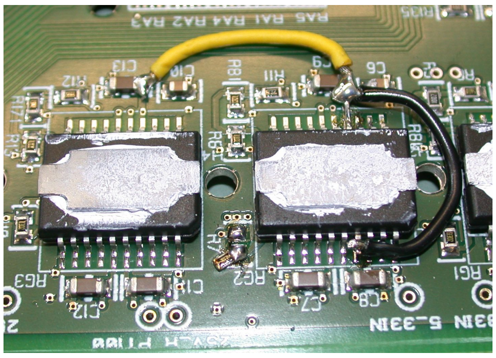


Figure 7.1: Newly soldered regulators. Left: 2.5V, middle: 3.3V, right edge: 5V

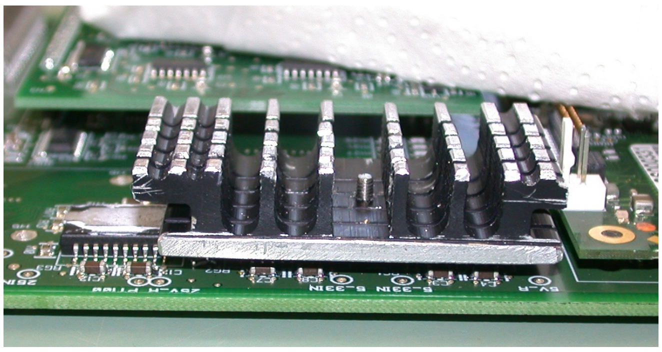


Figure 7.2: Aluminium heatsink on control board (and in the background the wrapped water cooling block)

The equilibrium of the system was stated, when the sensor temperature was constant in time. The equilibrium temperatures T_{eq} of the voltage regulators sensor and the equilibrium temperature T_{eq}^{5V} at the contact area between the particular heatsink and the 5V-regulator, which shows the largest power dissipation, are listed in tab. 7.1. All measurements are performed at ambient room temperature.

No heatsink The voltage regulators are equipped with an overheating protection, therefore the Control Board can be operated whithout cooling, without running the risk of destroying it. The temperature measured by the temperature sensor is shown in fig. 7.3 *left*. As the three voltage regulators were not covered by a heatsink, their temperatures could also be directly measured using a hand-held thermometer, conductive paste providing good thermal contact between regulator and tip of the thermometer. Measured temperatures in equilibrium were: 67.8°C for the 5V-regulator, 60.9°C for the 3.3V-regulator and 45.1°C for the 2.5V-regulator, respectively.

There was also a quick and dirty test of the influence on the temperature, when a slight air flow is directly pointed to the regulators. The temperature measured by the regulator sensor dropped to 32°C. (See peak around `4:02:00 PM` in fig. 7.3.)

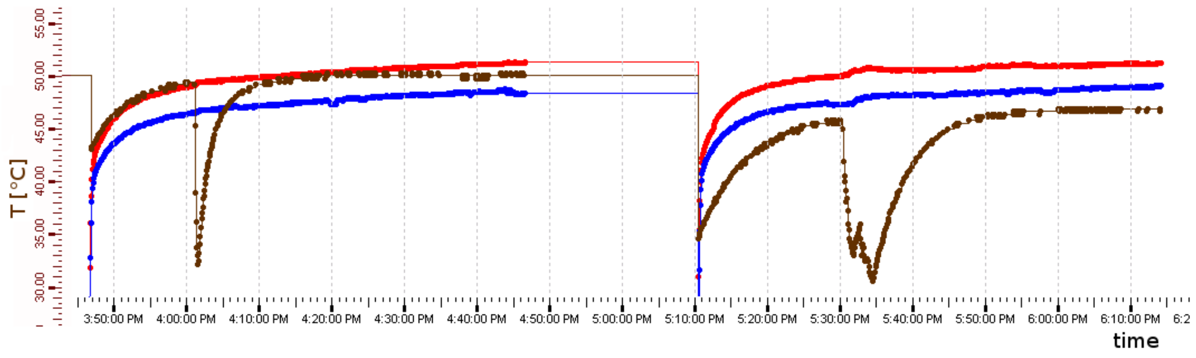


Figure 7.3: Comparison CB-regulator sensor temperature (brown) without (*left*: $T_{eq} = 50.1°C$) and with aluminium heatsink (*right*: measurement 2, $T_{eq} = 46.9°C$). Also drawn is the evolution of the two Hybrid temperatures in blue and red.

Aluminium heatsink In order to enlarge the heat radiating surface, a partially anodized aluminium heatsink with cooling fins was screwed to the Control Board on top of the regulators. The 2.5V regulator, which does not suffer from high generation of heat, was not covered by the heatsink. (See fig. 7.2.)

The temperature evolution was recorded in two independant measurements. In the first measurement the temperature measure by the regulator sensor rose even higher than without heatsink (measurement (I) in table 7.1). At the contact area between the hottest voltage regulator (5V) and the aluminium heatsink the temperature went up to 52.3°C.

Between the first and the second measurement the aluminium was demounted, the amount of conductive paste was increased and measurements with were repeated. In the second measurement ((II) in table 7.1) with the aluminium heatsink the equilibrium temperature was 3.2°C lower than without heatsink. (See right half of 7.3.)

Due to the bigger connected mass the cooling effect by blowing air on the fins is a bit delayed. (See peak around `5:35:00 PM` in fig. 7.3.)

Brass heatsink (no water flow) The cooling efficiency of the original brass heatsink without water circulation was also tested. The temperature was slightly higher compared to the aluminium heatsink cooled system. The temperature at the contact area between the 5V regulator and the heatsink was 51.1°C. (See fig. 7.4.)

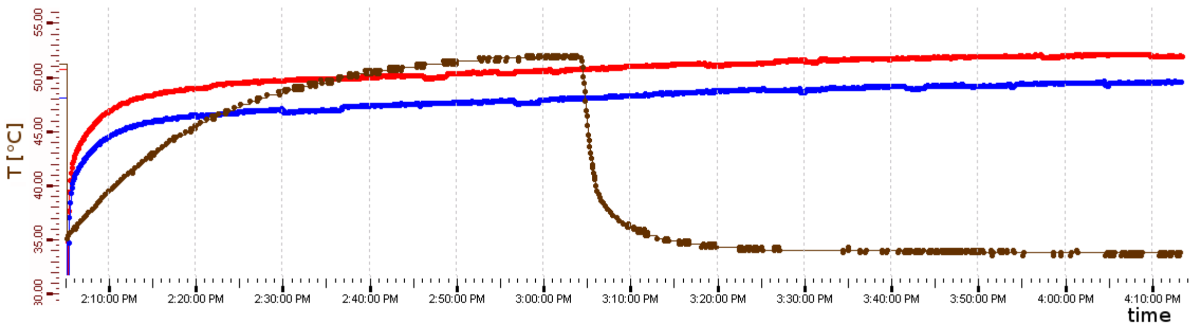


Figure 7.4: CB-regulator temperature (brown), transition from cooling without water flow ($T_{eq} = 52.1°C$) to cooling with normal water flow ($T_{eq} = 33.8°C$).

Brass heatsink With the original heatsink mounted using a customary computer cooling set for pumping and air cooling the water, the equilibrium temperature was 33.8°C. (See fig. 7.4.) The temperature at the contact area between the 5V regulator and the heatsink was 27.4°C.

cooling		T_{eq} [°C]	T_{eq}^{5V} [°C]	time [min]
no heatsink		50.1	–	60
aluminium heatsink	(I)	52.3	52.3	25
	(II)	46.9	–	70
water heatsink (no flow)		52.1	51.1	50
water heatsink		33.8	27.4	65

Table 7.1: Equilibrium temperature of regulator temperature sensor on the Control Board T_{eq} and equilibrium temperature of at the contact area between $5V$ regulator and heatsink T_{eq}^{5V}.

Discussion The passive heatsinks did not significantly lower the temperature at the sensors position, though they might improve the heat eduction from hot spots on the regulator chips. An additional air cooling would be conceivable, but would require significant effort to install in LHCb.

In the meanwhile the heatsinks and the feeding hoses on the LHCb's Control Boards have been substituted by heatsinks with inset brass pipes.

A Commissioning the Test Stand

A.1 How To Control Front-End Electronics

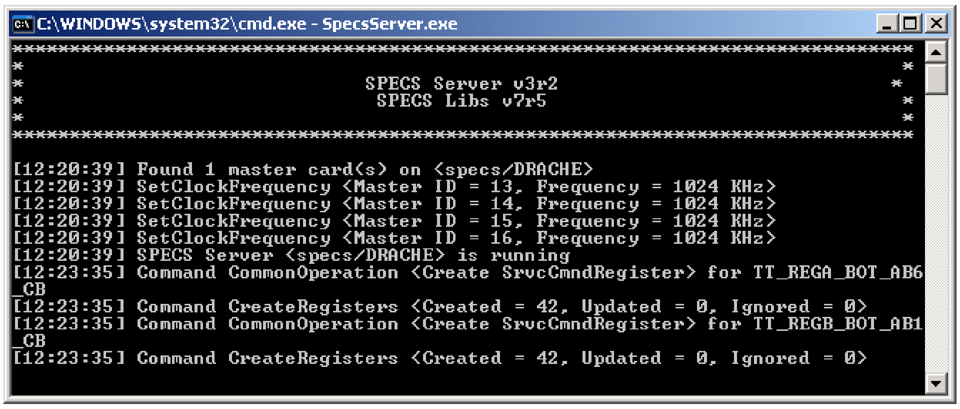
- Make sure, that the DIM name server (DNS) is running on pc *lama*'s *Linux*-partition: Type in a console

  ```
  $ ps aux | grep /app/lhcb/pvss/PVSS_tfc/fwComponents_20050810/bin/dns
  ```

  DNS is running, if the corresponding process is prompted.

  The PVSS-project for controlling the Odin must not be running.

- Log into the *WINDOWS*-booted pc *DRACHE* as *Institut*.



```
C:\WINDOWS\system32\cmd.exe - SpecsServer.exe
******************************************************************************
*                                                                            *
*                             SPECS Server v3r2                             *
*                              SPECS Libs v7r5                               *
*                                                                            *
******************************************************************************

[12:20:39] Found 1 master card(s) on (specs/DRACHE)
[12:20:39] SetClockFrequency (Master ID = 13, Frequency = 1024 KHz)
[12:20:39] SetClockFrequency (Master ID = 14, Frequency = 1024 KHz)
[12:20:39] SetClockFrequency (Master ID = 15, Frequency = 1024 KHz)
[12:20:39] SetClockFrequency (Master ID = 16, Frequency = 1024 KHz)
[12:20:39] SPECS Server (specs/DRACHE) is running
[12:23:35] Command CommonOperation (Create SrvcCmndRegister) for TT_REGA_BOT_AB6
_CB
[12:23:35] Command CreateRegisters (Created = 42, Updated = 0, Ignored = 0)
[12:23:35] Command CommonOperation (Create SrvcCmndRegister) for TT_REGB_BOT_AB1
_CB
[12:23:35] Command CreateRegisters (Created = 42, Updated = 0, Ignored = 0)
```

Figure A.1: SPECS-server output after successful subscription of the Control Board hardware type

- Start the SPECS-server: Open a *WINDOWS* command prompt. Go to the project directory:

  ```
  : cd C:\ETM\PVSS2\3.6\ANGI_TEST\bin
  ```

  Set the environment variable *DIM_DNS_NODE*:

  ```
  : set DIM_DNS_NODE=lama.physik.uzh.ch
  ```

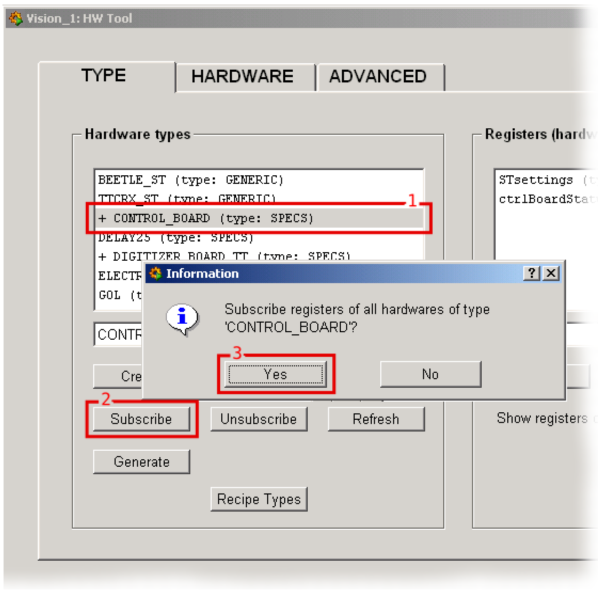
  Start the SPECS-server:

  ```
  : SpecsServer.exe
  ```

  Don't close the window.

- Start the PVSS Administration in the *WINDOWS* Start menu.

- Select the project *ANGI_TEST*, hit the refresh button and then the green traffic light to start up the project. Wait until the whole project is loaded.

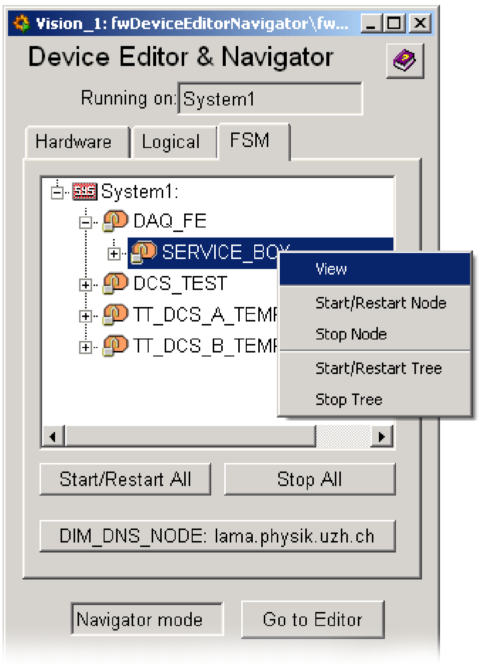


```
BEETLE_ST (type: GENERIC)
TTCRX_ST (type: GENERIC)
+ CONTROL_BOARD (type: SPECS)
DELAY25 (type: SPECS)
+ DIGITIZER_BOARD_TT (type: SPECS)
ELECTR
GOL (t
CONTR
```

```
STsettings (t
ctrlBoardStat
```

Subscribe registers of all hardwares of type 'CONTROL_BOARD'?

Figure A.2: Subscription at the SpecsServer of hardware-types



Running on: System1

```
System1:
  DAQ_FE
    SERVICE_BOX
  DCS_TEST
  TT_DCS_A_TEMP
  TT_DCS_B_TEMP
```

DIM_DNS_NODE: lama.physik.uzh.ch

Figure A.3: Device Editor & Navigator. Control unit objects marked with a lock, devices labelled with a yellow oval.

- Go to the panel called *HW Tool* and subscribe each hardware-type:

  -1- select each one of the items listed under the tab *TYPE*

  -2- hit `Subscribe`

  -3- accept the prompts.

  If the subscription was successful one can see in the output in the SPECS-server window (img. (A.1)) otherwise restart the server and try again. See fig. A.2.

- Go to the panel *Device Editor & Navigator*. Change to the tab *FSM* and ensure that you're in the `Navigator mode`. Perform `Start/Restart All`. Wait until the prompt disappears. See fig. A.3.

- Initialize SPECS and power partitions: Right-click on `DCS_TEST`, in the appearing menu choose *View*.

  A window will open displaying all the sub-systems and their states. In this case the object: `DCS_BOX`. Take over control by clicking on the lock-sign and then pushing `Take`, the color of the drop-down menu displaying `NOT_READY` should change to yellow. Double-click the button `DCS_BOX`.

Click on the yellow drop-down menu `NOT_READY` next to the buttons of the devices named `TT_REGB_BOT_AB1_INIT` and `TT_REGB_BOT_LVP5X2`, choose `SWITCH_ON` and send `PHYSICS` as paramter.

The menus should now display on green ground the state `READY`.

- Initialize front-end electronics: In the *Device Editor & Navigator* window right-click on the item `DAQ_FE`, choose *View*, take over the control. Double-click on `SERVICE_BOX`.

  Set the Control Board `TT_REGB_BOT_AB1_CB` ready by clicking on `NOT_READY` and choosing `Configure` in the menu. In the appearing prompt click `Send` using the recipe `PHYSICS`.

  Double-click on the partition `TT_DAQ_B_FEB_AB1_P5X2` showing every Digitizer Board and Ladder[9] of the current partition.

  Set the state of the Digitizer Boards `TT_REGB_BOT_SBAB1_DB1 – 3` (depending on the half-module) to `READY`. Use again the recipe `PHYSICS`.

  Set aswell the corresponding Ladders `TT_REGB_BOT_LX2_S11 – 13` to `READY`. But this time use the recipe `TEST|VFS400_NOTP`.

- Start temperature readout: In the *Device Editor & Navigator* window right-click on `TT_DCS_B_TEMP_BOT_AB1`, then choose *View*. After setting *READY* `TT_AB1_HYBR_TEMP_P5X2` and `TT_TEMP_SBAB1`[10] the tempertures of the Hybrid- and Control Board-sensors are readout.

  View the temperature curves: If not yet done start the *Trending Editor & Navigator* (look for `fwTrending` in the PVSS console). Ensure that it's in *Navigator mode*. Click on *Manage Plots/Pages*. Right-click on `System1:HYBR_TEMPS_3DB` to see the temperature plots (`CTRL`-`4` and -`8` allows zooming out in the x- and y-axis). Left-click allows configuration of the plot.

- A switch off has to be done by clicking on every drop-down menu displaying `READY` and choosing in the menu `Reset` or `Switch_OFF`. Note that objects can propagate commands to their sub-systems (should not be used for intialization).

A.2 How To Send Triggers

- Login in the Linux-boot of *lama* as *lhcb*.
- Check whether DNS is running:

```
$ ps aux|grep dns
```

[9] In the PVSS-software Ladder has a different meaning than in a hardware context. It's more or less equivalent to a Hybrid.

[10] Due to a nonrelevant programming error this device will change its state to *ERROR*.

and look for `/app/lhcb/pvss/PVSS_tfc/fwComponents_20050810/bin/dns`. If not start it:

```
$ /app/lhcb/pvss/PVSS_tfc/fwComponents_20050810/bin/Dns
```

Important: If `ps aux|grep dns` delivers more than two processes, then the PVSS-project for Odin might be running. *The Odin-PVSS-project has to be shut down properly.*

- Check if sticky bit of `/tmp` is cleared:

  ```
  $ ls -ld /tmp
  ```

  If not clear it (as root):

  ```
  $ chmod -t /tmp
  ```

- Start lock manager:

  ```
  $ source /app/lhcb/pvss/PVSS_tfc/setup
  ```

- Start Odin: Log in:

  ```
  $ ssh tfc@odinv100
  ```

  Execute:

  ```
  $ start_TFC
  ```

  Don't close the shell.

- Start PVSS-project: In a new console type:

  ```
  $ startConsole
  ```

  Click on the green traffic light to start the project `TFC_LC`

- Go to the *Graphic Editor*, click on *Panel* → *Open* and select `TFC_LC/panels/TFC_top.pnl`. Start the panel by clicking *Panel* → *Save & View*.

The following steps are visualized in fig. A.4.

- `-1-` *Select one or several subsystem(s)* tick `None`.

  `-2-` *Select one activity* choose `None` for *Available for all ODINs*.

  `-3-` Click `Find ODIN`. Wait until in *System Configuration* Port 0 is highlighted red.

- `-4-` Cycle through the prompt displaying `OdinDef` and pick `OdinV1_00`.

  `-5-` Click into the prompt. The Port 3 will highlight red.

- `-6-` Click on `Configure System`.

The window *TFC Local Run Control* should have opened. The numbers in the text refer to fig. A.5.

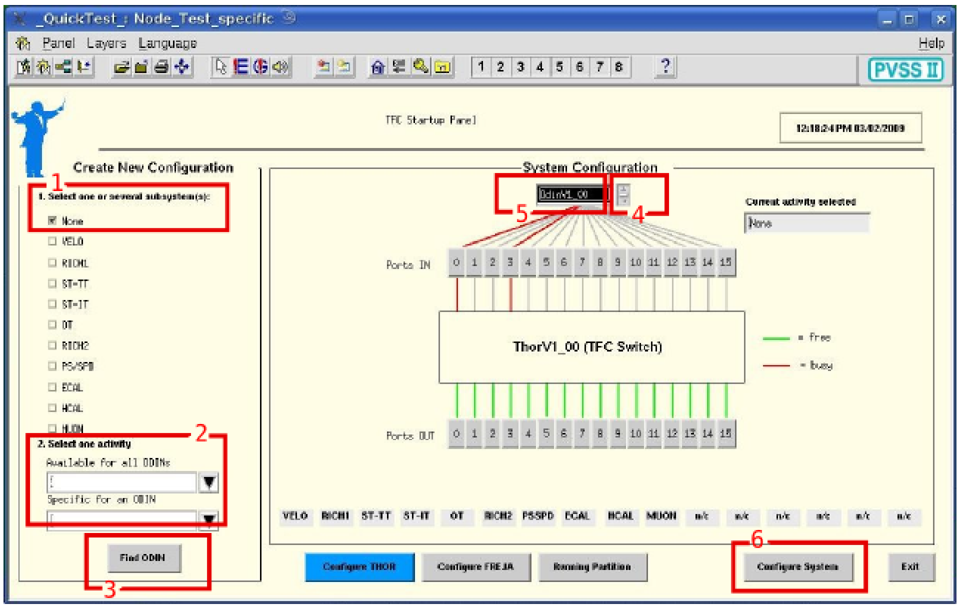


Figure A.4: TFC Startup Panel

- `-1a/b-` Verify that the states of `Partition_OdinV1_00` and `OdinV1_00` are both `RUN_NOT_READY` on orange ground. If this is not the case one has to perform a restart either of the project or of the entire PVSS environment.
- `-2-` In the section *Initialization* click on `Subscribe Cnts` and `Counter Reset`.
- `-3-` One can choose the desired triggers under section *L0 trigger*, eg. *Periodic trigger A*.
- `-4-` The configuration of the trigger can be done by clicking on `ODIN not ready` in the *Configuration* section.

  `-5-` Configuration concerning the trigger frequency, offset and length or delay respectively for periodic and calibration trigger can be set in the *Periodic/calibration trigger SM* section. `Current` loads the current data (right column), `Default` loads the default data, `Apply` applies the new values (left column).

  `-6-` `Apply (all)` writes the desired configuration. (`Default all` can be useful after a loss of standard configuration.)
- `-1a-` In order to start triggering click on the upper orange button `RUN_NOT_READY`, then on `GET_READY` and finally on `START_RUN`.

  `-7-` The L0 Trigger numbers should now count up.

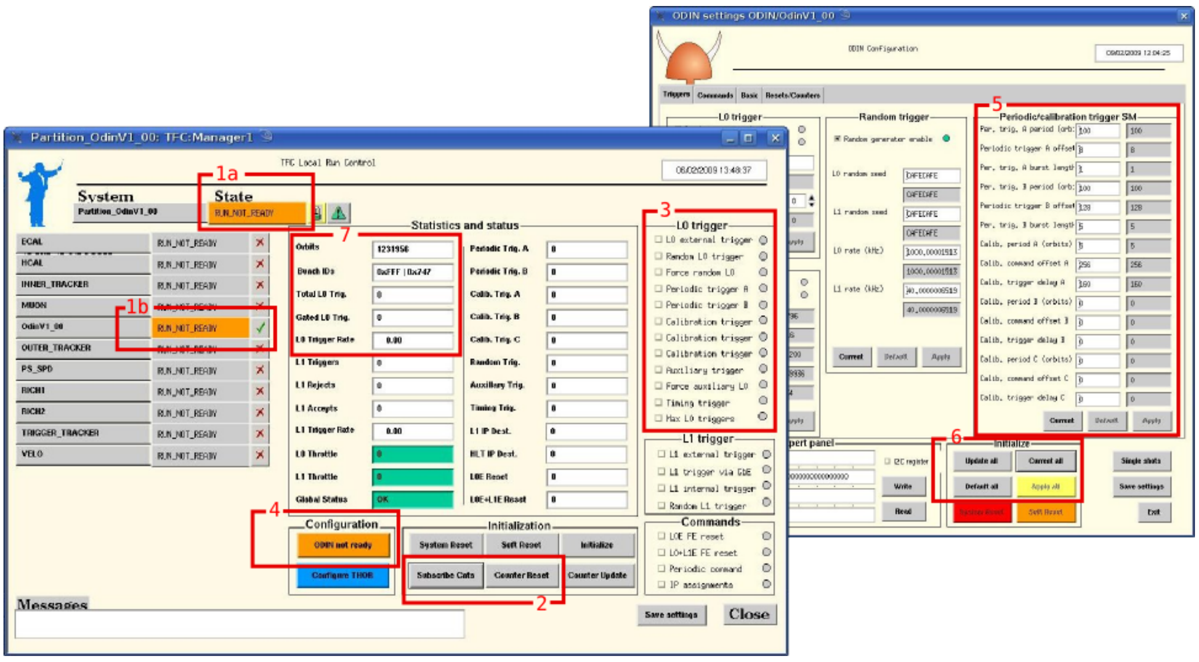


Figure A.5: *left*: TFC Local Run Control, *right*: Odin Configuration

A.3 How To Start Data Aquisition

- You must be logged in the Linux-boot of *lama*.
- Connect to the TELL1's credit card pc:

```
$ ssh cc@ccpc18
```

- Start aquisition:

```
$ daq_tell1 ST5.v26_TTTELL01_labUZH.cfg
```

- Log into *kamel*. Open a root-console. Change to the Event Builder directory:

```
# cd ~/Desktop/Gabriel/ebuild/writeEventsToBinary
```

- Write `<number>` events to binary file `<file>`:

```
# ./writeEventsToBinary <file> <number>
```

  (Binary files can be opened with `hexdump <file> | less`. The data is built out of the following elements: (magic pattern) (bank size) (version/type) (source ID). The magic pattern is in hex: `cbcb`.).

- Send desired triggers with Odin.

A.3.1 How To Open Data in ROOT

- Log into a machine on which the Gaudi anaylsis software is installed in the user's home directory open a z Shell. Append the following lines to the `~/.zshrc` file.

  ```
  # Environment for ROOT
  export ROOTSYS=${CERN}/root ;
  export PATH=${PATH}:${ROOTSYS}/bin ;
  export LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:${ROOTSYS}/lib

  # Environment for LHCb software
  export MYSITEROOT=/app/lhcb/sw
  export CMTCONFIG=slc4_ia32_gcc34
  export CMTPATH=$HOME/cmtuser
  export PATH=$MYSITEROOT/local/gcc-3.4.6/bin:$MYSITEROOT/local/binutils
         -2.15.92.0.2/bin:$MYSITEROOT/scripts:$PATH
  export LD_LIBRARY_PATH=$MYSITEROOT/local/gcc-3.4.6/lib:$MYSITEROOT/lo
         cal/binutils-2.15.92.0.2/lib:$MYSITEROOT/lib:$LD_LIBRARY_PATH
  ```

- Execute the following in a z Shell:

  ```
  $ . $MYSITEROOT/scripts/ExtCMT.sh
  $ cd cmtuser/
  $ setenvOnline v4r15
  $ source cmt/setup.sh
  $ cd ~/TTAnalysis
  ```

- Copy binary file to the current machine. Generate `.root`-file:

  ```
  $ ./GetData <file>
  ```

  The name of the `.root`-file is displayed on the last line of the output.

- Open the file with *ROOT*:

  ```
  $ root <file.root>
  ```

B Installation of New Hardware

B.1 Instantiation Of New Hardware Types

The numbers in the text refer to fig. B.1.

- In the *HW Tool* under the tab `TYPE` the desired hardware type
- -1- Change to tab `HARDWARE`,

  -2- the selected hardware type will be displayed in the upper left frame.
- -3- Fill in a compatible instance name.
- -4- Press `Create`.
- Apply settings to the instance using the appriopriate panel C.

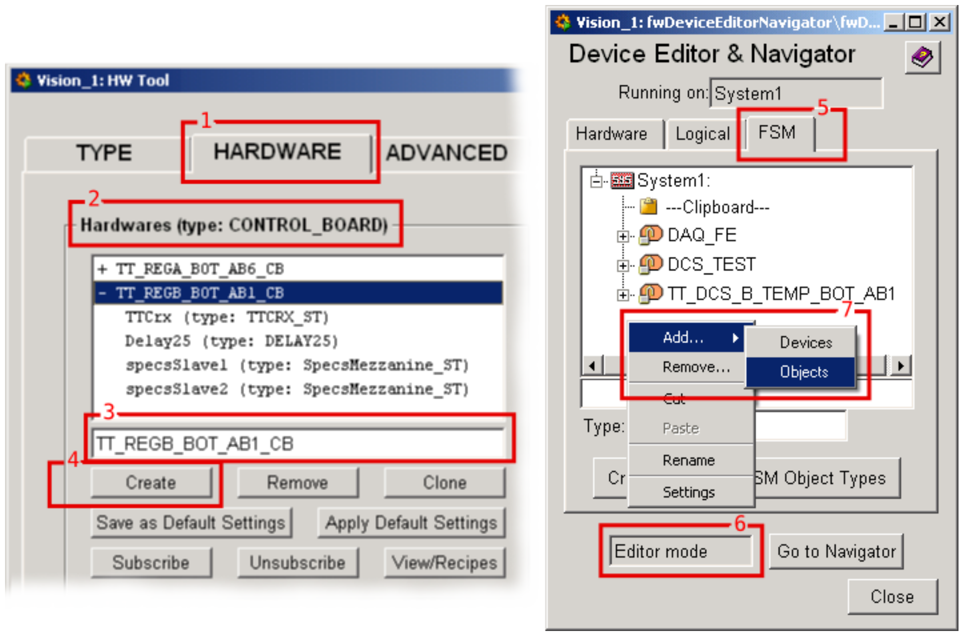


Figure B.1: *left*: Upper left corner of the *HW Tool* tab `HARDWARE`. *right*: *Device Editor & Navigator* tab `FSM` .

B.2 Add Hardware To Finite State Machine

The numbers in the text refer to fig. B.1.

- Go To the Devie Editor & Navigator
- -5- Change to tab `FSM`.

- `-6-` Ensure that you are in `Editor mode`.
- `-7-` Right-click into FSM object list, choose *Add...*.

  Pick either *Devices* or *Objects* (refer to the DAQ hierarchy).
- Choose out of the list the Hardware Type that you want to generate and assign it an appropriate name. Tick the box if you want to create a Control Unit. (refer to the DAQ hierarchy)
- Click on `Generate FSM`, wait until in the Logview the message "*Domain Created*" appears
- Change into `Navigator mode` and hit `Restart All`.

C List of PVSS Panels

Hardware Type	Panel	Functions
CONTROL_BOARD	CreateCBType.pnl	-1- Create ControlBoard Type -2- Apply Settings CB
	Methods[11]: -1- createCtrlBoardType -2- applySettingsCtrlBoards	
	direct libraries: -1,2- stCreateCtrlBoardDpsLibTT_TESTG.ctl	
DIGITIZER _BOARD_TT	CreateDigType.pnl	-1- Create DigitizerBoard Type -2- Apply Settings DigBoard -3- Create recipes DB
	Methods: -1- createDigBoardType -2- applyDBSettingsROSector_Teststand -3- createDigBoardRecipeType createDigBoardArrayRecipe	
	direct libraries: -1- stCreateDigBoardDpsLibTT.ctl -2- TTCommonLibs/namingPartitioningTT.ctl -3- TTCommonLibs/stCreateRecipesLibTT.ctl	
	CreateDBRecipes.pnl	-1- createDigBoardRecipe -2- Recipe type
	Methods: -1- createDigBoardRecipe -2- createDigBoardRecipeType	
	Libraries: -1,2- stCreateDigBoardRecipesLibTT.ctl TTCommonLibs/stCreateRecipesLibTT.ctl	
LADDER_TT	createLadderType.pnl	-1- Create Ladder Type -2- Apply Settings Ladder -3- Create recipes Ladder
	Methods: -1- createLadderType -2- applyLadderSettingsROSector_Teststand -3- createladderRecipe	
	direct libraries:	

[11]called on button click

	-1- stCreateLadderDpsLibTT.ctl -2- TTCommonLibs/namingPartitioningTT.ctl -3- TTCommonLibs/stCreateRecipesLibTT.ctl	
LV_PARTITION _TT	CreateLVPartitionType.pnl	-1- Create LVPartition Type -2- Apply Settings LVPartition -3- Apply Configs LV Partition -4- CreateLVPartition Recipe
	Methods: -1- createLVPartitionType -2- applyPartSettingsROSector_Teststand -3- applyConfigs -4- createLVPartitionRecipeType	
	direct libraries: -1- stCreateLVPartitionDpsLibTT.ctl -2,3- TTCommonLibs/namingPartitioningTT.ctl -4- createLVPartitionRecipe	
	CreateLVPartitionDps.pnl	-1- CreateLVPartitionDps
	Methods: -1- CreateLVPartitionDps	
	direct libraries: -1- TTCommonLibs/namingPartitioningTT.ctl TTCommonLibs/stConstantsLib.ctl stCreateLVPartitionDpsLibTT.ctl	
ELECTRONICS _INIT	CreateINIT.pnl	-1- Create INIT -2- Apply Settings INIT
	Methods: -1- createInitiatorType -2- applySettingsInitiator_Teststand	
	direct libraries: -1- stCreateInitiatorDpsLibTT -2- TTCommonLibs/namingPartitioningTT.ctl	
TEMP _PARTITION_TT TEMP_SVCEBOX _ST	CreateINIT.pnl	-1- Create Temperature Type -2- Apply Partition Settings Temperature -3- Apply service box settings temperature
	Methods: -1- createTempChannelType createHumBoxType createPartitionsType createSvceBoxType	

	-2- applySettingsPartitionsTemp_Teststand -3- applySettingsSvceBoxTemp_Teststand	
	direct libraries: -1,2,3- stCreateTempHumLibTT.ctl	
	createTempDps.pnl	-1- createTempDps
	Methods: -1- createTempDps	
	direct libraries: -1- stCreateTempHumLibTT.ctl	

Table C.1: List of panels.

master ID	0x10
slave1	0xB
slave2	0xC

References

[1] Olaf Steinkamp. Silicon strip detectors for the lhcb experiment. *Nucl. Instr. Meth. A541*, pages 83–88, 2005.

[2] Achim Vollhardt. *An Optical Readout System for the LHCb Silicon Tracker.* PhD thesis, Universität Zürich, 2005.

[3] *The Beetle Reference Manual.* http://wwwasic.kip.uni-heidelberg.de/lhcb/Publications/BeetleRefMan_v1_3.pdf.

[4] Phillip Sievers. *A Silicon Inner Tracker for the LHCb Experiment.* PhD thesis, Universität Zürich, 2002.

[5] C. Bauer, D. Esperante, R. Frei, U. Straumann, P. Vazquez, and A. Vollhardt. Grounding, shielding and power distribution for the lhcb silicon tracking. LHCb note 2004-101, 2005.

[6] P. Moreira et al. *GOL Reference Manual Version 1.5.* http://proj-gol.web.cern.ch/proj-gol/gol_manual.pdf.

[7] *TELL1 Specification for a common read out board for LHCb.* http://lphe.epfl.ch/tell1/specification_and_documents/TELL1.pdf.

[8] Guido Haefeli. *Contribution to the development of the acquisition electronics for the LHCb experiment.* PhD thesis, EPFL, 2004.

[9] Daniel Esperante and Achim Vollhardt. *Design and development of the Control Board for the LHCb Silicon Tracker.* http://cdsweb.cern.ch/record/1082457/files/lhcb-2007-153.pdf.

[10] *TTCrq Manual Version 1.5.* http://proj-qpll.web.cern.ch/proj-qpll/images/manualTTCrq.pdf.

[11] *TTCrx Reference Manual Version 3.11.* http://ttc.web.cern.ch/TTC/TTCrx_manual3.11.pdf.

[12] Dominique Breton, Daniel Charlet, Patrick Robbe, and Ioana Videau. *SPECS: A Serial Protocol for the Experiment Control System of LHCb.* http://www.usc.es/gaes/ST_Elec/Material/Specs4.0.pdf.

[13] R Cornat, J Lecoq, and P Perret. Level-0 decision unit for lhcb. Technical Report LHCb-2003-065, CERN, http://cdsweb.cern.ch/record/691537/files/lhcb-2003-065.pdf, Sep 2003.

[14] Z. Guzik and R. Jacobsson. *LHCb Readout Supervisor 'ODIN' with a L1 Trigger - Technical reference.* http://lhcb-online.web.cern.ch/lhcb-online/TFC/documents/OdinL0L1_lhcb2005-edms704078-1.pdf, 2005.

[15] C. Gaspar. *DIM Distributed Information Management System.* http://dim.web.cern.ch/dim/dim_intro.html, May 2006.

[16] C. Gaspar. Ecs tutorial. LHCb week, March 2006.

[17] *Gaudi framework.* http://lhcb-comp.web.cern.ch/lhcb-comp/Frameworks/Gaudi/.

[18] Ray Fitzsimmons and E. Henry Chia. Propagation mechanism and metallurgical characterization of first bond brittle heel cracks in alsi wire. *Components, Hybrids, and Manufacturing Technology, IEEE Transaction*, 25(6)::1081–1085, 2000.

[19] *Wire Bonding.* http://www.ept.tkk.fi/Teaching/S1133140/WB

[20] S. Ramminger, N. Seliger, and G. Wachutka. Reliability model for al wire bonds subjected to heel crack failures. *Microelectronics Reliability*, 40:1521–1525, 2000.

[21] George Harman. *Wire Bonding in Microelectronics.* Mcgraw-Hill Publ.Comp., 1997.