

**Development and Implementation of a
New Trigger and Data Acquisition System
for the HADES Detector**

Dissertation
zur Erlangung des Doktorgrades
der Naturwissenschaften

vorgelegt beim Fachbereich Physik
der Johann Wolfgang Goethe-Universität
in Frankfurt am Main

von
Jan Michel
aus Frankfurt am Main

August 2012
(D30)

vom Fachbereich Physik der
Johann Wolfgang Goethe-Universität als Dissertation angenommen.

Dekan: Prof. Dr. Joachim Stroth
Gutachter: Prof. Dr. Joachim Stroth
Prof. Dr. Christoph Blume
Datum der Disputation: 16.11.2012

Zusammenfassung

In Großexperimenten der modernen Kern- und Teilchenphysik ist neben dem Bau spezialisierter Detektoren auch das Design der Trigger- und Datenaufnahmesysteme von entscheidender Bedeutung. Insbesondere in Experimenten, die auf den Nachweis seltener Teilchen ausgerichtet sind, sind eine hohe Ereignisrate und eine schnelle Datenaufnahme zentrale Aspekte. Eine große Anzahl an Teilchenspuren erfordert eine hohe Granularität der Detektoren, wodurch das zu transportierende Datenvolumen wesentlich erhöht wird. In vielen Experimenten liegt die typische Datenrate deshalb bei 200 bis 1000 MByte/s.

Die Struktur des Trigger- und Auslesesystems ist bei allen Experimenten ähnlich aufgebaut: Eine zentrale Instanz generiert ein Signal, das die Auslese von allen Teilsystemen zeitgleich startet. Die Signale von jedem einzelnen Kanal des Detektors werden verarbeitet und digitalisiert, bevor sie zur weiteren Analyse und Speicherung zu einem Rechenzentrum transportiert werden. In manchen Systemen werden weitere Schritte zur Datenselektion durchgeführt, um die Menge an zu verarbeitenden Daten zu reduzieren.

Der Schwerpunkt dieser Arbeit liegt in der Entwicklung eines neuen Datenaufnahmesystems für das High Acceptance Di-Electron Spectrometer HADES am GSI Helmholtzzentrum für Schwerionenforschung in Darmstadt. Seit 2002 werden mit HADES Experimente durchgeführt. In den vergangenen Jahren wurde nun ein Teil der Detektoren ausgetauscht und auch das Datenaufnahmesystem grundlegend neu gestaltet. Das Ziel der Arbeiten war die Erhöhung der erzielbaren Ereignisraten in Schwerionenkollisionen um den Faktor 30 von etwa 700 Hz mit dem ursprünglichen System auf 20 kHz. Bei Experimenten mit leichten Kernen soll die Ereignisrate auf mehr als 50 kHz gesteigert werden.

Das Grundkonzept der neuen Elektronik basiert auf programmierbaren Logik-Bausteinen, die über das komplette Detektorsystem verteilt montiert sind. Zwischen den einzelnen Komponenten werden die Daten mittels optischer Fasern übertragen, um so die Menge an elektromagnetischen Störungen in den Detektoren zu reduzieren. Neben der Daten- und Ereignisrate muss das verwendete Übertragungsprotokoll noch weitere Bedingungen erfüllen. Die Latenz der Übertragung sollte sehr gering (weniger als 5 μ s) sein, um den Austausch von Triggerdaten und Statusmeldungen für jedes aufzuzeichnende Ereignis zu ermöglichen. Weiterhin müssen sämtliche Status- und Kontrollinformationen über dasselbe Netzwerk übertragen werden, da auf Grund der räumlichen Beschränkungen in vielen Fällen nur eine Datenleitung verfügbar ist. Aus diesen Gründen wurde ein spezielles Netzwerkprotokoll, das Trigger and Readout-Board Network (TrbNet), entwickelt.

Dieses Protokoll kombiniert alle Schritte der Datenauslese in einem gemeinsamen Netz-

werk, wobei die Abhängigkeit der einzelnen Funktionen untereinander durch getrennte Verarbeitung auf ein Minimum reduziert wird. Das Netzwerk weist eine sternförmige Struktur auf: Im Zentrum stehen die Trigger- und Kontrollsysteme, die ihre Informationen an alle Module jedes Detektorsystems verteilen. Insgesamt sind 500 solcher Module über den gesamten HADES Detektor verteilt. Ihre Daten werden durch die Netzwerkinfrastruktur zu 30 Datenströmen kombiniert. Diese werden dann mittels eines herkömmlichen Computernetzwerks (Gigabit Ethernet) zu mehreren Servern ("Event Builder") weitergeleitet, wo sie kombiniert und gespeichert werden. Neben den reinen Detektordaten werden auch Informationen zum Status aller Systeme und mögliche Fehlermeldungen für die spätere Analyse gespeichert.

Das Auslesesystem der Driftkammern stellte besondere Herausforderungen an das Design der Datenaufnahme: Die hohe Anzahl an Kanälen, die Kompatibilität zu bestehender Analogelektronik und das beschränkte Platzangebot mussten berücksichtigt werden. Im Zuge der Neugestaltung der Datenaufnahme wurde ein besonderes Augenmerk auf die Verbesserung und Vereinheitlichung der Kontroll- und Überwachungssysteme gelegt. So ist es nun möglich, jedes Modul individuell anzusprechen und seinen Status abzufragen.

Für das HADES Experiment wurden viele Elektronik-Module neu entwickelt, die nun auch in anderen Experimenten, insbesondere an der im Bau befindlichen Beschleunigeranlage FAIR, und Versuchsaufbauten zum Einsatz kommen. Zur Zeitmessung wurde bei GSI beispielsweise eine TDC-Plattform (Time-to-Digital-Converter) zur präzisen Zeitmessung im Bereich von 100 ps entworfen.

Das neue Datenaufnahmesystem von HADES wurde während mehrerer Test-Experimente mit Gold-Ionen in den Jahren 2010 und 2011 erfolgreich eingesetzt. Hier wurden alle Anforderungen an das System erfüllt und zum Teil noch übertroffen. Die erzielte Ereignisrate lag bei etwa 13 kHz und war durch die Belastung in den Detektoren und die Beschleunigeranlage limitiert. Die Totzeit der Datenaufnahme lag dabei unter 20%. In Versuchen mit künstlich generierten Daten erreichte das Datenaufnahmesystem jedoch Raten von mehr als 60 kHz und 800 MByte/s. Während des Experiments wurden Datenmengen von 250 MByte/s durch ein System von vier Servern empfangen und gespeichert.

Abstract

One of the crucial points of instrumentation in modern nuclear and particle physics is the setup of data acquisition systems (DAQ). In collisions of heavy ions, particles of special interest for research are often produced at very low rates resulting in the need for high event rates and a fast data acquisition. Additionally, the identification and precise tracking of particles requires fast and highly granular detectors. Both requirements result in very high data rates that have to be transported within the detector read-out system: Typical experiments produce data at rates of 200 to 1,000 MByte/s.

The structure of the trigger and read-out systems of such experiments is quite similar: A central instance generates a signal that triggers read-out of all sub-systems. The signals from each detector system are then processed and digitized by front-end electronics before they are transported to a computing farm where data is analyzed and prepared for long-term storage. Some systems introduce additional steps (high level triggers) in this process to select only special types of events to reduce the amount of data to be processed later.

The main focus of this work is put on the development of a new data acquisition system for the High Acceptance Di-Electron Spectrometer HADES located at the GSI Helmholtz Center for Heavy Ion Research in Darmstadt, Germany. Fully operational since 2002, its front-end electronics and data transport system were subject to a major upgrade program. The goal was an increase of the event rate capabilities by a factor of more than 20 to reach event rates of 20 kHz in heavy ion collisions and more than 50 kHz in light collision systems.

The new electronics are based on FPGA-equipped platforms distributed throughout the detector. Data is transported over optical fibers to reduce the amount of electromagnetic noise induced in the sensitive front-end electronics. Besides the high data rates of up to 500 MByte/s at the design event rate of 20 kHz, the network protocol has to fit to further constraints. The latency of the network must be particularly low to be able to establish a handshake mechanism between the central control system and front-ends for each event: 5 μ s to transport a data packet from the central controller to a front-end or back. Additionally, the network has to handle all control and monitoring functionality since only one common communication connection can be afforded due to space constraints. Hence, a dedicated network protocol, TrbNet, was developed.

The protocol combines all necessary steps of the data acquisition process on one network connection but keeps the dependency of the types of data low by splitting the data stream during every step of the transportation. The network is built with a star-like structure: The center is formed by the trigger and control systems which distribute their signals to all front-

ends of the individual sub-systems. In total, 500 front-ends are installed in the HADES detector. Their data is combined in 30 data streams by the network hubs. These streams are packed into packets sent on Gigabit Ethernet to the server farm (“Event Builders”). Here, the data streams are finally combined to full events and forwarded to permanent storage. All data contains error flags from front-ends to notify later analysis processes about possibly missing or corrupted data.

The front-ends and the data path for read-out of the multi-wire drift chambers (MDC) of the spectrometer asked for individual solutions with respect to data transceivers and network features like error correction mechanisms. The main aspects were strict space constraints, high channel count and a position close to the central part of the detector.

In the scope of the DAQ upgrade the control and monitoring possibilities of all sub-systems were greatly improved. Now, the status of each individual front-end can be read out. For example fill-levels of buffers and occupancy of the read-out paths can be monitored to be able to pin-point all possible errors that might occur during an experiment to their exact location in the system.

For the HADES DAQ upgrade, numerous new electronic modules have been developed and are operated successfully in the detector set-up. From these developments other experiments profit as well. For example, the time-measurement (TDC) platform TRB2 is used in several other experiments as well. Based on this board, a new version of the platform has been developed recently and provides up to 256 channel with a time resolution in the order of 10 ps. Additionally, the board can be equipped with AddOn modules to provide further functionality.

The new DAQ system was tested during several commissioning runs in 2010 and a final four-day heavy ion beam test in August 2011. The system showed a very good performance at or above the predefined requirements. The typical event rate was 13 kHz, mainly restricted by the load on detectors and the capabilities of the accelerator complex. The resulting data rate was 250 MByte/s and was handled by four servers sharing the load. Benchmarks of the full read-out chain showed that the system is able to handled event rates above 50 kHz with low data load and data rates above 800 MByte/s without problems.

Contents

Zusammenfassung	III
Abstract	V
1. Introduction	1
1.1. Motivation	1
1.1.1. Instrumentation	2
1.1.2. Data Acquisition	3
1.1.3. Technology	5
1.2. Overview	7
2. The High Acceptance Di-Electron Spectrometer (HADES)	8
2.1. The HADES Experiment	8
2.1.1. The Detector System	8
2.1.2. The Former Data Acquisition System	13
2.1.3. Experimental Program	15
2.2. The New Data Acquisition System	17
2.2.1. Requirements and Constraints	17
2.2.2. Electronics	21
2.2.3. The Complete DAQ System	27
3. DAQ Network Structure	30
3.1. Data Format and Network Channels	30
3.2. Network Layers	31
3.2.1. Physical Layer: Media Interfaces	33
3.2.2. Multiplexer	34
3.2.3. Link Layer: IO-Buffer	35
3.2.4. Transaction Layer: API	35
3.2.5. User Interface: Trigger, Data and Slow-Control	36
3.3. Special Features and Definitions	37
3.3.1. Network Addresses	37
3.3.2. Transfer Types and Network Transactions	38
3.3.3. Error- and Status Bitpattern	41

Contents

3.3.4. FOT Link Error Correction	42
3.4. Network Nodes	45
3.4.1. Endpoints	46
3.4.2. Network Hubs	47
4. The Trigger Sequence	52
4.1. Overview	52
4.2. LVL1 Trigger	53
4.3. Trigger Generation and Distribution	55
4.4. Trigger Handler	57
4.5. Trigger Interface	60
4.6. LVL1 Trigger Error and Status Information	61
5. The Read-out Chain	63
5.1. Read-out Data Format	63
5.2. Event Data Interface	64
5.3. CTS Read-out Request	67
5.4. Read-out Handler	67
5.5. Data Read-out Hub	68
5.6. Streaming API, Sub-Event Builder and Gigabit Ethernet Interface	72
5.7. The Event-Builders and Data Storage	73
5.8. Read-out Error and Status Information	73
6. The MDC Read-out System	76
6.1. MDC DAQ Upgrade	76
6.2. MDC Read-out Controller	79
6.2.1. Statistics and Monitoring	81
6.3. MDC OEP Data Format	82
6.4. Data Filtering and Reduction	83
7. Slow Control and Monitoring	85
7.1. The Slow Control Interface	85
7.2. Slow-Control Features	87
7.2.1. Start-up and Configuration	87
7.2.2. Monitoring and Online Statistics	87
7.2.3. Common Slow-Control Features	88
7.3. System Monitoring	88
8. DAQ Performance	91
8.1. Latency Measurements	91
8.2. Read-out Time and Dead-Time Estimations	94

8.3. Read-out Bandwidth	95
8.4. August 2011 Test Run	96
9. Conclusions, Synergies and Possible Improvements	103
9.1. Synergies: Employment in Other Experiments	104
9.2. Further Developments	104
9.2.1. Reduced Dead Times: Credits-based Trigger	105
9.2.2. Increased Bandwidth and Control Features: TrbNet v2	106
9.2.3. New Electronics	107
10. Kurzfassung	109
A. TrbNet Definitions	115
A.1. Network Addresses	115
A.2. Trigger Interface	118
A.3. Event Data Interface	123
A.4. Slow-Control Interface	125
A.5. Streaming API	129
B. Trigger Distribution	133
C. Slow Control Registers	134
C.1. Common Slow Control Registers	134
C.2. Hub Registers	138
C.3. MDC Configuration and Status Registers	141
D. Control and Monitoring Software	146
D.1. Control Software	146
D.2. Monitoring Software	147
Bibliography	156

List of Figures

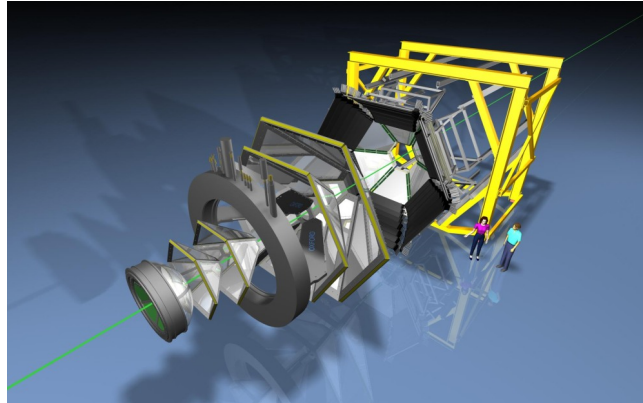
1.1. The QCD Phase Diagram	2
2.1. The HADES Spectrometer	9
2.2. Start Detector	10
2.3. RICH Detector	11
2.4. Pre-Shower Detector	12
2.5. RPC Detector	13
2.6. The Former HADES DAQ System	14
2.7. Di-Electron Production in Ar+KCl at 1.76 AGeV	16
2.8. Trigger and Read-out Board	23
2.9. MDC Optical Endpoint	24
2.10. pre-Shower Board	25
2.11. Central Trigger System Board	26
2.12. Network Hubs	26
2.13. The DAQ System	27
2.14. Network Setup	28
2.15. Ethernet Network Setup	28
3.1. The Network Layer Structure	32
3.2. The Data Flow Through Network Layers	32
3.3. The Links State Machine	34
3.4. The IO-Buf Handshake	36
3.5. The API Transaction Handling	37
3.6. Possible Dead-lock Between two Requests	39
3.7. The Error Correction Logic	43
3.8. The Error Detection State Machine	44
3.9. The Front-end Endpoint	46
3.10. The Hub	48
3.11. The Hub Logic	48
3.12. Data Merging in Hubs	49
3.13. The Hub IPU Logic	50
4.1. LVL1 Trigger Packets	53

List of Figures

4.2. The Trigger Handling Algorithm for Front-ends	60
5.1. HADES Data Format	64
5.2. The HADES Event Header	65
5.3. The HADES Sub-Event Header	65
5.4. Block Diagram: Data Handler	66
5.5. Read-out Request Sent by the CTS	67
5.6. Flowchart: Read-out Handler	68
5.7. Data generated by the Read-out Handler	69
5.8. Flowchart: Hub IPU Logic	70
5.9. IPU Data sent by hub in non-packing mode	71
5.10. IPU Data sent by hub in packing mode	71
5.11. The Streaming API	72
5.12. The Sub-Event Builder	73
6.1. MDC Read-out System	77
6.2. MDC FEE Schematics	77
6.3. MDC Front-end Logic Blocks	80
7.1. Hmon Monitoring and Visualization Tools	89
7.2. Hmon Monitoring Examples	89
7.3. Hmon Tactical Overview	90
8.1. LVL1 Trigger Latency	92
8.2. Retransmission Rates	97
8.3. MDC Data Reduction	98
8.4. Micro Spill Structure	99
8.5. Dead Times	99
8.6. Data Rate for Individual MDC Front-ends	101
9.1. The Trigger and Read-out Board version 4	108
10.1. Das HADES Spektrometer	110
10.2. Netzwerkaufbau	111
10.3. MDC Optical Endpoint	112
A.1. Timing Diagram: LVL1 Trigger Handler - Case 1	120
A.2. Timing Diagram: LVL1 Trigger Handler - Case 2	120
A.3. Timing Diagram: LVL1 Trigger Handler - Case 3	121
A.4. Timing Diagram: LVL1 Trigger Handler - Case 4	121
A.5. Timing Diagram: LVL1 Trigger Handler - Case 5	122
A.6. Timing Diagram: LVL1 Trigger Handler - Case 6	122

A.7. Timing Diagram: Event Data Interface	123
A.8. Timing Diagram: RegIO Data Bus Read Cycles	126
A.9. Timing Diagram: RegIO Data Bus Write Cycles	126
A.10. Timing Diagram: Streaming interface I	131
A.11. Timing Diagram: Streaming interface II	131
A.12. Timing Diagram: Streaming interface III	132
B.1. The CTS Trigger Distribution	133
D.1. Listing: Register Definition File	147
D.2. The Gigabit Ethernet Monitor	148
D.3. The Hub Monitor	149
D.4. The Logfile Monitor	149
D.5. Listing: Nettrace	150

1. Introduction



An artistic view of the HADES detector [1]

1.1. Motivation

The exploration of the phase diagram of strongly interacting matter is one of the main topics in modern nuclear physics. The current understanding of the different phases of strongly interacting matter is summarized in figure 1.1. At moderate temperatures and low net-baryon densities, quarks and gluons form bound states, the hadrons. At high temperatures a phase transition or cross-over from hadrons to free quarks and gluons, the Quark Gluon Plasma (QGP), a state of matter that was existing in the early universe, is predicted [2]. At high net-baryon densities that can be found in the interior of super-compact neutron stars, additional exotic phases of strongly interacting matter are expected [3]. The only possibility to study these extreme states of strongly interacting matter in the laboratory are relativistic heavy ion collisions. During the collision, a volume of highly compressed and heated matter is produced.

The density and temperature reached in nucleus-nucleus collisions depends on the size of the collision system and the kinetic energy of colliding particles. The high-energy collider experiments at LHC (Large Hadron Collider, CERN, Geneva) and RHIC (Brookhaven) focus on the region of high temperatures and low net-baryon densities. In the energy regime around 1 GeV per nucleon, fixed-target experiments such as at the GSI Helmholtz Centre for Heavy Ion Research in Darmstadt, Germany, study regions of high net-baryon density. An inter-

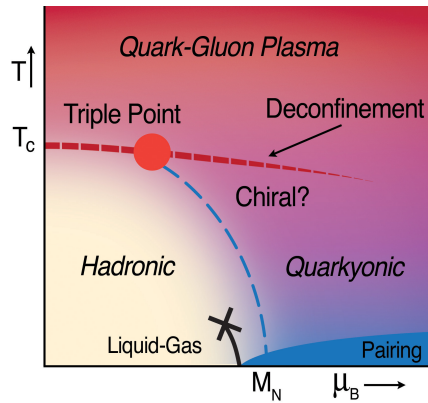


Figure 1.1.: The phase diagram of strongly interacting matter [4].

mediate region will be addressed at the FAIR accelerator complex (Darmstadt) or at NICA (JINR Dubna, Russia), currently under construction. In this region of moderate temperatures and finite baryo-chemical potential, the predicted first-order phase boundary and additional phases of nuclear matter are within reach.

Since a heavy-ion reaction only lasts for very short time ($\tau \approx 10^{-23} \text{ s} \approx 3 \text{ fm}/c$), it cannot be observed directly. To gain information about the properties of this matter under extreme conditions, an experiment has to detect particles emerging from the reaction zone. Ideal probes are particles that are produced and decay within the hot and dense phase of the collision. Such a probe are light vector mesons. These mesons, more precisely the ρ , ω and ϕ mesons, have a short life-time of less than 50 fm/c. Their daughter particles have to transport the information from the early stages of a nuclear collision until they can be detected by the experimental setup. In the ideal case, they do not undergo any further interactions with other particles. Hence, a leptonic final state is preferred over a hadronic one due to the fact that leptons interact only electromagnetically with the surrounding hadronic medium. The challenge of measuring the decay channel into leptons is that it is suppressed typically by four orders of magnitude compared to hadronic decays. In order to reconstruct the signal from such rare probes, a huge number of collisions has to be analyzed.

1.1.1. Instrumentation

A typical experimental instrument to record experimental data in nuclear and particle physics are spectrometers. The trajectory, momentum, charge and energy of each particle can be measured and used for particle identification. Due to the very different properties of the particles, a spectrometer consists of various layers of dedicated detectors. The required precision and granularity is defined by the aspects of the reactions the experiment aims to analyze.

The trajectory of a particle can be determined with a tracking system. Commonly chosen technologies are silicon micro-pattern detectors (e.g. CBM-STS [5]) or gas-filled detectors

(e.g. drift chambers, time projection chambers (TPC), straw tubes). These detectors are organized in several layers and the particle track can be reconstructed from the position of the hits in the individual layers. Another option are detectors that are able to record a track in three dimensions like Time Projection Chambers as used in the STAR¹ [6] and ALICE² [7] experiments. The tracking system can additionally identify the momentum of particles when placed inside or around a magnetic field that deflects all charged particles.

The velocity of particles is determined by a time-of-flight detector system that measures the arrival time of individual particles. Here, scintillator strips or resistive plate chambers provide the necessary high time resolution. Both types of detectors are used in the HADES setup and are explained in section 2.1.1. The energy of particles can be measured by calorimeters.

Many experiments base on a reliable identification of distinct particle species so that their setup is complemented with dedicated detectors like Cherenkov detectors (e.g. HADES RICH) for light particles or massive absorber blocks that filter all particles despite muons. Most detectors are blind for non-ionizing particles so that the detection of neutral particles is possible only if the neutrons and photons are converted to charged particles by nuclear or electromagnetic processes. This can take place in specialized detectors such as calorimeters or neutron detectors.

1.1.2. Data Acquisition

The electronics needed to operate and read out the detector systems are diverse and typically depend on the detector type. Typically, in a first stage analog electronics is used to amplify the small electric signals generated inside the sensitive part of the detector. Since the typical charge generated in, e.g. a gaseous detector, is less than 1 pC, these electronics have to be very sensitive and located closely to the detector to improve signal quality. The next processing stage is usually formed by a shaping and discriminating stage that modifies the raw signal such that it can be quantitatively evaluated. For example, the charge deposited in a detector over a time span of several hundred nanoseconds can be integrated in a first step. Subsequently, the resulting signal height can be converted to a signal length that can be measured in time-to-digital converters (TDC).

The next step of signal processing is the digitization of data, either in TDCs to measure the timing of the signal, or in an analog-to-digital converter (ADC) to measure signal heights. Some detectors require special converters, like direct charge-to-digital conversion (QDC). Silicon detectors, on the other hand often include the digitization circuits on the same chip as the sensitive area.

All sub-systems of a spectrometer have to be operated in a synchronous way to correlate hits from individual detectors to one common event. In most cases, this is accomplished by using a trigger system. A central instance determines, if a reaction took place and sends

¹STAR: Solenoidal Tracker At RHIC, BNL, Brookhaven

²ALICE: A Large Ion Collider Experiment, LHC, Cern

a synchronization signal to the sub-systems which start the read-out process for this event. The time window until the trigger decision has to be available is typically very short (less than 1 μ s) due to the characteristics of the front-end electronics. Hence, only simple and fast computations can be included in the generation of the signal. The method how this trigger is generated varies depending on the purpose of an experiment. A fixed-target experiment might use a dedicated detector that registers each beam particle passing through.

Since only few particles undergo an interaction within the target, this method yields in many empty events being read out so that more information needs to be included in the decision. Further raw signals from the detectors can be taken into account in this case. Dedicated hardware, e.g. analog adders can be used to generate a multiplicity signal, i.e. how many particles were detected to select only a subset of all events. Typically the most central collisions of two nuclei where a large number of particles is produced are of special interest. Digital electronics allows to run more advanced algorithms like defining regions of interest inside the full detector acceptance as used in the ALICE LVL1 trigger [8].

In many experiments, additional steps of event selection are added in further trigger levels. Due to the already reduced amount of data, a more detailed analysis of the data can be performed. In the ATLAS³ experiment located at the LHC, for example, the first level trigger reduces the primary interaction rate of 40 MHz to 75 kHz, the second level trigger cuts down the rate to 3 kHz and a third level selects events for full analysis at a rate of 200 Hz [9].

A rather new read-out concept is a so-called free-running DAQ: All channels of the detector are read out continuously and data is sent to a server farm. Only here data is combined to events and a first selection is made. Due to the large data rates that can be more than 500 GB/s, this set-up is only feasible with huge computing grids and for experiments running at high event rates like planned for CBM⁴ [10].

Another important step is transporting data from the front-ends to a central storage. The amount of data recorded with current spectrometers is typically in the order of 100 MB/s to 2 GB/s (e.g. PHENIX⁵ 1.8 GB/s [11], ATLAS 300 MB/s [9]). These data rates can not be handled by a single server, so server farms are used to process all data.

The computing needs are further increased by the data analysis which is the last step of a typical spectrometer experiment. Depending on the experiment, data is either analyzed in real-time, or stored for later analysis. During analysis, data from all sub-systems is combined and distinct data points are matched to reconstruct particles and their trajectories. Based on this information, conclusions on the physical properties of matter and particles under various conditions can be drawn.

³ATLAS: A Toroidal LHC Apparatus

⁴CBM: Compressed Baryonic Matter Experiment at FAIR

⁵PHENIX: Pioneering High Energy Nuclear Interactions eXperiment, BNL, Brookhaven

1.1.3. Technology

Despite analog signal processing electronics in the proximity of the sensors, a major part of data acquisition systems is based on digital logic. Even though the front-ends have to be adapted to specific needs, in many experiments it is possible to use a common structure of the data acquisition system for all sub-systems. The implementation of this digital part can be accomplished in various technologies.

One choice are micro-processors like DSPs⁶. Due to the need of high I/O band-width but comparably low computation performance, they can not easily be utilized in many systems. Full flexibility is provided by custom designed ASICs⁷. Here, all special requirements can be implemented without limitation from commercially available devices. In high-rate experiments, the radiation tolerance needed for all front-end electronics can usually not be provided by commercial devices so that ASICs are the only viable possibility. Due to their high offset costs and long development time compared to other, new technologies available since few years, the usage is not feasible for many experiments.

This third option are Field Programmable Gate Arrays (FPGA). FPGAs are integrated circuits that can be programmed to provide any logic functions. The structure consists of a matrix of several thousand basic blocks (slices) that can be connected with each other by a huge set of connection wires. Each slice consists of a small look-up-table with typically four or six inputs that generates one output signal based on any possible logic function on the input signals. It is complemented with a flip-flop as storage element. Additionally, most FPGA provide dedicated function blocks like blocks of memory, multipliers or even serializer-deserializer (SerDes) blocks that can be used to interface with serial data links.

The typical clock speeds reached with FPGA are in the order of 50 to 200 MHz. Although the clock speed is comparably slow, an FPGA can reach high over-all computation speeds since all functions run in parallel. This is especially true for algorithms that can be pipelined, i.e. functions that can be split into several steps that can be computed serially. Most data pre-processing and data transport operations fall into this category.

The functionality to be implemented in an FPGA is described using a dedicated language, VHDL (Very high speed integrated circuit Hardware Description Language) [12]. VHDL is, as stated in its name, not a programming language as used to define a program for a micro-processor. It is rather a language to describe and simulate of logic circuitries. One special feature of VHDL is that it can be used to represent logic functions that are implementable in an FPGA by a dedicated tool chain. The configuration of the FPGAs⁸ can be stored in a small flash memory.

The additional electronic devices on the front-end boards have to be chosen to fit to the

⁶DSP: Digital Signal Processor

⁷ASIC: Application-Specific Integrated Circuit

⁸In a strict sense “configuration” is the only term applicable for the logic design loaded into an FPGA. In this work I use also “design” and “firmware” as synonyms describing the internal configuration of an FPGA.

requirements of each detector. Hence, no common selection for all boards is possible. Nevertheless, complexity of the system and development time can be greatly reduced by reusing the same hardware platforms throughout the whole system. As an example, the LHCb experiment uses detector-specific front-end cards that send their data to one common read-out platform, “TELL1” [13], that pre-processes and forward data to the DAQ system. The board consists of five FPGAs that share the tasks of data processing and communication with the DAQ system. The front-end specific interface can be added to the platform with AddOn cards.

Data Transport

A challenge in all modern nuclear and particle physics experiments is the design of the data transport network. Besides the band-width for data transport and error tolerance, the network latency is of particular concern for triggered DAQ systems. In many cases, no commercial solutions can be found in view of dictated design aspects like compatibility or radiation hardness. In these cases, a protocol has to be custom designed.

On the other hand, the data transport between the detector system and the computer farm can be based on a widely available standard, e.g. Gigabit Ethernet (GbE). In this part of the network no special timing requirements have to be fulfilled so that typical latencies of several hundreded microseconds can be accepted. Data is transported in a single direction only and no direct feedback from servers to the DAQ system has to be foreseen.

The LHCb experiment uses also an approach with two data networks [13]: All fast, timing critical information such as triggers, control messages and synchronization are handled by a dedicated data bus [14] while all data transport from the pre-processing modules is covered by a GbE installation. Even though the total data rate of 35 GByte/s from 300 read-out nodes has to be routed to a farm of 500 data processing servers, a single Ethernet network is able to handle all traffic [15]. This demonstrates the scalability of a commercial network for experimental setups with very high data rates.

The data transmission hardware to transport data between different modules of the system can be based on two basic technologies: electrical signal transmission on copper wires or optical transmission using glass fibers. Compared to electrical signals, optical data transmission has several advantages. First of all, optical signals do not produce any electromagnetic noise which might have an influence on the detector. Vice versa, the optical signals themselves can not be distorted by external noise sources. The two transceivers used on both ends of a connection are the only places in which noise can have an influence on data integrity⁹.

Optical fibers can be operated at data rates of more than 5 Gbit/s per link while electrical signals usually have to be split over several shielded wires and differential transmission standards have to be employed. The optical fibers have a diameter of 2 mm and are much smaller than electrical wires capable of the same data rates, fitting to tight space constraints

⁹This was observed in the HADES network and taken account for as shown in section 3.3.4.

as found in many experiments. These differences characterize optical fibers as the technology of choice for a modern DAQ system.

1.2. Overview

The main goal of this work is to develop, implement and commission the new data acquisition system for the HADES experiment.

In chapter 2 the HADES experiment is described and the requirements on the new data acquisition system are determined. Based on these requirements, it is discussed which technologies are chosen and the electronics that have been developed are described. Finally, an overview over the complete data acquisition system setup is given.

A network protocol that matches all requirements and provides a reliable data transmission between diverse electronic platforms distributed in the detector setup was built up. A brief description on all network features is given in chapter 3.

The following chapters are devoted to three major blocks contained in the data acquisition system: Trigger generation and distribution (chapter 4), data read-out from front-ends to data servers (chapter 5) as well as the slow control system (chapter 7). A special focus is put on the front-end control system of the MDC sub-system which is described in detail in chapter 6.

Finally, performance values obtained during commissioning runs are shown in chapter 8. Furthermore, additional developments and possibilities for future upgrades of the system are analyzed therein.

2. The High Acceptance Di-Electron Spectrometer (HADES)

2.1. The HADES Experiment

The High Acceptance Di-Electron Spectrometer (HADES) at GSI is one of the experiments built to study properties of matter at finite net-baryon density. Being designed as a general purpose spectrometer with an emphasis on di-electron measurements in both elementary and heavy ion collisions, the HADES detector has to fulfill diverse requirements.

In a central Au+Au collision at energies of 1 to 2 AGeV, about 200 charged particles are emitted from the reaction zone. The reconstruction of their trajectories requires high granularity of the tracking and particle identification system. As mentioned before, the leptonic decays of vector mesons are strongly suppressed compared to the hadronic decay channels. This calls for both a high rate capability of the whole detector system as well as an efficient particle identification to apply filters to select the events of interest. Furthermore, a high geometric acceptance is required to achieve a high probability to detect lepton pairs. The geometrical acceptance of the HADES experiment covers polar angles between 18° and 85° and provides an almost complete coverage in the azimuthal angle.

The HADES detector setup comprises of different detector types. The central part of the experimental setup is formed by the tracking system consisting of four layers of multi-wire drift chambers combined with a magnetic field. Charged particles can be identified by time-of-flight measurements in a Time-of-Flight (TOF) wall and Resistive Plate Chamber (RPC) detectors. The identification of electrons is further improved by a Ring-Imaging Cherenkov detector (RICH) and a pre-Shower detector. The hadrons not interacting during a collision, so-called spectators, are detected by a Forward Wall Hodoscope that covers polar angles below 7° . The full setup is shown in the cross-section of the HADES detector in figure 2.1. All detectors are described in the following sections. More detailed information can be found in [16].

2.1.1. The Detector System

Start and Veto Detectors

The hadron identification based on time-of-flight measurements relies on a precise determination of the time of the primary collision. Additionally, the triggered read-out system of

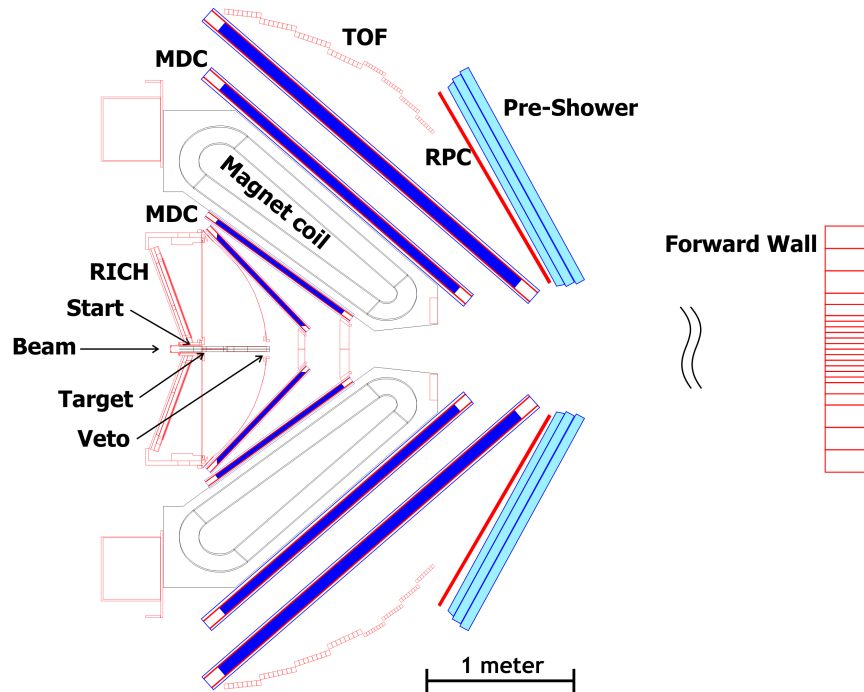


Figure 2.1.: The HADES spectrometer consists of several independent detector systems. Tracking of particles is provided by four layers of multi-wire drift chambers and a magnetic field. Particle Identification is done by the time-of-flight and RPC detectors in combination with the electron identification provided by RICH and pre-Shower detector. A start detector measures the reference time of an interaction and the Forward Wall (positioned 7 m downstream from the target, not to scale) detects all particles at low polar angles.

HADES requires a start signal once a reaction took place.

These two requirements are met by two diamond detectors [17], Start and Veto, mounted in front of and downstream the target behind the RICH detector, respectively. The Start Detector placed in front of the target generates a signal for each beam particle before it enters the target region.

The target is usually designed for an interaction probability of about 1%. Thus, most particles traverse the target region without any interaction. These particles are registered by the Veto detector. Both detectors are segmented into eight individual channels as shown in figure 2.2. Hence, the position of the beam can be monitored accurately and re-adjusted if necessary.

Based on the hit information from both detectors, the central trigger system is able to distinguish the interacting particles from all other beam projectiles and generates triggers for actual collision events only. The timing measurement is performed by fast Time-to-Digital converters (TDC) [18] with a binning of 25 ps while the intrinsic time resolution of the

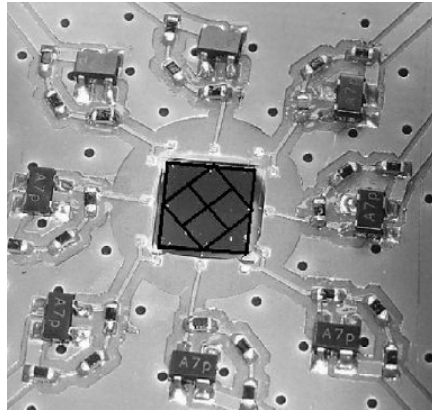


Figure 2.2.: The Start Detector is composed of a 8-fold segmented diamond detector placed in front of the target region. A similar diamond is used as Veto detector [17]. The electronics surrounding the detector provide a first amplification of the small signals generated inside the diamond by particles passing through.

detector is about 30 ps for heavy ions [19].

Electron Identification Detectors

The Ring Imaging Cherenkov Detector (RICH) [20] is used for electron identification and placed around the target region. It consists of a large volume filled with C_4F_{10} as radiator gas. It has a refraction index of $n = 1.0015$ [21] so that all particles with a Lorentz factor of $\gamma > 18.3$ produce Cherenkov radiation in the detector. This corresponds to a threshold energy of $9.3 \text{ MeV}/c^2$ for electrons and $2.6 \text{ GeV}/c^2$ for pions. Hence, all electrons but only very few high energetic hadrons generate Cherenkov radiation resulting in a high pion suppression factor in the order of 10^3 .

The produced light cone is reflected by a mirror onto a plane of multi-wire proportional chambers residing in a separated gas volume. Here, the ultra-violet photons are converted to electrical signals on a CsI coated pad plane, amplified and read out using over 4,700 individual channels per sector. A cross section of the detector is shown in figure 2.3.

The second stage for electron identification is introduced by the pre-Shower detector [22], mounted as the outermost part of the spectrometer at polar angles between 18° and 44° . It consists of three layers of wire chambers with lead plates mounted in between.

All charged particles traversing the lead plates lose energy by emitting bremsstrahlung. These bremsstrahlung photons trigger an electromagnetic shower which is detected in the wire chambers as depicted in figure 2.4. Hadrons produce only few additional particles, while electrons produce many orders of magnitude more particles. Hence, electrons and hadrons can be distinguished by comparing the charge entry in all three layers of the detector for each track.

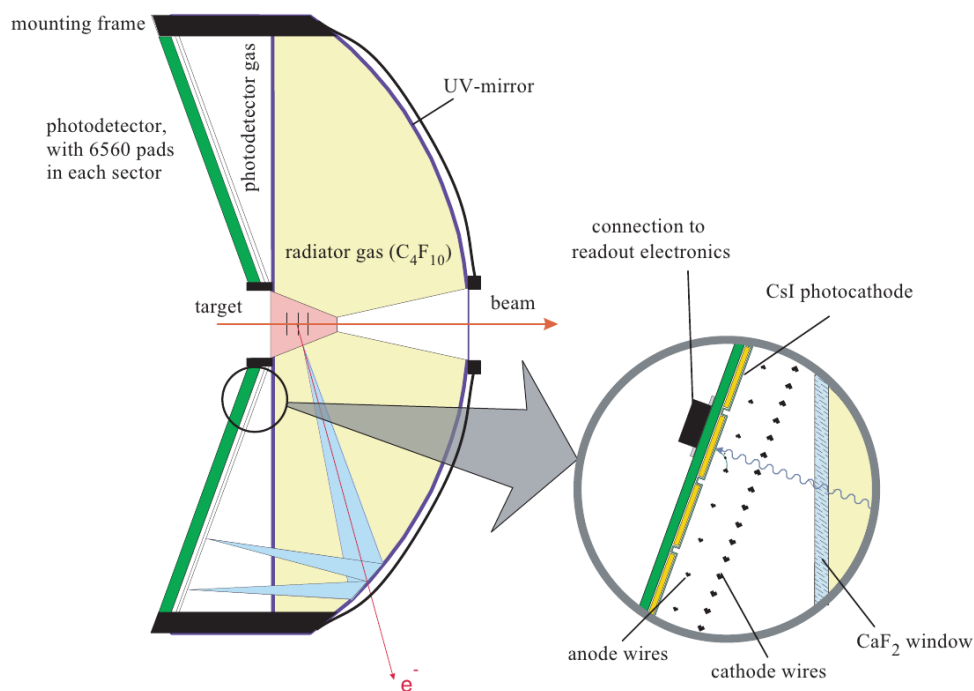


Figure 2.3.: The Ring Imaging Cherenkov Detector. [21]

In the wire chambers, all charged particles produce electron - ion pairs in the detector gas. An electric field separates these charges. The electrons are amplified in the high-field region of the anode wires and influence a signal in the pad planes which is further amplified and read-out. The detector consists of a grid of 942 read-out pads per detector plane and sector.

The Tracking System

The tracking of charged particles is provided by a set of multi-wire drift chambers (MDC) [23] and a magnet [24]. In each sector of the detector, four chambers are mounted. Two are placed in front of the toroidal superconducting magnet and two behind. The particle tracks are bent by the magnetic field so that their momentum over charge ratio can be determined.

In each chamber, the charged particle passes through a gas filled volume, producing electron-ion pairs by electromagnetic interaction with the shell electrons of the gas molecules. These charges are amplified, transported to sensor wires and read-out using the same mechanism as in the Shower detector.

Each MDC chamber is composed six layers of sensitive wire planes with different wire orientation to provide two-dimensional coordinate information in each chamber.¹⁰ The particle trajectory can be found by searching for the crossing point of fired detector cells. The drift velocity of electrons inside the gas volume is about 4 cm/ μ s. Hence, a more precise po-

¹⁰The angles are $\pm 40^\circ$, $\pm 20^\circ$ and 0° (two layers)

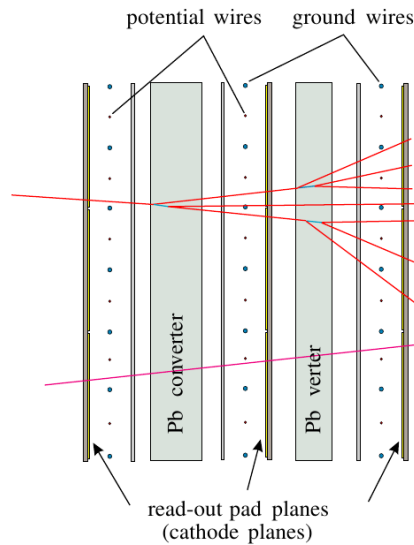


Figure 2.4.: A cross section of the pre-Shower detector. Charged particles are detected by three layers of wire chambers. Electrons produce a shower of charged particles due to bremsstrahlung conversion in lead plates. [22]

sition information is given by the arrival time of the electron cloud at the wire. The resulting spatial resolution is in the order of 100 μm .

The total number of channels inside the detector is 25,964¹¹. Typically, each charged particle causes signals in 40 drift cells [25]. Hence, the MDC forms the biggest sub-system of the HADES detector not by means of individual channels but by active channels per event.

The Time-of-Flight System

The time-of-flight wall (TOF) [26] provides trigger information and particle identification for hadrons within a polar angle between 44° and 88° . Each sector consists of 64 scintillator bars organized in 8 modules. A charged particle passing through the detector produces light that is guided to both ends of the rod. Here, the light is amplified by photomultiplier tubes.

From the measurement of both, time and signal amplitude¹², the precise arrival time, the position on the bar and the energy deposited in the scintillator can be deduced. The time resolution of the detector is 150 ps [16].

The lower polar angles between 18° and 44° are instrumented with resistive plate chambers (RPC) [27]. It replaces the Tofino detector which was in use until 2007 and provided four channels per sector only and a time resolution of 420 ps [16]. The low granularity and spatial

¹¹The total number of read-out channels is 27,648, but not all channels are actually connected to the detector.

¹²First, a conversion circuitry is used to convert the signal amplitude to a signal length, then the signal is measured by a TDC

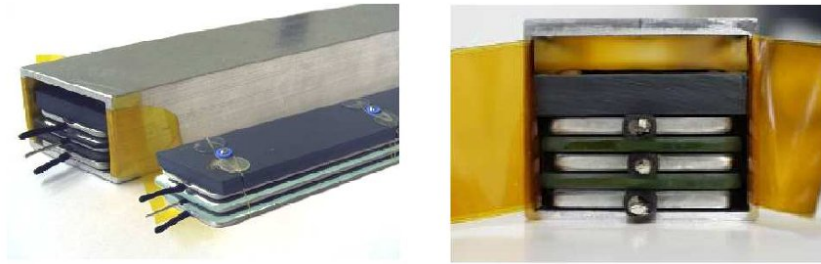


Figure 2.5.: One cell of the RPC detector. It consists of three aluminium electrodes separated by two glass plates. The inner aluminum plate is connected to a high voltage power supply. Both ends of the cell are connected to read-out circuits. [29]

resolution is not sufficient for measurements in heavy ion collisions due to the high particle multiplicity. Compared to that, the new RPC detector features 372 channels per sector and a time resolution of 75 ps [28]. One cell of the RPC detector is shown in figure 2.5.

The Forward Hodoscope

The Forward Wall is a hodoscope consisting of a total of 287 scintillator blocks. Placed 7 m behind the target the detector covers polar angles below 7° . The granularity of detector cells varies from small cells ($4 \text{ cm} \times 4 \text{ cm}$) near the beam axis to large cells ($16 \text{ cm} \times 16 \text{ cm}$) at larger polar angles according to the expected particle flux. The forward wall has been installed in 2007 and provides the measurement of low p_t particles such as spectators that do not interact within the target.

2.1.2. The Former Data Acquisition System

The original HADES data acquisition system was intended to operate at the highest beam intensities available at SIS18 of 10^8 particles per second in combination with an interaction rate of about 1% in the target. In order to reduce the high event rate to a reasonable amount of data, HADES was designed with a three level trigger system [30].

The first stage was a multiplicity trigger generated by a given amount of particles in the time-of-flight wall. Depending on the physics case, the trigger rate can be reduced to about 10% of the reaction rate by selecting events with high particle multiplicity only. The second stage was built to perform a first data analysis to identify events with electron candidates. Here, the so-called Image Processing Units (IPU) searched for electron signatures in RICH and Shower detectors. Combined with hit information in the time-of-flight wall a rough momentum estimation and the charge of all electron candidates could be determined in the Matching Unit (MU). Subsequently, selection criteria for interesting events were applied. This trigger stage was supposed to be able to reduce the trigger rate by two orders of magnitude. The result was an event rate of 1 kHz and about 15 MByte/s data volume. A third

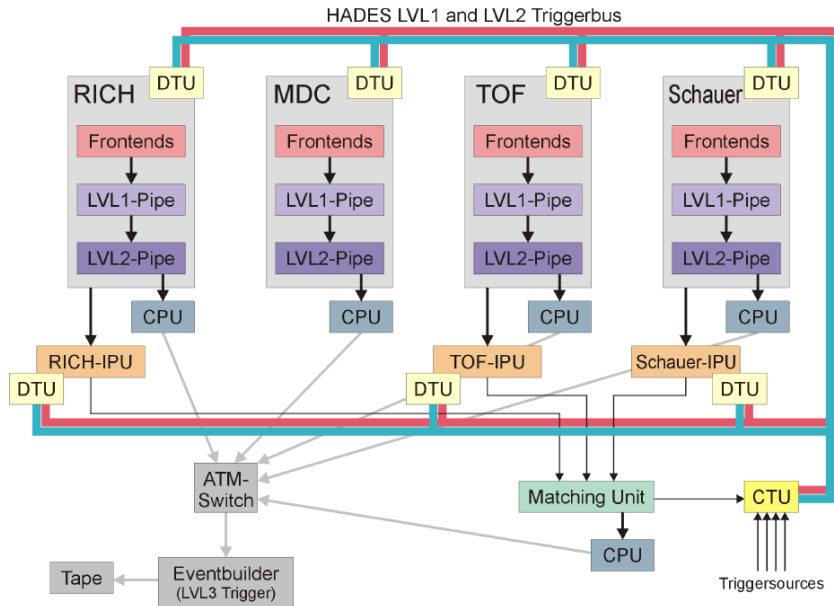


Figure 2.6.: The former HADES Data Acquisition System used until 2008. Triggers were distributed by the CTU to detector specific trigger modules (DTU). Data were stored in several pipeline stages until the trigger decision was available. The data transport backbone was formed by VME crates and an ATM switch [30].

trigger level was planned to provide higher level analyses and to further reduce the amount of data recorded but was never realized.

Trigger signals were distributed by the Central Trigger Unit (CTU) to detector specific trigger controllers (Detector Trigger Unit) which send read-out control signals to the frontends. All data and control signals were transported over parallel buses, the interconnect was provided by flat cables. Data were stored in several pipeline stages until the trigger decision was available.

All data processing modules were mounted in VME crates that used the integrated data busses for data transport. A CPU received this data and forwarded it to a server (Eventbuilder) that stored the data on tape. The data paths in the LVL2 and later stages were designed to transport the predicted amount of data. In combination with the planned event rates at that time, this is only sufficient if the expected suppression factor in the two first trigger stages is about 1000. The developed trigger algorithms were not as efficient as expected, so that this aim was not reached. The main reasons were the insufficient purity of electron candidates and the quality of rings produced in the RICH detector as well as the many electrons resulting from conversion of photons inside the detectors. The highest suppression factor in the LVL2 stage reached was 3 instead of a factor 100 projected. As a result, the LVL2 stage was skipped in later experiments and only multiplicity triggers were implemented in

Year	System	Energy	Events (rec./LVL1)
2002	C + C	2 AGeV	$0.2 \times 10^9 / 0.6 \times 10^9$ [32]
2004	C + C	1 AGeV	$0.6 \times 10^9 / 1.1 \times 10^9$ [31]
2005	Ar + KCl	1.765 AGeV	$0.9 \times 10^9 / 2.2 \times 10^9$ [34]
2006	p + p	1.25 GeV	0.9×10^9
2007	p + d	1.25 GeV	2.0×10^9
2007	p + p	3.5 GeV	1.1×10^9
2008	p + Nb	3.5 GeV	4.2×10^9

Table 2.1.: Between 2002 and 2008 several experiments with different collision systems and beam energies have been conducted using the HADES spectrometer. The number of events recorded is shown, in experiments in which a LVL2 trigger was used, also the number of LVL1 triggers is given.

the LVL1 stage. Due to the width of data paths high event rates can not be achieved and the total rate was limited to approximately 10 kHz for light systems and estimated 0.7 kHz for Au+Au collisions.

2.1.3. Experimental Program

Between 2002 and 2008, several experimental runs were performed with the HADES spectrometer. Various collision systems at different beam energies have been investigated. The full list of experiments is given in table 2.1, test runs are not listed. low

The first runs using carbon induced reactions at 1 AGeV [31] and 2 AGeV [32] respectively were performed to confirm data that has been taken by the DLS collaboration [33]. Next, the medium sized system Ar+KCl [34] was measured. A comparison to measurements in elementary reactions conducted in 2006 and 2007 allows to draw conclusions on the origin of virtual photons in elementary collisions. Lastly, an experiment with cold nuclear matter at saturation density was made in 2008 with p+Nb reactions.

The di-electron production measured in the two C+C experiments showed a significant excess in the mass region between $150 \text{ MeV}/c^2$ and $500 \text{ MeV}/c^2$ compared to all theoretical models available at that time. As a comparison the p+p and n+p experiments were done. These showed that the enhancement in C+C collisions can be explained by an overlay of elementary collisions [35].

The next heavy-ion system measured with the HADES experiment was Ar+KCl with about 40 particles per nucleus. Here, an excess in di-electron production above the value extrapolated from elementary collisions was found [34]. In the medium mass region the enhancement factor is about 2 – 3 as shown in figure 2.7.

From these results, several questions arise: How does the excess change when going to

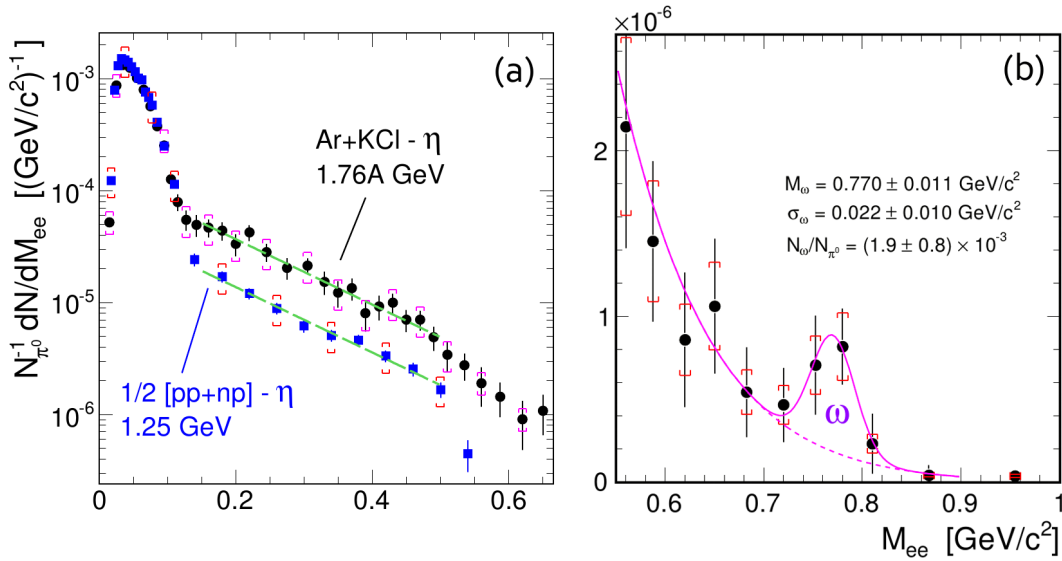


Figure 2.7.: Exemplary results for di-electron production in Ar+KCl at 1.76 AGeV [34]. Figure (a): An excess over the up-scaled production rates in elementary reaction of about 2 – 3 was found. Figure (b): A zoom on the ω pole mass region. In both figures the quite low statistics are visible through error bars. These are in the order of 50% in the ω region.

larger systems such as Au+Au? How does the excess scale with energy? Is there a difference between central Ar+KCl and peripheral Au+Au collisions with the same number of participants? The first question can be addressed at the current accelerator complex at GSI which is able to deliver any particle beam, including Au beams up to 1.25 AGeV. Higher energies of up to 10 AGeV can be achieved at the FAIR accelerator SIS100 which is currently under construction.

The observed excess can be further investigated by multi-differential analyses that divide data into several classes depending on various observables. E.g. a possible anisotropy in the angular distribution could point to non-thermal production mechanisms. The statistical errors indicated in figure 2.7 as vertical bars demonstrate that a multi-differential analysis is not feasible with data currently available due to the comparably low statistics. Already without further differentiation, the error on the count rates is in the order of 50% in the vector meson region. Here, higher statistics help to improve the extracted production yields and allows for a better background determination. This is especially important for ρ meson measurements due to its very broad shape.

Future Experiments

During the next years several heavy ion experiments are planned. The series will start in 2012 with Au+Au reactions at the highest beam energy available at SIS18, i.e. 1.25 AGeV¹³. In the following years experiments with other heavy systems such as Nickel and Silver are scheduled. A measurement of pion induced reactions is planned as well. Here, a secondary pion beam is generated in a production target placed about 20 m upstream from the HADES experiment target.

For a second set of experiments HADES will be moved to the CBM¹⁴ cave at the SIS100 accelerator of the FAIR facility which will be completed in 2017. The new accelerator is able to provide particles with energies of up to 11 AGeV for heavy ions and 30 GeV for proton beams. HADES experiments with various systems at energies between 4 AGeV and 10 AGeV are planned [37].

Almost all proposed experiments can not be done using the old HADES data acquisition system due to performance reasons. The system was able to reach about 10 kHz for p+p collisions and 3 kHz during the Ar+KCl run. In the latter case the limiting factor was the total data throughput which was limited to about 15 MByte/s [16]. For heavy ion experiments the number of particles produced and following the amount of data per event scales linearly with the number of participating nucleons in the collision. Hence, a central Au+Au collision (197 nucleons each) produces about five times more particles than in Ar+KCl (40 nucleons each).

Since the data rate limit can not be overcome without major changes in the data acquisition system, the achievable event rate drops to about 700 Hz. As experienced in former experiments, the integrated number of events recorded during a full experiment is a factor 4 lower due to the acceleration cycle of the synchrotron, failures in one of the systems or parallel running experiments. This rate results in less than 0.25×10^9 events collected during a typical four-week experimental run which is not sufficient for a detailed analysis.

The HADES detectors were built to withstand interaction rates of up to 1×10^6 and achieve read-out rates well above 50 kHz. These rates can be achieved by a complete upgrade of the now already ten year old electronics. This upgrade, including electronics and data transport protocol is the main topic of this work.

2.2. The New Data Acquisition System

2.2.1. Requirements and Constraints

Data Rates

The most central design criteria of every data acquisition system are the data and event rates

¹³The theoretical limit of SIS-18 with a magnetic rigidity of 18.5 Tm is an energy of 1.5 AGeV for Au⁷⁹⁺ but needs a two-step acceleration with intermediate beam cooling in the Experimental Storage Ring (ESR) and has a limited duty cycle and beam intensity [36].

¹⁴CBM: Compressed Baryonic Matter

Year	System
2012	Au+Au at 1.25 AGeV
2013 – 2015 (SIS18)	p+ π , Ni+Ni, Ag+Ag at up to 2 AGeV
2017 – (FAIR)	p+p, C+C, Ca+Ca, Ni+Ni at 4, 8 and 10 AGeV [37]

Table 2.2.: Planned experiments with the HADES detector include runs at the current SIS18 accelerator as well as experiments at the upcoming FAIR facility

that are to be achieved. In an ideal case, the event rate is limited by the detectors but not the DAQ system. First estimates for the total data rates to be expected can be obtained from simulated Au+Au events. Depending on the centrality of the collision, the number of particles produced varies. Here, a selection of 25% most central events was assumed [38]. For this centrality class at energies reached at the SIS-18 accelerator, on average 110 charged particles are emitted within the acceptance of the HADES experiment. For the most central collisions, the number can be up to 140 particles per event.

In the TOF and RPC detector, each charged particle generates two signals in the detector¹⁵. For each signal, two values are measured, corresponding to the time and amplitude. For each hit in a drift cell, the time when the signal crosses the threshold and the time above the threshold is recorded. The drift chambers consist of 24 stacked layers of sensitive volumes which the particle has to pass through, resulting in a large number of cells delivering data.

In case of the RICH and Shower detectors the total number of fired channels is extrapolated to the desired particle multiplicity from earlier measurements with smaller collision systems. A summary of the expected hit occupancies is given in table 2.3. For all detectors combined, about 6,000 channels of the detector are expected to deliver data per event. For all detectors, data from each channel is encoded in one 4-Byte data word resulting in a data size of 25 kByte of data per event. Additional data headers contain further addresses for the individual subsystems summing up to a total event size of 27 kByte per central Au+Au event.

The design event rate for heavy ion collisions is 20 kHz, so that the total data rate will be about 550 MByte/s. Typically, not only central but also a fraction of minimum-bias events will be recorded. Assuming a contribution of 50% minimum-bias events [38] with an average event size of 14 kByte/s, the final design data rate is reduced to 410 MByte/s. This rate is not necessarily higher than the average data rate since all front-ends will contain data buffers that balance short-term fluctuations in the event rate.

This data has to be transported to the server farm located 20 m far from the experiment outside of the cave. Here, data has to be distributed to several servers. An industry-standard technology using off-the-shelf components should be employed to guarantee high compati-

¹⁵Each particle hits one detector cell which has read-out electronics on two sides to determine precise position information.

System	Channels	Occupancy	Words/Evt	kB/Evt	MB/s @ 20kHz
MDC	25964	0.17	4500	18.0	360
RICH	28272	0.015	430	1.7	35
TOF	768	0.15	150 ¹	0.6	12
RPC	2232	0.07	510 ¹	2.2	44
Shower	16956	0.025	500	2.0	40
Others	316		200	0.8	16
Headers			500	2.0	40
Total	74504		6800	27 kB/evt	550 MB/s

Table 2.3.: Estimated data rates for simulated 20 kHz central Au-Au collisions [25].⁽¹⁾ including the measured reference times.

bility and the possibility of future upgrades.

Trigger Rate and Dead Times

The HADES detector was designed as a triggered experiment. This read-out scheme was kept also during the upgrade program for two reasons. First, the required event rates are reachable with the common triggered architecture. Second and more important, a change to a different read-out scheme would require an almost complete exchange of all front-end electronics since most of them must be operated in a triggered mode with variable dead-times.

At 20 kHz event rate, the mean time between two recorded events is 50 μ s. Since the particle beam delivered from the accelerator is not uniformly distributed but follows a statistical process, the dead-time of the detector system has to be significantly lower. Furthermore, there are additional fluctuations in beam intensity caused by the extraction process (compare also figure 8.4). To be able to record a high fraction of all events, the dead-time has to be restricted to a much shorter time, i.e. ideally 15 μ s, corresponding to an event rate of 66 kHz uniformly distributed.

During this time, the trigger information has to be transported to the front-ends and the busy-release (see section 4.1) has to be transported back. Additional time is needed for both the CTS to generate the trigger signal and the front-end to process the event until the busy-release can be sent. Hence, the data transmission latency from the central systems to front-ends and vice-versa is limited to less than 7 μ s.

On short time-scales, both the trigger rate and the amount of data produced with one event are not constant but varying within a wide range. To accommodate for these changes, the front-end modules should contain a reasonably sized data buffer to even out these fluctuations and lower the peak data rate for read-out.

All sub-systems rely on precise time information for each event for measurement. Conse-

Chapter 2. The High Acceptance Di-Electron Spectrometer (HADES)

quently, a central unit has to send a reference time signal with low time jitter. This signal is generated from analog signals from various detector channels that are combined to detect if an event took place. Here, the trigger decision is to be implemented in an FPGA in order to have a flexible setup that can be optimized during a run if necessary.

In the next step, the reference time signal has to be distributed to all detectors with a very high timing precision. The jitter must be substantially below the intrinsic timing resolution of the detectors which is less than 30 ps in case of diamond detectors [17]. Additionally, the signal has to arrive at the front-ends within a narrow time window after an event took place. E.g. the pre-Shower detector has to be supplied with the signal within less than 500 ns [39]. This time window is given by the analog signal processing that has a peaking time of 500 ns and needs to trigger a sample-and-hold stage to keep the signal until digitization can take place.

Data Transport

The demands on data rates and especially data transport latencies in the HADES DAQ require the use of a dedicated network protocol. Other, commercially available protocols were evaluated but often rely on fixed hardware components, are not platform independent or are not suitable due to space constraints. The protocol has to be able to handle trigger information, data read-out as well as control and monitoring functions on one physical connection. Here, the three types of information have to have different priorities assigned and the interference between them has to be kept at a minimum. Hence, a special network protocol, TrbNet, has been developed [40]. An overview of the features of TrbNet is given in chapter 3.

The data transport between the detector and the server farm is based on commercial Gigabit Ethernet solutions. One GbE link is able to transport about 110 MByte/s only, but the total data rate of 400 MByte/s in the HADES detector can easily be handled by splitting the data stream over several connections. The central routing of data streams to different servers is done by a 10-GbE network switch. Additional switches are used to combine data streams from several subsystems onto one GbE connection. These switches feature both optical and copper connections and operate as media converters between the optical signals sent by the DAQ electronics and the copper cables used to transport data to the server room.

The full HADES data rate can not easily be handled by a single software core. Splitting the data stream to eight or more data servers reduces the data rate to 50 MByte/s and can be handled by one software process. Additionally, the data can be written to a single hard drive without data losses. Modern multi-core processors are able to handle several of these processes on one machine so that the amount of servers is reduced to two to four. Again, a normal 10 GbE switch can manage the distribution of data to different physical servers.

Further Constraints

Constraints on the amount of space for both the front-end electronics and cables apply espe-

Parameter	Description	
Data rate	400 MByte/s	sustained data rate
Dead time	15 μ s	average dead time per event (Au-Au)
Latency	< 500 ns	delay on reference time distribution
	< 7 μ s	for trigger information distribution
Front-ends	500	front-end read-out controllers
Distance	40 m	endpoint-to-endpoint in main DAQ network

Table 2.4.: Summary of all main requirements on the HADES data acquisition system.

cially on the read-out for the two innermost MDC planes. The front-end electronics need to be placed on the frames of each chamber and outside of the acceptance of the detector. The free space for the read-out controllers are about 4 cm times 5 cm only. Additionally, all data transmission and power supply cables have to run through a small gap. The technology used has to be selected to minimize the electromagnetic influence on the detector to reduce the background noise in the system.

2.2.2. Electronics

The HADES DAQ upgrade addresses two major parts: data transmission and control systems. The third part of the DAQ system, front-end electronics and analog signal processing will mainly be based on the electronics used in the existing setup and is not within the scope of this work.

The read-out controllers for the HADES front-end electronics need to fulfill diverse tasks. The trigger and busy-release architecture of the DAQ system puts hard real-time constraints on the operation. Receiving data from the digitization circuits usually involves data streams running at speeds of 100 MHz or more and using custom communication protocols. The number of required I/O lines per read-out controller is above 100 using varying signaling standards. Field Programmable Gate Arrays (FPGA) provide the necessary flexibility in programming and amount and type of input and output signals along with highly parallel operation. The form factor of available devices also fits to the space constraints found for the MDC read-out controllers.

All data transmission will be based on bi-directional optical links. The use of industry standard transceiver modules guarantees the wide availability of devices and later upgradability of the system. Here the SFP (Small Form-factor Pluggable) standard provides the interface to different plug-in modules for data transmission. Besides several types of transceivers for different purposes and speeds, also adapters for different types of electric cables are available. In the HADES setup mostly Optoway SPM-8100WG SFP transceivers [41] that are capable of data transmission at speeds of up to 4.25 Gbit/s and line lengths of up to 500 m

are used. The MDC read-out controller can not house an SFP connector due to its size. Here the Firecomms FDL-300I FOT transceiver [42] was found to provide a good compromise between foot-print size and communication speed (250 MBit/s).

Parallel to data transmission, all detectors have to be supplied with a reference time signal. The required time resolution of few ten picoseconds can not be provided by any kind of data transported on the optical fibers within the DAQ network. Hence, a dedicated signal, based on the differential PECL or LVDS signaling standards has to be distributed to all subsystems.

For the HADES DAQ system, individual front-end electronics had to be developed for each subsystem depending on their respective requirements. Nevertheless, the whole read-out system is based on the same electronic building blocks. As described in section 1.1.3, all boards employ one of a small set of different FPGA types and feature one of two optical transceiver types for data communication. The FPGA designs are stored in a Flash memory that can be reprogrammed via the DAQ network so that it can be upgraded without physical access to the board. The network relies on the FPGAs being identifiable in the network. This information can not be hard-coded or provided by discrete switches on the boards due to the high number of individual platforms. Thus, a dedicated chip (Dallas DS1820 1-wire temperature sensor [43]) is placed on each board that provides a unique ID to the FPGAs.

These similarities of all boards make it possible to reuse the same VHDL code basis on all FPGA which greatly reduces development time and also facilitates debugging of the complete DAQ system. The boards shown below have been developed in a cooperation of several institutes, in particular at GSI in Darmstadt, TU Munich and the University of Krakow.

TOF, RPC, Forward Wall

The complete time-of-flight wall and the forward hodoscope are read out by the Trigger and Read-out Board (TRBv2 [44], see figure 2.8). The front-end electronics are formed by detector specific boards such as the TOF-AddOn which contains charge-to-width converters, amplifiers and signal shapers. The TRB is equipped with a Xilinx Virtex4-FX40 FPGA [45] that controls a 2 GBit/s optical link. A Texas Instruments TLK-2501 chip [46] generates the serial data stream since the FPGA does not contain Serdes elements. The front-ends are connected to 128 TDC channels provided by 4 HPTDC [47] chips. These provide a binning of 100 ps (40 ps resolution) in standard mode and 25 ps binning in a high-precision mode with 8 channels.

The TRBv2 was designed as a multi-purpose read-out platform and is also able to operate in a stand-alone mode. An Etrax FS embedded Linux system [48] acts as an interface between the FPGA and an Ethernet connection. It can configure all devices on the TRB and act as read-out and slow-control interface if desired. Additionally, a TigerSHARC DSP is foreseen to provide data processing features but is not used in the HADES setup.

On its backside the TRB provides an AddOn connector that allows to connect additional electronics that provide specific features. For example, the TOF-AddOn makes uses of this

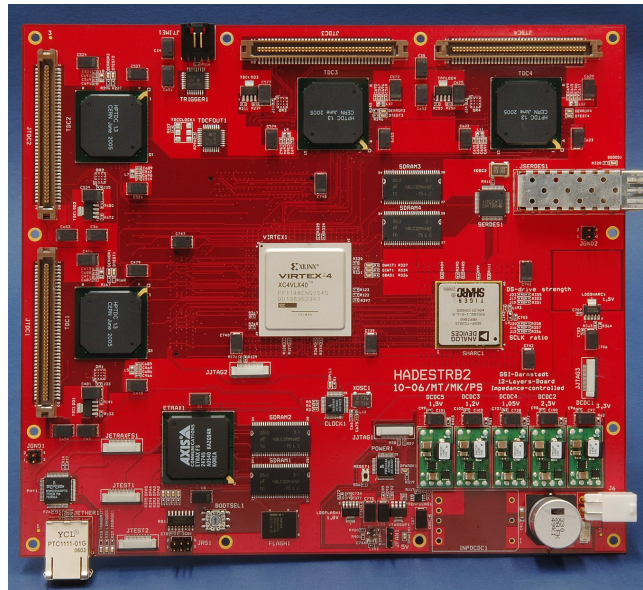


Figure 2.8.: The Trigger and Read-out Board (TRBv2) is used to read-out the TOF, RPC and forward wall detectors. It features a Xilinx Virtex4-FX40 FPGA, an optical link controlled by a TLK-2501 chip, 128 TDC channels with 100 ps binning provided by 4 HPTDC chips, a TigerSharc DSP and an Etrax FS embedded Linux processor.

connector for power supply and threshold settings. Other boards like the Hub or Shower-AddOn are also able to perform high-speed communication between the FPGAs on both boards.

RICH

The RICH read-out system consists of the front-end electronics equipped with pre-amplifier circuits. These front-ends are connected to the ADC module (ADCM) that features a Lattice ECP2M-100 FPGA [49] and a SFP transceiver module. A 16 channel, 12 Bit, 40 MSPS ADC [50] digitizes the data from a total of 1024 front-end channels. A microcontroller supervises the power supply and controls the start-up procedure. In total, 450 front-end cards and 30 ADCM are built into the RICH read-out system.

MDC

The front-end of the MDC system is composed of the Optical Endpoint (OEP) which is connected to a motherboard (MBO) equipped with TDC chips [18] and daughterboards (DBO) housing a pre-amplifier, shaper and discriminator circuit [21]. The OEP is mounted on top of the MBO on the edge of the detector chambers. This gives very strict space constraints of 4 cm times 5 cm for each OEP.

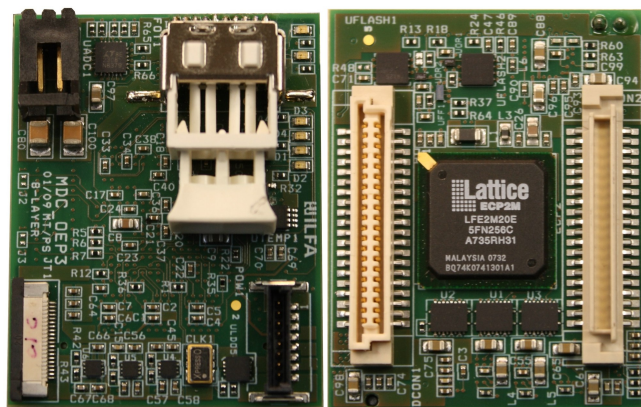


Figure 2.9.: The Optical Endpoint Board (OEP, left: top view, right: bottom view) is used to control the read-out of the MDC system. These boards are connected to the front-end electronics equipped with TDC chips. The OEP itself contains a Lattice ECP2/M-20 FPGA, an 250 MBit/s FOT transceiver, voltage regulators, level converters, two Flash ROMs and an ADC. The size of the board is 4 cm times 5 cm.

It contains a Lattice ECP2/M-20 FPGA and a fiber optical transceiver (FOT, 250 MBit/s). Voltage regulators generate and stabilize four of the six voltages needed for the front-end electronics. All voltages can be monitored by an on-board ADC. Two Flash ROMs provide two different configuration of the FPGA. One acts as a fall-back solution in case of a failure of the second design which can be altered remotely. The interconnectivity to the MBO is given by level converters that adapt the FPGA outputs to the 5 V signal levels used on the MBO.

The full MDC system contains 372 OEPs shown in figure 2.9, controlling 64 or 96 read-out channels each, depending on the type of motherboard. The complete front-end read-out logic is described in chapter 6.2.

pre-Shower

The pre-Shower detector is read-out by the Shower-AddOn board. It is designed as an AddOn to the TRBv2 but is used in a stand-alone mode in the HADES setup. Only the power supply is shared with the TRB.

The board is equipped with three Lattice ECP2/M-50 FPGA. Two of these FPGA are connected to a total of 12 8-channel, 60 MSPS, 10 bit ADCs. The inputs are connected to the front-end electronics that contains signal amplifiers and a multiplexer. In total, each sector has 2826 channels organized in a 32 x 32 matrix read out by one board.

The third FPGA on the board merges the data from both read-out FPGAs and connects to two optical links. One provides the usual 2 GBit/s link of the DAQ network, the other is used

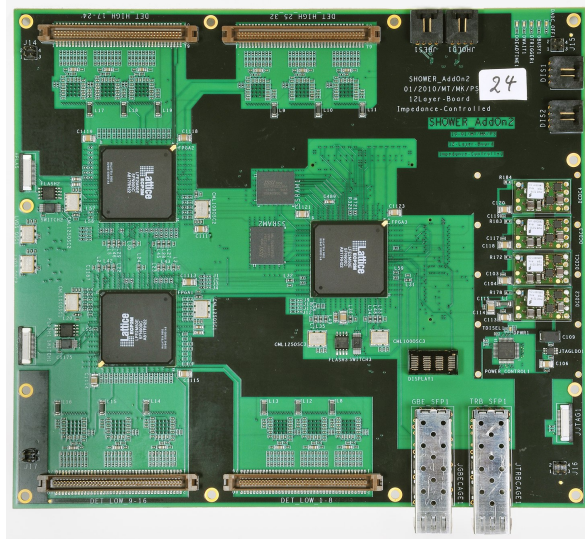


Figure 2.10.: The Shower-AddOn contains three Lattice ECP2/M-50 FPGA. Two read-out a total of 96 10-Bit ADC channels, the third act as a data merger. Two optical links provide connections with both the DAQ network and Gigabit Ethernet. The board is designed as AddOn to the TRB but can also be operated stand-alone.

to transport data to the server farm using a Gigabit Ethernet link.

CTS

The Central Trigger System is implemented on a dedicated board. A fast LatticeSCM-40 FPGA [51] generates the trigger signals. The connectivity for signals generated by the detector front-ends is given by 84 general purpose LVDS I/O and 30 PECL outputs used to distribute the reference time signal. Many additional monitoring features for the inputs are implemented in this FPGA, the full functionality is described in section 4.3.

A second FPGA, Lattice ECP2/M-100 controls the LVL1 trigger and read-out sequence over the optical network. In total four SFP connectors are available on the board. The CTS board is an AddOn to the TRB but can also be operated in a stand-alone mode.

Hub Boards

The Hub board (figure 2.12) has been developed as interconnection between all front-end boards. It is equipped with two Lattice ECP2M-100 FPGA and 20 SFP links. The first FPGA is connected to 16 SFP and serves as the main network hub. A parallel bus is used to connect it to the second FPGA where special features are included. It provides the uplink to the central trigger system and slow-control devices, as well as a Gigabit Ethernet connection to send data.

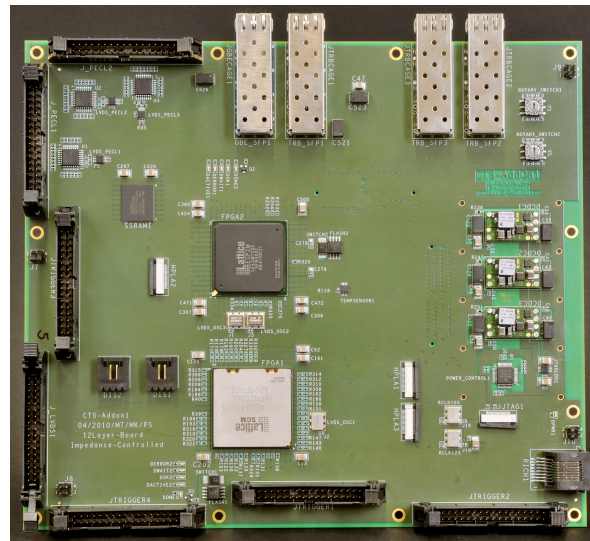


Figure 2.11.: The Central Trigger System (CTS) is based on a two-FPGA board. A fast LatticeSC/M-40 FPGA generates trigger signals based on various inputs from the detectors. A LatticeECP2/M-100 FPGA controls the LVL1 trigger and read-out process. Four optical links provide the interconnectivity with the DAQ network.

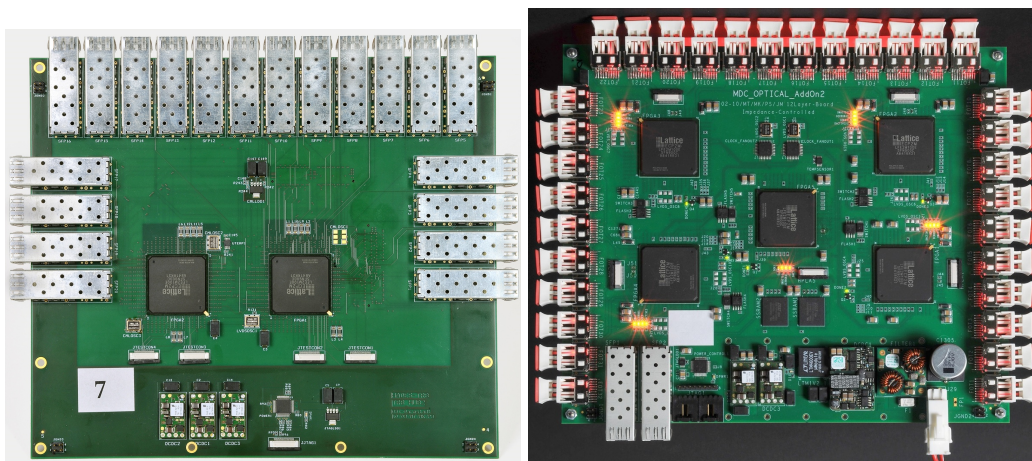


Figure 2.12.: The Network Hubs are used as interconnection between all other electronics. The Hubv2 (left picture) features 20 optical links with 2 GBit/s each while the MDC-Hubv2 (right picture) is equipped with 32 FOT-transceivers and 2 2 GBit/s optical links. On both boards, Lattice ECP2M-100 FPGAs are used; 2 on the Hub, 5 on the MDC-Hub. Both boards are capable of transporting data via Gigabit Ethernet to the server farm.

2.2. The New Data Acquisition System

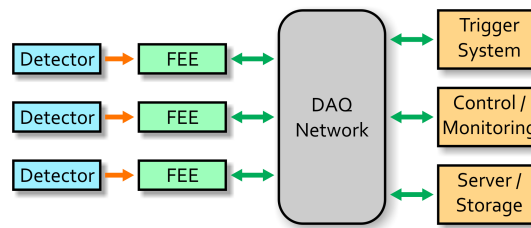


Figure 2.13.: A simplified view of the full data acquisition system. Detector signals are digitized in front-end modules. A DAQ network connects all front-ends to data storage, monitoring and the central trigger system.

The MDC-OEP is equipped with a different type of transceiver to fit the MDC OEP board. Hence, a special hub board is needed. Four Lattice ECP2/M-100 FPGA are serving 32 250 MBit/s FOT transceivers. Between those a fifth FPGA provides the connection to the 2 Gbit/s DAQ network and Gigabit Ethernet. The on-board interconnect is built by a mesh of serial data links. One of these hubs serves two MDC chambers as read-out interface.

2.2.3. The Complete DAQ System

The data acquisition system consists of several individual parts: The detectors are equipped with front-end modules that digitize the signals produced in the detectors. The second part of the system consists of the data storage servers and all control and monitoring systems. All sub-systems are connected using a dedicated data acquisition network. The trigger and read-out process for the full system is controlled by the central trigger system.

The complete DAQ network setup consisting of all boards shown in section 2.2.2 is shown in figure 2.14. The two types of network hubs provide the interconnect between all subsystems. The network is organized in a tree-like structure, with one central network hub that connects the Central Trigger System and the slow control interface to other systems.

From the central hub, the network is separated into its subsystems which are again connected using additional hubs. This separation gives the possibility to run each subsystem individually during test periods. Additionally, data from all subsystems is not merged and can easily be disentangled during data analysis. All front-end boards are connected to the CTS using three intermediate Hub boards at most to keep the latency for data transmission as short as possible.

The backbone of the DAQ system is formed by a set of servers (“Event Builders”) which collect data from all subsystems, combine it into one data block and write it to the data storage. First, data is stored on local hard drives and subsequently transported to tape or disk storage in the GSI computing center.

The data transport to the server farm and slow control from the control room is handled via Gigabit Ethernet links and a standard infrastructure (see figure 2.15). The DAQ network

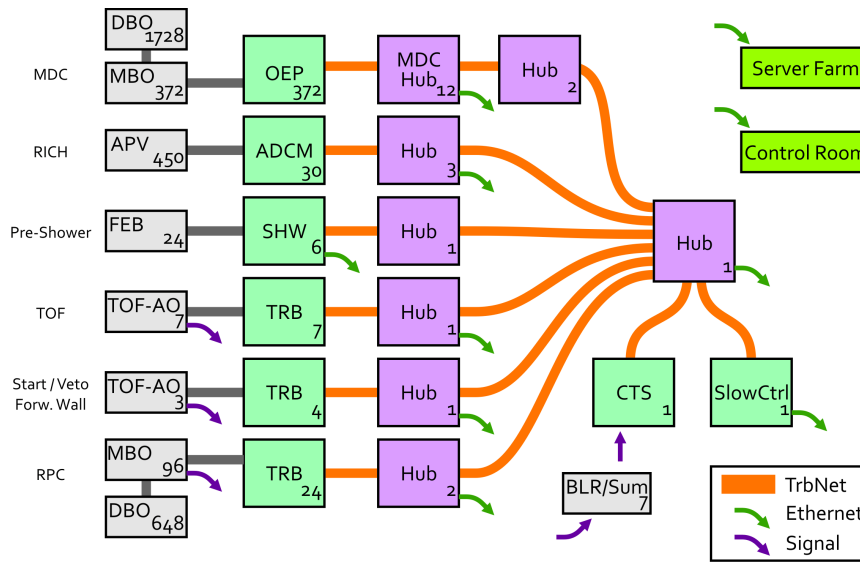


Figure 2.14.: A schematic view of the HADES DAQ network setup. Network hubs are shown in purple, all read-out boards in green. Additional front-end electronics are shown in gray. The small numbers indicate the amount of boards of each type in the DAQ system.

All boards use a common network protocol for data transport within the DAQ system. The interconnect to the server farm and control room is built by standard Gigabit Ethernet links. Parallel to the data network, differential signals are used to generate (dark purple arrows) and distribute trigger signals (not shown).

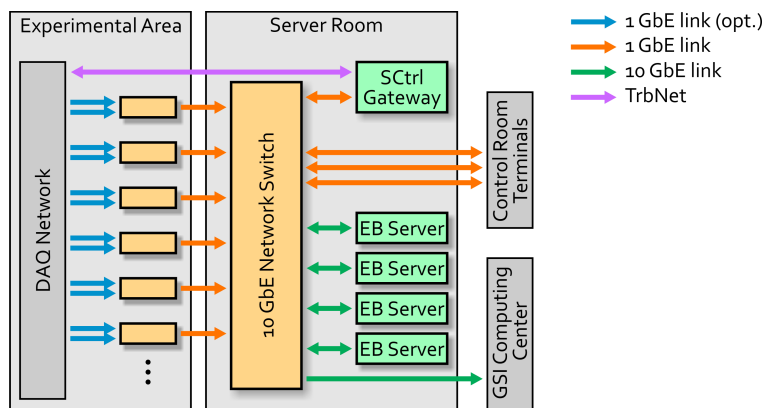


Figure 2.15.: The second part of the data acquisition system is formed by a Gigabit Ethernet Network. Data is sent by the DAQ electronics via 25 Gigabit Ethernet links to Ethernet switches which forward the data to the server farm and to permanent storage. All slow-control and monitoring facilities are connected to this network as well.

2.2. The New Data Acquisition System

sends data using 25 optical GbE links to network switches that convert the data to copper-based media. A central 10-GbE network switch connects all systems to the server farm and the computing center.

In parallel to the data network, a set of signals retrieved from the front-end modules of the timing relevant detectors are being transported to the CTS and are used to generate the trigger signal. Here, the reference time signal is generated and transported to all subsystems using a dedicated network of differential signals.

3. DAQ Network Structure

This section describes the logical structure of the TrbNet protocol. In the first section, a general overview of the network features is given and the data format used to transport data is shown.

The logical blocks implementing the network features are described in further detail in section 3.2 followed by a description of some special features, which are implemented within the protocol stack, in section 3.3. In the last part of this chapter, the operation of the two types of network nodes, namely endpoints and hubs, are explained.

3.1. Data Format and Network Channels

All data transported over the HADES DAQ network is organized in so-called *packets*. These packets are the smallest units of any data transfer. Each packet has a fixed size of 80 Bit including a 16 Bit header that defines the content and context of the packet and 64 Bit of payload. Internally, all data are handled in 16 Bit wide *words*¹⁶, allowing for a simple adaption to any word size of higher level data like 32 or 64 Bit. Hence, a packet consists of one header word (H_0) and four data words ($F_0 - F_3$).

The data format foresees five different packet types which are identified by the packet header H_0 . The contents of the related data words are shown in table 3.1.

DAT This packet type is used to transport any kind of data within its 64 Bit payload.

HDR The header packet initiates a transfer and contains the network addresses of both source and receiver of the transfer. Besides, it contains a sequence number that can be used to keep track of all transfers being made. A 4-Bit data type field is used to specify the contents of the transfer. As an optional feature, the length of the transfer can be included.

TRM The termination packet marks the end of a transfer on the network and contains basic status and error information as well as a checksum. The packet also contains the sequence number and data type which are sent within the header packet.

EOB The end-of-block packet is used to separate transfers into smaller *blocks* of data, typically 1 kbyte, which have to be acknowledged by the receiver.

¹⁶At 16 Bit word width, a clock speed of 100 MHz is sufficient to match the speed of the fastest optical links used.

Name	ID	F0	F1	F2	F3
DAT	0x0	64 Bit data payload			
HDR	0x1	source	target	length	seq.no. & type
EOB	0x2	CRC ¹	reserved	word count	buffer number
TRM (request)	0x3	CRC ¹	payload		seq.no. & type
TRM (reply)	0x3	CRC ¹	error & status information		seq.no. & type
ACK	0x5	resv.	buffer size	reserved	buffer number

Table 3.1.: Network Packet Types. The ID is encoded in the H_0 header word. The TrbNet protocol defines a set of 5 different types of network packets. For each, the corresponding payload is given in columns F_0 to F_3 .

⁽¹⁾ in case the CRC is not used, this word is available for additional payload / status information

ACK The acknowledge packet is sent by the receiver of a data stream to inform the sender that data has been received correctly.

According to the different tasks the network has to fulfill, it is divided into 3 virtual network channels which carry the LVL1 triggers, read-out and slow-control data. The transport of LVL1 triggers has to be guaranteed to be carried with the shortest possible latency not influenced by data transfer on other channels. The next-highest priority is assigned to the read-out channel while the slow-control has the lowest priority. Unlike in other network protocols, the virtual channels are separated inside each network node and do not share buffers (compare figure 3.1).

A switch between network channels is possible after each full packet has been transmitted. Due to the very low granularity the delay introduced in the transport of high-priority packets is always less than 50 ns on fast optical links.

3.2. Network Layers

Typically, the structure of a data transport protocol is described in a model of different layers [52], each providing specific features and coping with a specific part of data transmission.

The TrbNet protocol stack consists of six layers as shown in figure 3.1. Each layer adds another feature or information to the data stream as shown in figure 3.2. The lowest level is formed by the media connecting two network nodes, the physical layer. Data is separated by the multiplexer which is followed by the transport layer securing data transfer and checking data for their integrity. Network addresses and message filtering is implemented in the transaction layer while the connectivity to the users logic is provided by the application interfaces. These features are described in detail in the following sections.

Chapter 3. DAQ Network Structure

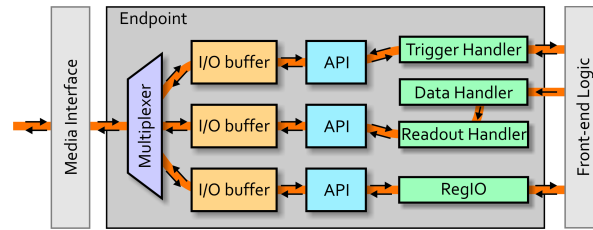


Figure 3.1.: The TrbNet protocol stack consists of six layers, the lowermost is composed of the physical layer (media interface), followed by the I/O buffers (transport layer), the API (transaction layer) and the application layer. The three logical network channels are handled separately inside each network node.

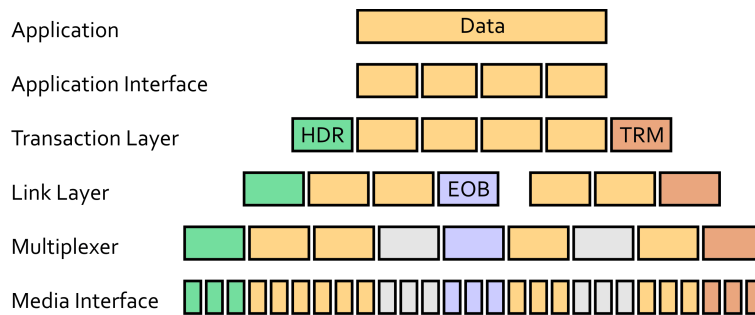


Figure 3.2.: Each network layer adds an additional feature to the data transported. The application interface prepares the data offered by the “User”, the transaction layer adds a header and termination word. The link layer organizes data in blocks with checksums and handshake signals (purple). The multiplexer merges data from different network channels (gray) and the media interface converts data into a format the physical link can handle.

One remarkable feature of the TrbNet protocol stack is that it separates all data into three logical channels directly above the physical layer. Each channel serves one of the different functions joined in the network protocol. Nonetheless, all channels incorporate the same logical blocks and differ in their respective application interfaces only. The different channels are further described in section 3.1.

The whole protocol stack is available as one building block and referred to as the endpoint. The physical layer is not included within this block since it has to be chosen depending on the exact hardware configuration of each module and can not be totally unified throughout the system.

3.2.1. Physical Layer: Media Interfaces

The media interfaces provide the low-level connection between different network nodes. In the Hades data acquisition system, most links are based on serial data transmission. On-board connections make use of electrical signals while all off-board links use optical data transmission. Due to different FPGA types (Lattice SC/M, Lattice ECP2/M, Xilinx Virtex-4) various interfaces are available. Additionally, the inter-FPGA connection on the Hub2 board is based on a parallel 20 bit bus running at 100 MHz.

Independent from the hardware, all media interfaces share a common set of features. First, appropriate hardware blocks need to be implemented to transport data into and out of the FPGA. For serial links, the Serdes hard-cores of the Lattice FPGAs are employed. On Virtex-based boards, an external chip generates the serial data stream and is interfaced to the FPGA using a parallel bus. Here, basic I/O-Flipflops are sufficient to transport data.

All links transport data combined with a clock signal (synchronous transport) so that the receiver has to convert the incoming data stream from the external clock signal to the internal system clock of the FPGA before it can be handled further. In some cases, such a transfer is also necessary on the transmitter side.

Each interface features a set of control words or signals that facilitate link operation. On serial links the 8b/10b encoding provides a set of ten control characters that can be used. On the parallel interface an additional signal is used to distinguish between data and control words.

The start-up of a link is controlled by a fixed sequence to ensure a synchronous operation of the transceivers on both ends. Depending on the link type, a special control word is used to signal the receiver the availability to transmit data. Now, a initialization sequence is started with a fixed timing. After a given time, receiving of data is enabled; after another delay also the transmitter is fully activated. If the link fails or invalid words are received, the sequence is restarted. Figure 3.3 shows the start-up sequence used on all serial links; the parallel data interface uses a similar but simpler scheme since no clock recovery and synchronization has to be performed.

The TrbNet data format relies on 80 bit packets, divided into 5×16 bit words. The internal logic of the endpoint requires the 16 bit words to be tagged according to their position in the packet. This information is not directly transported between the nodes but has to be regenerated inside the media interface, e.g. by counting the number of received words or by insertion of special code words to mark the beginning of each packet.

All links are capable of transporting a global reset signal that forces all connected boards into a defined state. On some optical interfaces an error detection and correction feature is implemented. This function is further explained in section 3.3.4.

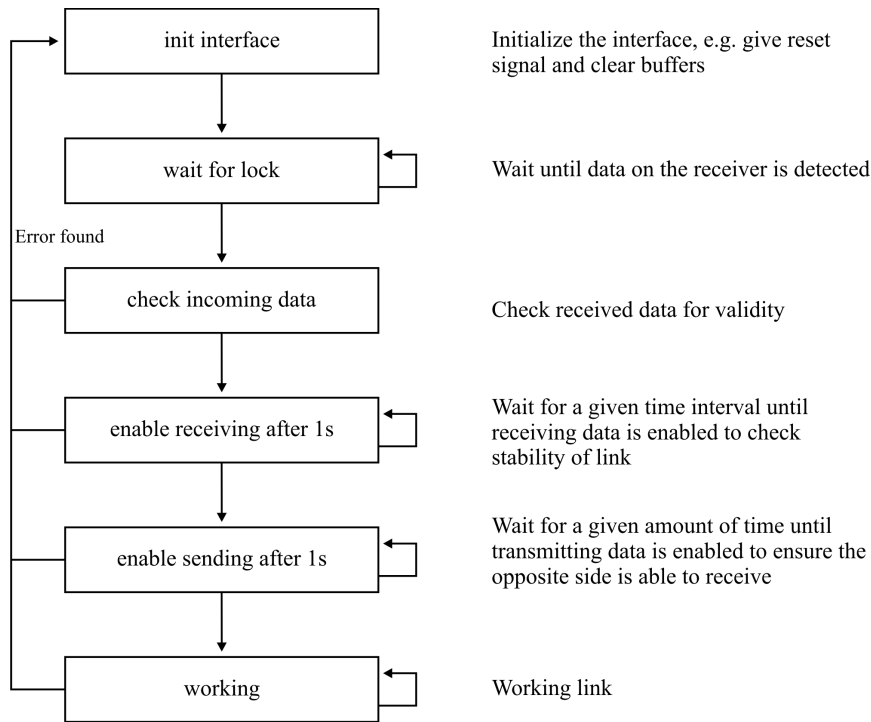


Figure 3.3.: The activation of each network link is controlled by a link state machine that starts a defined sequence of steps as soon as a valid signal is received. After a lock to the data stream is achieved, and a certain time has passed, the receiver is enabled; after another delay also the transmitter is activated. This state machine is used on all serial links, the parallel interface uses a similar scheme.

3.2.2. Multiplexer

The low latency and non-blocking behavior between different channels can only be guaranteed if data from all three channels are handled separately inside the network nodes. On the receive path, a de-multiplexer forwards the incoming data to the different IO-Buffers, depending on the channel number provided in the first word (H_0) of each network packet.

In the transmit path a priority arbiter selects which channel is allowed to send data based on the availability of data and its priority. New decisions are taken after each network packet so that the worst case latency for a high priority packet is five data words. Depending on the endpoint logic, there might be additional clock cycles delay in case a packet is not sent in one block but has intermediate pauses caused by higher level logic.

The transmit path can be configured with additional buffers to overcome this problem by first collecting the full network packet and forwarding it to the multiplexer only when it is complete. This introduces a fixed latency to the data transmission but is necessary for correct operation of network hubs. Here, a special sequence of input packets could produce a dead-

lock between two channels if packets were forwarded without buffering.

3.2.3. Link Layer: IO-Buffer

The IO-Buffer controls the data transfer between two adjacent network nodes. It consists of a local output buffer connected to a remote input buffer and vice versa. The two duties of the link layer are the prevention of data losses due to buffer over-flows and to provide a check for data integrity.

Whenever data is sent over the network, it has to be assured that the receiver is able to handle the data. It might always happen that data can not be forwarded due to various reasons. For example, the interfaces to Gigabit Ethernet or to a PC on the slow-control channel have a smaller bandwidth than most parts of TrbNet. This can lead to delays in data transport. Another major cause is the data merging done inside network hubs. Only data from one front-end link can be transported at a given point in time while all others have to be stalled.

Hence, all receivers contain a data buffer of defined size (usually two times 102 network packets, i.e. the buffer size is 2×102 packets which is 1020 words). All data transferred is divided into blocks, each not bigger than the buffer in the receiver. The receiver has to acknowledge the receipt of each block of data as soon as the input buffer is cleared and ready to accept the next block of data.

This handshake is implemented in the link layer as shown in figure 3.4. The output buffer adds a `EOB` packet to the end of each block of data. The `EOB` including a running number and the number of words of the current block. As soon as this word is read from the buffer in the receiver, a `ACK` packet is sent back that allows the transmitter to send the next block of data. The receiver is able to store two full blocks of data to eliminate the waiting time between sending a block of data and receiving the acknowledge.

Additionally, each `EOB` and `TRM` contains a 16 Bit CRC check-sum calculated from the full payload of the data block. This check-sum is validated by the receiver and, if a mismatch is found, a error bit in the termination packet ending the transfer is set. A retransmission of data is not foreseen on this level due to complexity reasons.

3.2.4. Transaction Layer: API

The transaction layer provides the application interface (API) to the network. In particular, it accepts raw data from the application side and also strips received data from all overhead added for the network protocol. As a consequence, the application is provided with the words `F0` through `F3` from each packet only.

The header packet is checked if the transfer is addressed to this particular end-point. If not, it is automatically answered with a short transfer (see section 3.3.2). If data has to be processed by the application, all data payload is forwarded. Afterward, the API waits for reply data that is sent back to the requester. These data is automatically equipped with a

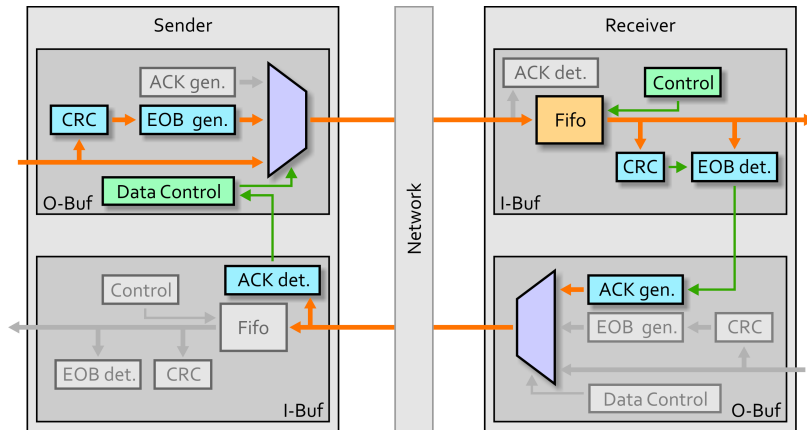


Figure 3.4.: The IO-Buffer perform a handshake to guarantee that the receiver is able to accept data. From top left, clock-wise: The output buffer of the sender divides data into blocks and at a EOB packet with a checksum. The receiver input buffer temporarily stores data in a buffer and detects the EOB. A ACK packet is sent back to the sender which may send the next block of data.

header and termination packet and all data is packed into DAT network packets.

3.2.5. User Interface: Trigger, Data and Slow-Control

The uppermost layer of the TrbNet structure is formed by the user interface. This interface provides all connections needed for the front-end logic to communicate through the data acquisition network with the highest abstraction possible. That means, the interfaces are kept as simple as possible to ease connecting any front-end logic to the network endpoint.

Due to the different purposes on each of the network channels, there are three independent blocks for each of the channels. First, the LVL1 Trigger Handler provides the interface on which trigger signals are received, checked and provided to the front-end logic. The front-end can give back a busy release signal along with basic status information. This block is described in detail in section 4.4.

The read-out is fully controlled by the Data and Read-out Handlers. The front-end fills its data into a configurable buffer and does not have to implement any read-out control features. The data interface and the read-out process are fully described in sections 5.2 and 5.4, respectively.

All slow-control, configuration and monitoring features are handled by the RegIO component. It provides a set of standardized features such as network address handling and a common set of registers. All front-end dependent functionality can be added to the general purpose data bus provided by the entity. These features are described in chapter 7.

3.3. Special Features and Definitions

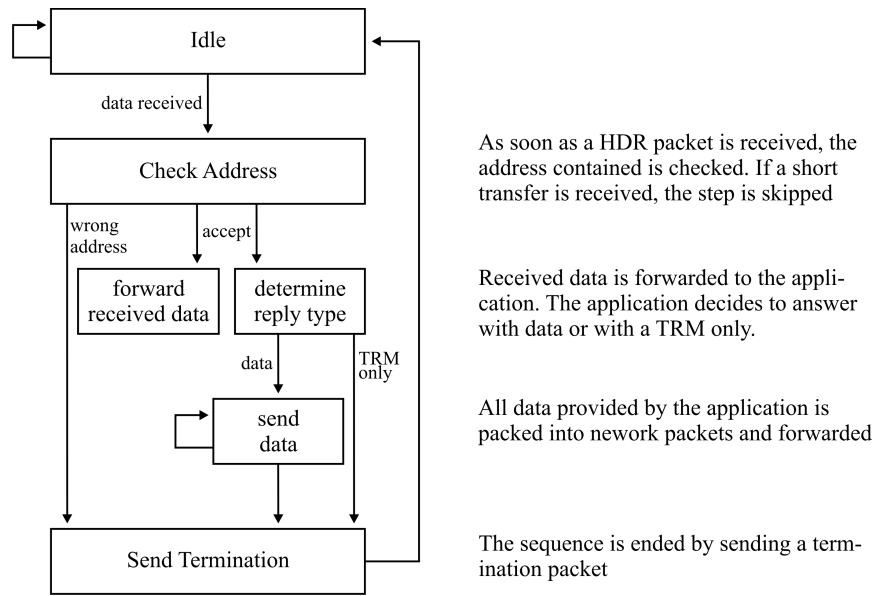


Figure 3.5.: The application interface (API) receives data from the network, checks the network address and forwards all received data to the user interface. The user interface provides the data to be sent back to the network. After data transfer in both directions is finished, a termination packet is added and the cycle is finished.

3.3. Special Features and Definitions

The TrbNet includes a set of special features which are dedicated for the use in a data acquisition network which are described in this section. Network addresses are assigned based on unique board identifiers as described in section 3.3.1. All data transfer on the network is organized in transactions, consisting of a request and a reply all boards are forced to sent (see section 3.3.2). On the low-level media interface, an error detection and configuration mechanism is implemented (section 3.3.4).

3.3.1. Network Addresses

Within the network, all network nodes are individually addressable by a 16 bit address. These addresses are used to send control messages and requests to single boards and to mark the source of status information. They are also used to identify the data source during read-out in form of sub-event and sub-sub-event IDs (see section 5.1).

During the start-up process of the network, these addresses are assigned to all boards based on unique hardware IDs. For this reason, every network node is equipped with a DS1820 1-wire temperature sensor [43]. These devices contain a 64 Bit unique identifier that is used to identify individual hardware components in the network. This number is automatically read-out during power-up and made available to the slow-control system. The start-up software

contains a table with all hardware ids and assigns the corresponding network addresses by sending the id/address combinations as a broadcast to all boards. After this procedure, all network nodes are individually accessible for any kind of transfers.

The actual addresses are chosen to represent both the physical structure of the detector and the different sub-systems. Hence, all addresses can easily be decoded without requiring excessive look-up tables. All addresses are listed in table A.2.

Besides the individual addresses, the network protocol also features a set of broadcast addresses. Addresses in the range 0xFF00 – 0xFFFF are broadcast addresses for all boards or for boards belonging to a specific sub-system. In this block, nine different broadcast addresses are available by applying a hard-coded bit-mask to the lower eight bits of the network address. 256 further broadcast addresses for specific FPGA types are available within the 0xFE00 – 0xFEFF range. The full list of assigned network broadcast addresses is given in table A.1.

3.3.2. Transfer Types and Network Transactions

A full network transaction consists of two transfers: First, a request is sent to the network. The hubs distribute this request to all network nodes since no routing is being done. Second, all nodes receiving a request have to send a reply. This reply is routed toward the node that issued the request.

In either case, only if a node is addressed by the request the user logic is allowed to send back a reply of either transfer type. If a node is not addressed, the API automatically sends back a short transfer to acknowledge the receipt of the request. Before all nodes have sent their termination packet, no other request may be done.

The fixed two-step process for each transaction gives several advantages: First, it directly resembles a busy logic for trigger distribution if the front-ends delay their answer until they are finished processing a trigger. The requirement that all nodes must answer also gives the possibility to identify if all boards are working properly. On the slow-control channel where the number of replies varies depending on the address it gives the hub a way to determine how long it has to wait for additional data.

If a erroneous node is connected to the network, it may fail to send a reply. As a result, the transaction could not be completed and the network would be blocked. To prevent the network from being locked, the network hubs contain a time-out limit. If the reply has not been received within this time limit, the network link is deemed to be broken and the transaction is forcefully finished.

The network protocol foresees two types of network transfers: full and short transfers. A full transfer consists of a header and termination packet and an arbitrary number of data packets in between.

In case only a small amount of data has to be sent as a request, a short transfer can be issued. Here, only a termination packet is sent. Due to the lack of a network address within

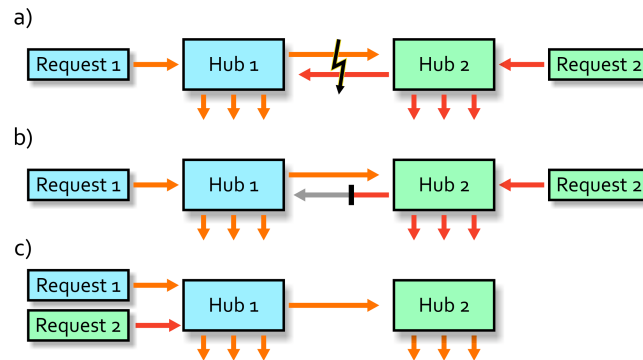


Figure 3.6.: (a) A dead-lock might happen if two requests are sent at the same time by different nodes. (b) If one direction of request sending between hubs is blocked, no collision can happen. (c) The condition can not happen if both requesters are connected to the same hub.

this packet, a short transfer is always a broadcast to all nodes. This transfer type is sent by the CTS as LVL1 trigger and read-out request.

While a long transfer has no inherent limit on the amount of data transported, a short transfer is limited to the payload available within a termination packet. Here, 36 Bit of payload can be included; on network channels with a disabled check-sum feature, the allocated space can also be filled with information so that a total of 52 Bit of payload can be transported¹⁷. The short-transfer can also be send as a reply. In that case, no data but only error information can be transported.

Transaction Collisions

On the other hand, the fixed two-step transaction process also has one drawback: It is not possible to have an arbitrary number of request-senders. If two requests are issued by two different nodes connected to different network hubs at the same time, there will be a dead-lock as depicted in figure 3.6: The first hub each request is sent to, accepts the transmission and transmits it to the other hub in the depicted system. Since this hub is already blocked by the second request, it has to stall the incoming request. Hence, both transfers block each other and can not be completed.

There are two possible ways to circumvent this specific problem¹⁸. Both base on the fact that this kind of dead-lock can only happen if one network link is used by both requests in opposite directions. If we assume that a node that sends requests is not accessible by requests from a remote node on the same channel¹⁹, there are two possible solutions.

¹⁷This is the case on the LVL1 trigger channel only. All other channels use check-sums.

¹⁸The HADES DAQ setup uses only one request sender per network channel so that this problem does not exist.

¹⁹This is true for almost all system setups. Either a node is a addressable front-end or functions as a request

First, all request senders have to be connected to the same logical network hub. Here, both senders are connected to the hub using dedicated links and all further transmission paths are shared between both requests. The hub stalls one request until the other is completed without errors.

The second solution is that one network link is configured as a one-way connection on this channel. That means that only one request can reach the complete network while the other is kept within a smaller sub-network. This configuration matches nicely with typical data acquisition setups in which a full system is made up by several sub-systems. If a request is addressed to one sub-system only, it can be issued by the restricted node, or it is issued by the global node if it is sent to all sub-systems.

Termination Packet

The termination packet is handled differently on the request and reply paths: On the request path, the packet is transported unaltered by all intermediate network nodes (Network Hubs, see section 3.4.2). In the reply path all network nodes are required to send at least a termination packet back. Since the full DAQ system consists of about 550 individual network nodes, this results in a substantial amount of data. On the other hand, all termination packets sent by not addressed nodes do not contain viable information besides the acknowledge that the request has been received. Hence, all received termination packets are merged in the network hubs to one common packet. A common-or logic is applied to the 36 Bit payload of all packets to obtain the final termination packet forwarded to the requester.

One important implication is that the termination word can not be used to transport arbitrary data in the reply path. Moreover, if a single node sets a given bit in the termination packet, it will remain set in the final packet. This results in the requirement that these bits are used to transport warning messages and error information only and should remain unset in most cases. A further explanation of these messages is given in section 3.3.3.

Time-outs

During network operation, it might always happen that a single network node fails and does not send a reply. Since the hub is waiting for replies from all connected front-ends before finishing a transaction, this will lead to a blocked network were no further accesses are possible. Hence, the hubs contain an intrinsic time-out will waiting for a reply. If no data are received within a given time, the transaction is automatically ended by sending the termination packet. The time-out delay can be adjusted for each channel and each hub separately via slow-control. Additionally, the time-out can be fine-adjusted in steps of milliseconds to compensate for latency introduced by network topology.

sender, e.g. as CTS or as slow-control access point.

3.3.3. Error- and Status Bitpattern

The termination packet on the reply channel contains a 32 Bit word with status and error information. This word is generated by all front-ends but merged into a single word by the network hubs (see section 3.3.2). Hence, this data word can not be used to transport usual data but each bit within the word has a specific meaning if it is set. Moreover, the default value of a bit has to be cleared since if one bit is set by one front-end, it will also be set in the final word.

The word is divided into two parts. The lower half contains error information generated by the network that occur within the end-points or during data transport. The upper half is reserved for channel specific error information and described in the appropriate chapters (sections 4.6, 5.8 and 7.1).

The least significant bit plays a special role. For a given network request, all boards are allowed to answer the request by sending a short-transfer. This reply can by itself not be distinguished from a situation where the addressed board did not exist in the network and no node accepted the request. Therefore, the bit is assigned by all network nodes that were addressed by the request. If a reply returns with this bit unset, it is evident that the address given does not exist.

The further bits are assigned according to the following list. The bits not listed are reserved and not defined in the current implementation.

Bit 2: Word Missing The `EOB` packets contain a count of packets that have been sent in advance. If this number of packets does not match the received data, the bit is set by the input buffers.

Bit 3: Check-sum Mismatch On each point-to-point link, a check-sum is generated by the output buffer and validated by the input buffers. If a mismatch is found, it is reported in the error information.

Bit 4: Don't Understand Not all nodes support all kinds of requests, e.g. not all possible data types encoded in the header are implemented. In this case, the application can answer a request with a short transfer including this error bit.

Bit 5: Buffer Mismatch The handshake between IO-Buffers contains a running number for both `EOB` and `ACK` packets. If these numbers do not match, some packet must have been lost.

Bit 6: Answer Missing The network hubs contain a time-out while waiting for a reply. If the reply is not received within the time limit, the transaction is finished and this error bit is set.

3.3.4. FOT Link Error Correction

The HADES DAQ system consists of about 500 bi-directional optical links, 372 belong to the MDC read-out system and are formed by plastic optical fibers. During tests of the whole DAQ system, these links were found to be operating stably without transmission errors. The total Bit error rate (BERR) of the system was found to be below 10^{-16} , corresponding to less than one transmission error per day.

The first commissioning run revealed that the high voltage systems of the detectors had a severe influence on the optical receivers in the MDC system. Here, the increased electromagnetic noise environment caused bit errors in the received data stream which caused the data acquisition to fail. Depending on the high-voltage settings, the error rate was up to 10^{-12} , i.e. one error every five seconds.

These errors were seen on the 250 MBit/s FOT transceivers but not on the 2 GBit/s SFP transceivers used in all systems apart MDC. Hence, it was decided to implement a error detection and correction protocol into the FOT optical link media interfaces. This protocol was not implemented for the 2 GBit/s links for two reasons: First, no transmission errors was detected throughout the full commissioning phase and second, the limited set of functions of the TLK2501 transceiver [46] used on TRB boards did not allow for an extensive low-level protocol based on control characters.

The full error handling sequence consists of error detection, retransmission request and retransmission. The error detection is based on three sources: First, the 8b10b encoding scheme all serial links use assigns only less than half of the available codes to valid characters so that single bit errors in the data stream are likely to change a valid character into an invalid one and can be detected. In other cases a single bit error changes a valid data word into another valid character. Hence, a checksum is added at the end of each TrbNet packet that is used to verify the validity of the packet. Lastly, all TrbNet data transfers are based on fixed sized packets. Thus, if data with an invalid number of words is received, this is likely to be caused by a transmission error. If one of these checks fails, the receiver is able to identify an error in the data stream.

As soon as the error is detected, all data starting with the erroneous packet are being discarded. The near transmitter sends a retransmission request containing the current buffer position. The far receiver gets the request and configures the transmitter to sent a retransmission. Here, the read position in the ring buffer is adjusted according to the request. A start character is inserted, repeating the buffer position followed by all data words that are available in the buffer. As soon as the receiver gets the start character, data checking and forwarding is re-enabled. If the start character is not received within a reasonable time, the retransmission request is deemed to be lost and repeated.

Name	Character	Value	Payload
Idle	K28.5	0xBC	0x50 or 0xC5 according to the GbE standard
Reset	K30.7	0xFE	None
CRC	K29.7	0xFD	CRC value
Start	K27.7	0xFB	Position in ring buffer where transmission starts
Error	K23.7	0xF7	Position of receive counter where corrupted data occurred.

Table 3.2.: The control character definitions used on optical links. Both the character name and the 8 Bit representation as well as a description of the 8 Bit payload following the character is given.

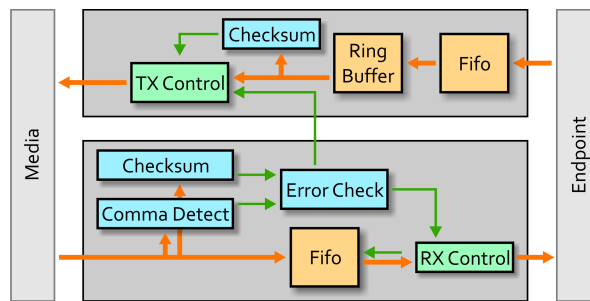


Figure 3.7.: The error detection and correction logic implemented in the FOT media interface

The Implementation

The 8b10b code foresees 10 special control characters that can be used to transport additional information besides normal data. From this set of characters, three have been chosen to transport all information to detect transmission errors and to request a retransmission of data if data was corrupted. These characters are described in table 3.2.

The complete error handling logic is divided into two parts: One is located in the transmitter, one is placed in the receiver. A schematic view showing the most relevant functional blocks is given in figure 3.7.

In the transmitter, data is first converted from the FPGA internal bus (16 Bit, 100 MHz) to the data rate used for the optical links (8 Bit, 25 MHz). This data is fed to a ring buffer with a depth of 256 words. Here, data is stored for few clock cycles until it can be read by the TX controller and be sent on the medium. Additionally, all data is available in the ring buffer until they are overwritten at least 256 clock cycles or 10.24 μ s later²⁰. If a retransmission request is received, the read position in the ring buffer can be adjusted according to the request and all data beyond this point is resent.

²⁰Precisely, data is stored until 256 additional data words have been sent. If the link is not entirely occupied by data, this time may be much longer.

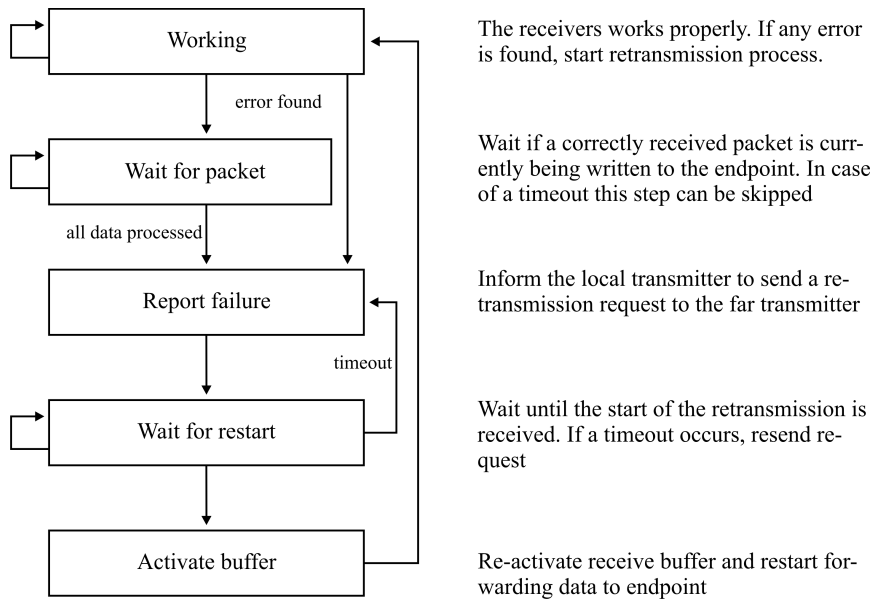


Figure 3.8.: The error detection and correction state machine is part of the receiver block.

The buffer time is also the maximum affordable latency of a valid retransmission request. The round-trip time for a packet is less than 1 μ s due to latencies in both transmitter and receiver blocks plus the transit time of the optical signal which is about 5 ns/m. Hence, in all setup scenarios the selected storage size is sufficient.

The transmission controller generates the data stream to be sent on the network. It reads data from the buffer and inserts the defined control characters to the data stream:

Idle K28.5 The idle character is added whenever no other kind of data or control character is waiting for transmission. The data payload consists of the 0x50 data character according to the Gigabit Ethernet standard²¹.

CRC K29.7 At the end of each network packet, a 8 bit checksum is added. It is marked by the CRC comma character and followed by the checksum value.

Reset K30.7 The reset character is sent to force a complete reset of the connected endpoint. It is always sent in a block of several dozen words, the reset takes place when the sequence finishes.

Start K27.7 The start character marks the beginning of transmission after a connection has been established or after a retransmission request has been received. The 8 bit payload contains the current position in the buffer for transmitting data.

²¹The GbE standard also requires the running disparity to be negative whenever an idle character is being sent. This is not implemented.

Error K23.7 The error character is used to inform the far transmitter about an error in the received data streams. The payload contains the position in the transmit buffer at which the retransmission has to be started, i.e. all data up to this point has been received correctly.

Data If no transport messages are to be sent and there is data available in the buffer, these words are read and sent.

On the receiver side, the incoming data is stored in a Fifo until the validity check gave a positive result. A comma detection logic extracts all comma characters from the data stream and gives the corresponding information to the control logic. In particular, the data is checked for code violations, undefined control characters and the correct position of checksums. In parallel, the checksum over all incoming data is calculated and compared to the one received with the data.

The control logic also monitors the contents of the Fifo if it contains incomplete packets which can be an indication for data loss. The reaction of the controller if any type of transmission error is detected is depicted in figure 3.8. First, all valid data remaining in the Fifo is being forwarded to the endpoint, then the buffer is disabled and the retransmission request is sent. After a Start control character has been received, the controller goes back to normal operation.

3.4. Network Nodes

All logical parts of the network, so-called network nodes, can be subdivided into two categories: Endpoints and Hubs.

A network endpoint is any component which is able to send and/or receive data on the network. It has a unique identifier and a network address that makes it accessible from other network nodes. In this sense, all front-end and control boards form network endpoints.

A network hub, on the other hand, is a device with several connections to the network which is used to route data streams between several other network nodes. A typical hub in the TrbNet network has 4 to 13 ports. Network hubs can also be supplied with additional components that implement conversion logic from one transport protocol to another. The most common implementation of such a network bridge is used to receive information on the TrbNet data channel and transport it to the Eventbuilder server farm using a Gigabit Ethernet protocol.

The two types of network nodes are not necessarily located on different physical devices. Some boards, e.g. the Shower AddOn, contain a hub in one FPGA and network endpoints in two additional FPGA. In some cases, both network node types are also located inside the same chip. In fact, each network hub also contains a network endpoint for control and monitoring purposes.

3.4.1. Endpoints

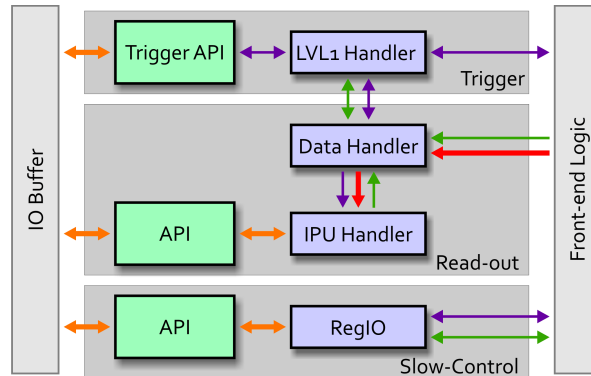


Figure 3.9.: A network endpoint with LVL1 trigger and data handlers as it is used in the front-ends. Note that only the upper network layers are shown. Compare figure 3.1.

The TrbNet endpoint for front-ends provides the interface to receive triggers and send event data over the network to the front-end logic. The interface to the slow control channel provides the internal data bus and preconfigured status and control registers.

The LVL1 Handler checks the inputs from the reference time signal and relates them to the LVL1 trigger packets received over TrbNet. The front-end logic is provided with all necessary information about the trigger. After finishing processing data, the busy release signal is given to the LVL1 handler.

The event data from the front-end is written to the Data Handler that contains the necessary buffers. The complete read-out process on the data channel is encapsulated in the IPU Handler and therefore independent from the front-end logic.

These data handling components, together with the RegIO component on the slow control channel, form the uppermost layer of TrbNet. According to the protocol stack, they connect to the transaction layer which is contained in the functional endpoint block as well. Likewise, the link layer is encapsulated.

The logic blocks for the upper protocol layers in the network endpoint are shown in figure 3.9.

There are several other endpoint configurations available, depending on the purpose in their implementation:

- A endpoint with three application interfaces only is used to implement the active TrbNet control endpoint that sends all requests within test setups.
- A front-end endpoint without Data and IPU Handler is used on the RICH front-ends. Here, the processing logic for raw data already contains the features so that an additional buffer stage is not necessary and data is fed to the application interface directly.

- The central trigger system uses a specialized version of the endpoint. The transaction layer on the LVL1 and slow control interface are equipped with logic for efficient generation of LVL1 triggers and read-out requests. At the same time, the slow control channel is implemented similar to front-ends and gives the same monitoring and control functionalities as in front-end modules.

3.4.2. Network Hubs

In Ethernet applications, a network hub is a device that sends all data it receives on any input to all outputs without changing them. A TrbNet hub fulfills additional tasks:

- Data from all network channels are handled separately. Like in any network endpoint, the incoming data stream is de-multiplexed. The data are passed on to three distinct hub logic blocks.
- Each port of the hub contains the point-to-point handshake logic contained in the IO-Buffers as described in section 3.2.3.
- The hub logic blocks process and route the data as described below. Responses from all connected ports are merged into a single data stream forwarded to the sender of the network request.
- On the IPU data channel headers from front-ends are checked for validity and additional headers are generated.
- Various statistics containing information about data rates, dead-times and error messages are generated. These monitoring features are described in section 3.4.2.
- The hub contains control logic to configure all ports individually, e.g. ports can be switched off on a per-channel basis to exclude parts of the system from data taking or slow control. These features are described in section 3.4.2.

A schematic view of the different parts of a network hub is given in figure 3.10. Here, the hub is shown with two ports only. A real network hub can be configured with a flexible number of ports. The current implementation allows up to 17 ports while the maximum used in current hardware are 13.

Hub Logic

The central part of the network hub is the hub logic. Here, data streams are routed and responses are merged before being forwarded. The logic can be divided into two parts which work in parallel. One part waits for network requests on either input channel and forwards it to all other connected ports. Once this data stream is started to be sent, no other input on the request channel is accepted until all responses have been received.

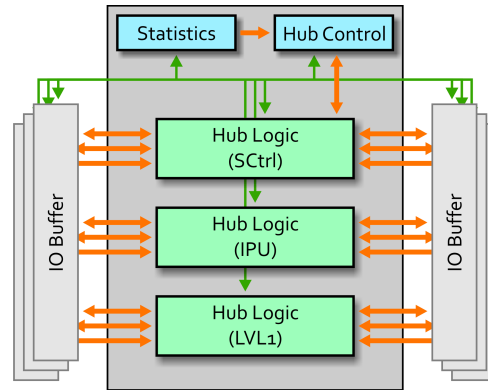


Figure 3.10.: The TrbNet Hub. Here, a hub with six ports is shown. The actual implementation of the network hub is freely configurable and usually has four to twelve ports. Data from different TrbNet channels is handled separately by the corresponding hub logic block. Additional components generate statistics and control the hub behavior.

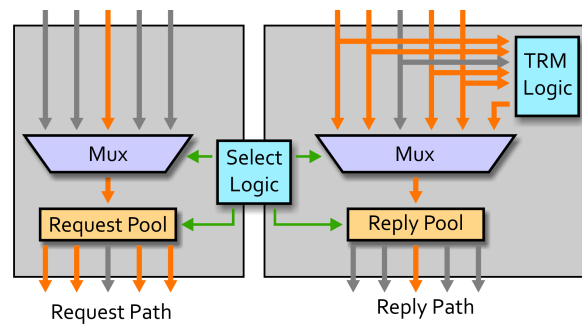


Figure 3.11.: The TrbNet Hub Logic. A hub with five ports is shown. On one a request is sent (upper left) and forwarded to the other four ports. On the reply path, data from these four links is received and forwarded to the one link that initiated the transfer.

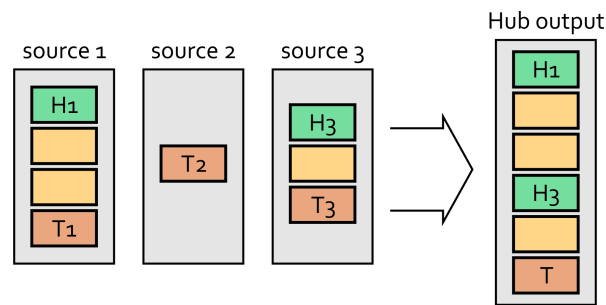


Figure 3.12.: The network hubs merge all incoming reply data into one data stream. All data from incoming streams is forwarded while the termination packets are merged into one packet using a common-or logic and added to the end of the stream.

The receiving and forwarding of responses is part of the second half of the hub logic. Data streams from all connected ports, despite the one on which the request has been received, are forwarded to the requester. The termination packets are not forwarded but automatically read by a special logic block. The status and error information words are merged and the resulting termination is sent to the requesting port. The resulting data stream is depicted in figure 3.12.

In a basic hub implementation, all input ports are allowed to be the source of a TrbNet request without distinctions. This gives the flexibility to connect an arbitrary number of active network nodes. Nevertheless, most experimental setups have an inherent structure which implies the existence of dedicated up-links. To save logic resources, the hub implementation can be configured to resemble this structure and accept requests only on a dedicated port or set of ports.

Hub Logic on IPU Channel

The IPU channel is operated by a special hub logic block that contains additional features to handle the event and data headers. Due to the format of the HADES SubEvent, information about the sender and length of a transfer have to be collected at the beginning of each data transfer. Thus, a calculation of total length and the generation of these special data header words has to be implemented. The full description of the logic is given in section 5.5.

Hub Control

Besides the options for hub configuration that are set during synthesis of the code, each hub contains a set of control registers that are used to configure the hub during run-time.

For example, each port of the hub can be switched off individually to exclude sub-systems from the network if necessary. Furthermore, the switching can be done separately on each of the three TrbNet channels. Thus, a sub-system might be excluded from triggering and data taking while still being accessible over the slow control channel. This feature can be

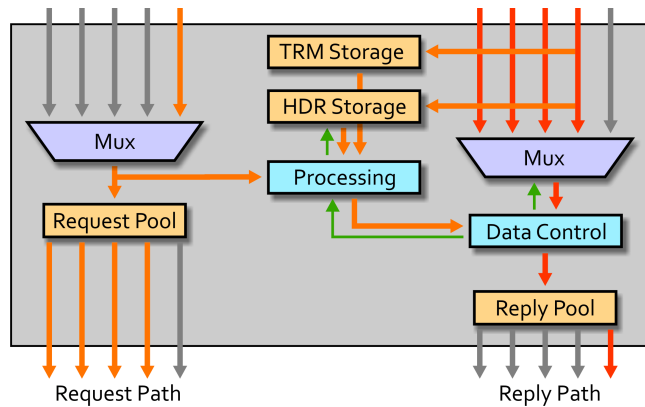


Figure 3.13.: The TrbNet IPU Hub Logic. A hub with five ports is shown. On one a request is sent (upper left) and forwarded to the other four ports. On the reply path, data from these four links is received and forwarded to the one link that initiated the transfer. Compared to the generic hub logic, data is also checked for validity and data headers are regenerated.

used to re-configure one board which experienced an error while the rest of the system keeps running.

In some cases it happens that one front-end board does not send an answer to a TrbNet request. Consequently, the hub it is connected to keeps waiting for an answer before it can finish the transfer and blocks all communication on the slow-control channel. To avoid this situation, the hubs contain a time-out logic. The maximal time the hub waits for an answer can be set by slow control. The typical response time until a reply arrives is less than 1 ms for all slow-control accesses but may be up to 10 ms on the LVL1 channel due to long lasting calibration measurements.

The time-out duration can be set in several steps in the range from 32 ms up to 8 s. This value can be set individually for each network channel and each network hub. In a full network setup with several layers of hubs have to be adapted to routing delays in the network: If a front-end board fails to answer, only the hub it is connected to is supposed to generate a time-out. The time-out duration has to be shorter on hubs connected to front-ends and longer on more central hubs. Consequently, the duration can be fine adjusted in steps of 2 ms.

The full list of configuration registers can be found in section C.2.

Hub Monitoring

The current operational status of the data acquisition can be summarized in three quantities: The availability of front-ends, the amount of data sent and the busy time. All these are monitored on-line in the network hubs. The information which hub ports have currently established connections and the amount of data received on the links is available via slow

control.

The hub keeps track of the busy time on the LVL1 trigger channel. This time is defined as the period between sending a trigger information packet towards the front-end and receiving back the busy release packet. Further information can be gained from comparing the total busy time and the exclusive busy time. That is the period during which only one of all connected ports is busy. Thus, it is a measure for if the observed busy time is a global effect from several sub-systems or if only one front-end causes a significant part of the total dead time. More status information is available to see if time-outs occurred on a connection or if the output buffer is currently waiting for an acknowledge of the receiver side. The media interfaces report about fragmented network packets and necessary retransmission requests. All this information may point to possible connection problems which may have caused a DAQ failure.

The merging of data streams also combines the error and status word sent by all front-ends. The information which front-end sent which message is not conserved. Nonetheless, the most important parts of error information are stored in the hubs for each port separately for later analysis.

Last but not least, a board tracking feature can be used to investigate the full data path from the central hub to any front-end. This mechanism is further described in section D.2. The total amount of status and monitoring features sums up to more than 100 registers in a 12-port network hub. The full list can be found in section C.2.

4. The Trigger Sequence

The first important step of acquiring data in any triggered experimental setup is the generation and distribution of the trigger signal to all sub-systems.

The HADES data acquisition system uses a trigger-busy-release architecture. In particular, for each recorded event, a trigger signal is generated, supplied with additional information and distributed to all systems. Then the triggering circuit is stopped until all sub-systems have acknowledged their capability to take the next event.

Additionally to precise timing, distributing information about the trigger type and handling of busy times of the front-ends is required. This sequence is described in the following sections.

4.1. Overview

The full HADES trigger sequence can be divided into five steps based on a normal trigger-busy-release architecture:

- 1. Trigger Generation** Analog circuits process information from start/veto detectors and the time-of-flight wall. Discriminators generate digital signals with configurable properties.
- 2. Trigger Decision** In the Central Trigger System (CTS), all trigger sources are evaluated and if all criteria are met, a positive trigger decision is made.
- 3. Reference Time Distribution** The CTS generates a reference time signal sent to all front-ends using a dedicated LVDS line.
- 4. LVL1 Trigger** The CTS sends a trigger information packet to all front-ends via optical links.
- 5. LVL1 Busy Release** The trigger information packet has to be acknowledged by all front-ends. Hereby a busy-release information is resembled which allows the CTS to send another trigger.

This sequence is used for all triggers that are supposed to take data synchronously from all sub-systems. Opposed to that there are also kinds of triggers which do not rely on precise timing information and which are not triggered by a physical event. This group consists of different calibration triggers for individual sub-systems for example.

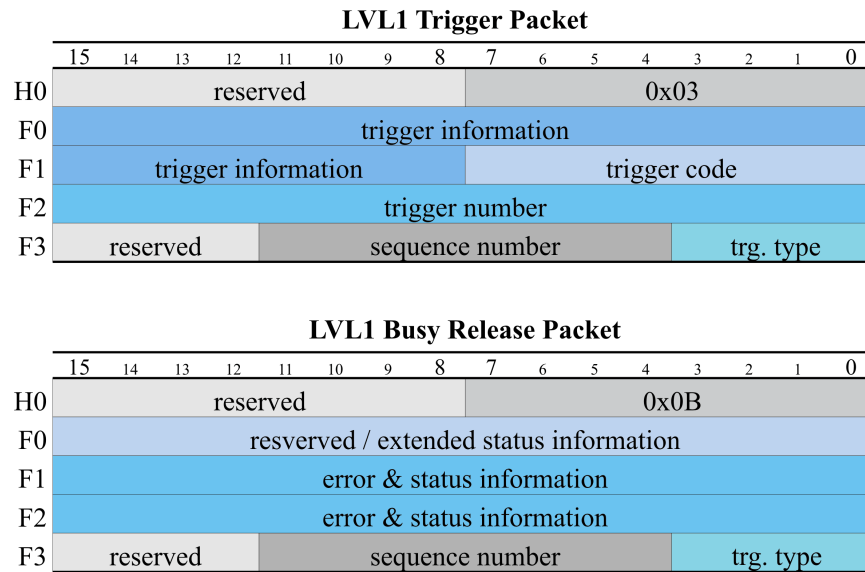


Figure 4.1.: The Central Trigger Systems sends a LVL1 trigger packet to all front-ends (upper part). This packet is answered with a LVL1 busy release packet (lower part) by all front-ends

Since no timing information is needed in this case, the first three steps of the trigger sequence are skipped and only steps four to five are executed.

4.2. LVL1 Trigger

The communication between CTS and front-ends related to the LVL1 trigger is encoded in two network packets that contain all necessary information as shown in figure 4.1. The LVL1 trigger is sent to all front-ends as a broadcast making use of the short-transfer feature of TrbNet [40].

The data transported with the LVL1 trigger contains the following information:

Trigger Number Each trigger is marked by a 16 Bit trigger number. This number is used to identify events during the read-out process.

Trigger Type The HADES Trigger System foresees sixteen different trigger types of which five are currently in use. Trigger types 0x0 to 0x7 are reserved for physical triggers that are sent in combination with a correlated reference time, while the rest is reserved for reference-time-less triggers (RTL-trigger). The complete list of assigned trigger types is shown in table 4.1.

Trigger Information Additional information about the desired front-end behavior is encoded

Trigger	Description
0x1	Physics Trigger
0x9	MDC Calibration Trigger
0xA	Shower Calibration Trigger
0xB	Shower Pedestal Trigger
0xE	Status Information Trigger

Table 4.1.: LVL1 Trigger Types. In total, 16 trigger types are available. One is used to mark physical events while the other four are special purpose triggers.

Bit	Description
0	Discard Data. If this bit is set, all front-ends will discard their data. The read-out request for this event will be answered with an empty data packet.
1 – 6	Unassigned
7	RTL Trigger. This bit is used to additionally mark any trigger that was not preceded by a reference time signal.
8 – 10	RICH Processing Type. Selects the type of data pre-processing in RICH front-ends. See table 4.3
11 – 16	Unassigned
17	TRB TDC. Suppress sending all headers generated by the TDC chips on the TRB
18 – 23	Unassigned

Table 4.2.: The LVL1 Trigger Information bits are used to configure the front-end behavior on an event-by-event basis. The complete trigger information is formed by 24 bit of data.

in 24 bit of trigger information. Here, the CTS can request that data should be processed but not stored for read-out. This mode can be used to run detector tests with automatic data analysis on the front-ends where no data has to be stored for further processing. For some sub-systems the kind of pre-processing done can be selected as well on an event-by-event basis [53]. The list of assigned trigger information bits is shown in table 4.2.

Trigger Code The LVL1 trigger packet is further complemented by a 8 bit random code. This code is used during the read-out phase to determine if all sub-systems deliver data that belongs to the right event. Compared to a sequential counter a random value adds a bigger amount of security since it can not be aligned or misaligned due to a simple

Value	Name	Description
0	Raw128	Raw data from all 128 channels
1	Ped128	Pedestal corrected data from all 128 channels
2	Ped128Thr	Pedestal corrected and threshold selected data from 128 channels
3	Raw64	Raw data from all 64 physics channels
4	NC64Ped64	Baseline corrected data from physics channels and pedestal corrected data from auxiliary channels
5	NC64	Baseline corrected data from physics channels
6	NC64Good	Baseline corrected data from physics channels with under- and overflows removed
7	NC64Thr	Normal Mode: Baseline corrected data from physics channels with under- and overflows removed and thresholds applied

Table 4.3.: The LVL1 Trigger Information Bits 10 to 8 are selecting the data pre-processing mode for RICH front-ends [53].

counting error.

The busy release packet consists mainly of a 32 Bit wide field of status and error information from the front-ends. These packets are sent by each front-end and merged inside the network hubs using a common-or logic. That is, the CTS receives a set error flag if at least one front-end reported the corresponding error. The contents of the field are explained in section 4.6.

4.3. Trigger Generation and Distribution

The Central Trigger System is the component that has control over the full data acquisition system. Here, detector signals are evaluated, triggers are generated and the read-out process is controlled.

The CTS consists of a central FPGA-based board which is supplemented by several analog boards that provide data from the detector inputs.

The central trigger system provides a total of 36 inputs from the different sub-systems:

- 16 channels are receiving data from each of the channels of the Start and Veto detector.
- 12 channels are connected to signals from each individual sector of both TOF and RPC detectors.
- 8 channels (“physics triggers”) can be connected to any other trigger source such as external pulsers or more sophisticated trigger signals.

Chapter 4. The Trigger Sequence

The signals from all detectors are first converted to digital pulses by the front-end electronics and then connected to the trigger logic. The physics triggers are usually preprocessed by a dedicated electronics component. E.g., analog signals from the front-ends are put to an analog summing unit. The height of the resulting signal is directly proportional to the amount of hits that were detected. This signal is fed to a discriminator with configureable thresholds which generates a trigger signal if the multiplicity was above the threshold.

Currently, three different threshold settings corresponding to three different multiplicity levels are provided. Additionally, a signal from the forward hodoscope can be selected as trigger source.

In the central trigger logic, all signals are first put through a signal shaper stage. The incoming signal is usually of varying length so that it has to be converted into a signal of defined length. This is provided by a sampling logic running at 800 MHz, corresponding to a time resolution of 1.25 ns, and an edge detection circuit. The resulting signal can then be further processed.

Most trigger decisions rely on coincidence measurements between different signals so that the different length of cables and signal latency has to be taken into account. Individual delays with a granularity of 1.25 ns can be applied to each channel. The logic has to allow for the signals to arrive with a changing delay as well. This is accomplished by again changing the width of each signal to a pulse with a length corresponding to the desired coincidence window.

In many read-out scenarios, it is interesting to accept different triggers with different selection criteria. Often, one is occurring more seldomly than the other but an equal distribution is required. Hence, a downscaling logic is implemented that allows to accept only a fraction of the signals from one channel.

The analysis of experimental data requires precise knowledge about the total numbers of events, no matter if they led to a positive trigger decision or the hardware was able to read-out the corresponding data. For this reason, the CTS features a variety of signal counters to track the amount of pulses on each input and in each of the intermediate processing steps.

Several triggers can be generated internally based on the inputs from all six sectors of the RPC and TOF detectors. The multiplicity generation is based on the number of sectors which showed a signal. Additionally, a limitation on non-neighboring or opposite sectors can be requested.

All multiplicity and physics trigger signals can be gated with a anti-coincidence signal generated from Start and Veto detectors, i.e. it is required that the Start detector showed a signal but no signal was seen in the Veto detector.

An internal trigger source can be used to generate triggers without any external signal input. Last but not least, the CTS generates triggers for all special trigger types like calibration triggers and status events.

The implementation of the trigger logic is fully configurable during run-time. All delays,

widths and down-scaling factors can be adjusted via slow-control. The inputs and trigger paths can be enabled and disabled individually as well. The full description of functions implemented in the CTS can be found in [54].

The total latency from a trigger input signal until the final trigger is sent to all sub-systems is between 140 ns and 180 ns [55], depending on the selected sources and not accounting for additional delays added on the inputs.

The reference time signal is formed by a positive pulse of 105 ns length. This pulse is distributed to all sub-systems using the LVDS or PECL signaling standard. Here, twisted pair as well as Cat-6e cables are installed. In combination with dedicated fan-out devices to distribute the signal, this results in a high immunity against electromagnetic noise and a very low jitter below 40 ps [56]. The distribution of trigger signals is described in further detail in section B.

At the same time of sending the reference time signal, the LVL1 trigger is prepared and sent to all front-ends via the optical network.

4.4. Trigger Handler

One of the most crucial points in the trigger system is the correct validation of the reference time by all front-ends. Therefore, all possible error scenarios have to be checked by each front-end. The corresponding logic is placed inside the network endpoint and is identical for all sub-systems.

The reference time signal is directly forwarded to the front-end electronics to keep the precise timing information. In parallel, the signal is evaluated by the LVL1 Trigger Handler and checked for correctness. The read-out logic is informed of any possible error on the signal that might have an influence on the event data.

The general rule of thumb how to handle errors is “Data sent by the CTS is always right, but there might be errors on the reference time introduced by noise or wrong connections”. This assumption is valid since the CTS trigger logic has been tested in great depth and the data transmission on optical links is more immune against noise than a differential signal transported over ten or more meters on partly unshielded twisted pair cables.

The necessary checks that have to be done by the LVL1 Trigger Handler can be summarized as follows:

- The length of a reference time signal is defined to be at least 100 ns. Therefore, all signals shorter than 60 ns are to be rejected as noise while longer signals should be accepted as valid reference time signals
- The reference time signal must not be longer than 200 ns. If this is the case, there are two possible error sources: The timing signal is not connected due to a mechanical failure in the connector or the polarity of the signal is inverted. In both cases, this

Chapter 4. The Trigger Sequence

status has to be reported to the monitoring system via slow control. If the problem is an inverted timing input, the signal might be inverted using a configuration register.

- The TrbNet protocol ensures that the trigger information is transported as fast as possible and arrives at every front-end within 5 μs , typically. If the time between the reference time signal and the LVL1 Trigger exceeds 20 μs , the timing signal can be treated as erroneous.
- For each reference time signal there must be a corresponding LVL1 Trigger.
- A RTL-trigger must not be preceded by a reference time signal.

The reaction of the LVL1 Trigger Handler in case one of these rules is not matched has to respect the typical behavior of the front-end electronics. If there is a too short reference time signal, for example, the front-end electronics might already be triggered and the read-out cycle has to be correctly finished by the read-out controller. In any case, the LVL1 Trigger Handler only checks the inputs from the CTS and informs the read-out logic about possible errors. The read-out logic has to make sure that the correct, front-end dependent procedures are carried out.

In total, we can distinguish between two normal trigger cases and six error types as listed below.

Case 1: Valid Timing Trigger

The normal physics trigger sequence consists of a pulse of at least 100 ns length on the reference time input, followed by the LVL1 Trigger about 2 to 5 μs later. The trigger type must be in the range between 0x0 and 0x7 and the trigger information must indicate a physics trigger.

The LVL1 Trigger Handler accepts the reference time signal if it has a length of at least 60 ns. Typically one clock cycle after this condition has been met, the Trigger Handler signals a valid physics trigger to the read-out logic.

Case 2: Valid reference-time-less (RTL) trigger

The RTL-trigger consists of a LVL1 Trigger only. This must contain a trigger type greater than 0x7 and a trigger information indicating a RTL-trigger. Additionally, there must not be any signal on the reference input before.

If these criteria are met, the Trigger Handler signals a valid RTL-trigger to the read-out logic about one clock cycle after the LVL1 Trigger arrives.

Case 3: RTL-trigger preceded by a reference time signal

In case of noise on the reference time input, a RTL-trigger might be preceded with a valid reference time signal. In this case, the LVL1 Trigger Handler has already

accepted the reference time as valid physics trigger according to case 1 when the RTL-trigger information arrives. As soon as the RTL-trigger is received, the trigger handler announces the accepted reference time to be a spurious timing signal and additionally reports a valid RTL-trigger.

The behavior of the read-out controller depends on the capabilities of the specific front-end electronics. If possible, the read-out sequence should be interrupted, data collected so far should be discarded and the requested RTL-trigger should be carried out. Nevertheless, this ideal behavior is not possible for most front-ends since usually the data is directly stored in the event buffer from where it can not be deleted any more. In that case, the RTL-trigger must either be skipped (if the read-out controller already finished writing data) or the RTL-trigger event data might be added to the erroneously taken normal event data.

In any case, the event is marked as possibly corrupted in the data stream automatically.

Case 4: Timing Trigger without a reference time signal

In some cases, the reference time signal might be lost and not detected by the LVL1 Trigger Handler. Even though the front-end electronics are not triggered in this case, it might be necessary for the read-out logic to take measures to keep trigger numbers in sync or to reset the front-end electronics. Thus, the read-out logic is notified of an invalid trigger when the trigger information arrives.

Case 5: Multiple reference time signals before a LVL1 Trigger

If the LVL1 Handler detects several valid pulses on the reference time input before a LVL1 Trigger arrives, only the first seen signal results in a valid trigger signal to the read-out controller. With the first subsequent timing signal, the read-out is only informed of the multiple signals and the corresponding error information is set automatically.

Case 6: Too short reference time

A short pulse on the reference time input which is not treated as valid (i.e. in case it is shorter than 60 ns) is announced as a spike to the read-out logic. Opposed to the read-out logic which uses the filtered trigger valid signal from the LVL1 Trigger Handler, the front-end might already have triggered and collected data. The correct handling of this situation is up to the read-out logic.

Case 7: Too long delay after reference time

The network transport protocol ensures that the LVL1 Trigger arrives within a short time after the reference time signal. Typically, this time is less than 5 μs but might be slightly longer when there is heavy traffic on the network. If the trigger information did not arrive within 20 μs after the timing signal, it is likely that it was not a real

Chapter 4. The Trigger Sequence

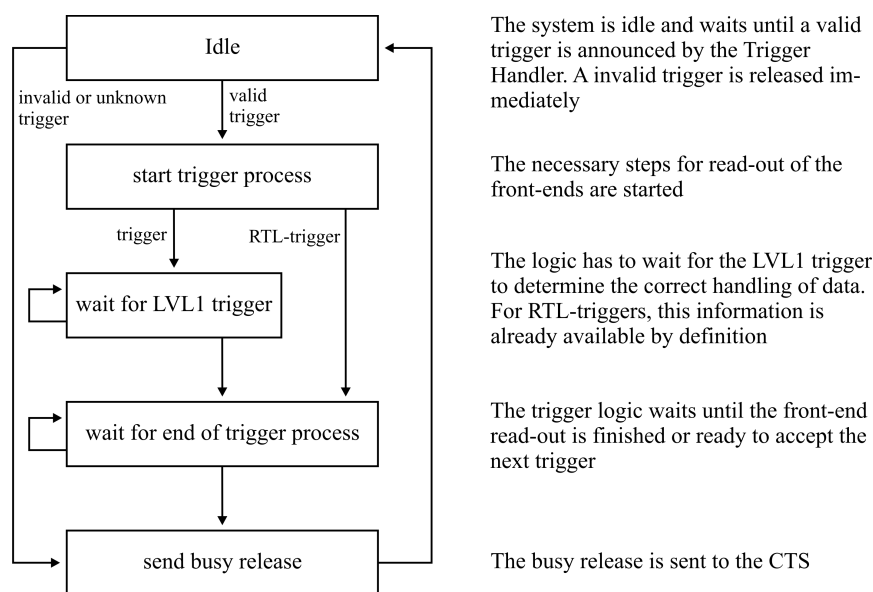


Figure 4.2.: The typical trigger handling algorithm that has to be implemented in the front-end logic.

signal. Since the read-out already started, the logic is informed about the time-out and the corresponding error bits are set in both the LVL1 and data channel.

Case 8: Long signal on reference time

The signal on the reference time input might be much longer than a normal trigger. Usually this happens if either the input is floating without a defined voltage level or the polarity of the input is inverted. In both cases, the rising edge of the signal is accepted as a valid timing signal. If the signal stays asserted for more than 1 μ s an error flag is assigned to show this state. The front-end logic has to acknowledge the accepted trigger in the usual way. The error condition is marked in both the LVL1 and the data channel error information.

The typical algorithm implemented in the front-end logic to handle triggers is depicted in figure 4.2.

4.5. Trigger Interface

The interface to the user consists of several individual ports. Three signals are used to inform the front-end logic about the different types of valid triggers as well as invalid triggers. These signals trigger the data taking process in the front-end logic.

Additionally, six signals inform the front-end logic about all problems and errors that have been detected by the Trigger Handler as described in section 4.4.

4.6. LVL1 Trigger Error and Status Information

All information sent by the CTS within the LVL1 trigger is given to the front-end logic as soon as it is available. The delay between the reference time and the arrival of the LVL1 trigger is usually between 2 μ s and 5 μ s depending on the sub-system. Here, the number of intermediate network hubs and the types of data links is relevant while the length of cables has a minor contribution only (see section 8.1 for detailed performance numbers).

Each valid or invalid trigger has to be acknowledged as soon as the logic is able to process the next trigger (“busy release”). Parallel to this signal, a 32 Bit word containing status and error is given to the trigger handler (see section 4.6).

The full interface between the trigger handler and the front-end logic is described in appendix A.2.

4.6. LVL1 Trigger Error and Status Information

The network packet sent by the Trigger Handler as busy release signal contains a 32 bit word with error and status information of the trigger logic. Here, all error conditions of either the LVL1 Trigger Handler or the front-end control logic are marked and information about buffer fill levels is sent to the CTS. Most information is generated automatically by the trigger and data handling logic, but all information can also be set by the front-end logic. The individual bits of the status information are defined as described below. (The lower 16 Bit are used for network information and are described in sect

Bit 16: Trigger counter mismatch

The trigger handler contains an internal counter of received triggers while the LVL1 trigger sent by the CTS contains a trigger number as well. These two values are forced to be synchronized to detect possible losses of trigger signals. If a discrepancy is found, a trigger counter mismatch is reported.

Bit 17: Reference time missing

A normal trigger consists of both a reference time signal and the LVL1 trigger. If a LVL1 trigger is received without a preceding reference time, this is reported to the CTS.

Bit 18: Multiple reference times

If noise is introduced on the reference time signal, the trigger logic detects several reference time signals. Since it can not be guaranteed that the signal actually sent by the CTS has been used for data taking, this event might be corrupted.

Bit 20: Buffers half filled

The data buffers in the front-end can store a certain number of data words before read-out must take place. If read-out is delayed further, the front-end will block triggers until the

buffer is freed. This might result in a performance loss. Hence, the CTS is informed about the fill-level of buffers and might take appropriate measures, e.g. lower the trigger rate.

Bit 22: Front-end not configured

Due to an error during the initialization phase, a front-end was not configured correctly and is not able to record data for this event. Either the configuration can be repeated by an automatic control logic in the front-end, or the initialization sequence has to be repeated.

Bit 23: Front-end error

Data recorded during an event can be corrupted as a result of various types of errors: The answer from a front-end module can be missing, it might be not synchronized or an electronic component failed. Such errors are reported as front-end error.

Bit 24: Spike on reference time detected

The reference time signal can be corrupted by short spikes that can have an influence on the proper function of front-ends. This is detected by the LVL1 trigger handler and reported to the CTS.

Bit 25: Trigger Time-out

TrbNet guarantees a maximal latency of the LVL1 trigger after the reference time of about 5 μ s . If this time is exceeded, the recorded reference time is likely to be invalid.

Bit 26: Data missing or buffer overflow

The read-out logic is usually able to handle a limited number of data words only. In case of misconfiguration or failure, this limit might be exceeded and subsequent data are lost.

5. The Read-out Chain

When receiving a trigger signal, all sub-systems perform the read-out of the front-ends and store the data in internal buffers until they can be sent to the data processing servers .

This second step of the data acquisition process is controlled by the CTS in the same manner as trigger distribution. First, a request is sent to all front-ends, which respond with their data with a final termination packet. While the termination packet is transported back to the CTS, all data is stripped by the Sub-Event Builders located in the network hubs and forwarded to the server farm via Gigabit Ethernet.

During this process, all necessary header words are generated to match the data to the HADES data format described in section 5.1. The following sections show the full read-out process starting with the interface to send data in the front-ends and the CTS read-out request. Data is forwarded by the network hubs where additional data words are generated and data is sent to the server farm. Finally, the error and status information contained within the data stream is explained in section 5.8.

5.1. Read-out Data Format

The HADES software tool-chain for data recording and analysis is based on a fixed data format [57]. Every data file consists of a set of Events, which are divided into a number of Sub-Events that might be further sub-divided into Sub-Sub-Events (SSE, see figure 5.1 for an example). This structure was proved to be well suited throughout all levels of data processing and does not need to be changed significantly for the upgraded DAQ system.

Each logical block within the file starts with a header as shown in figures 5.2 and 5.3. Each header is generated in a different place within the DAQ system. The event header contains the basic information about an event and is written by the Event Builders after merging all sub-events.

The sub-event header and data structures below are generated within the DAQ network. Here, the straight-forward implementation would be that all front-ends generate their own sub-event header with all relevant information. Due to the vast amount of header data produced by all 500 front-ends, this is not feasible. Hence, sub-events are generated for sub-sections of the detector only. More precisely, sub-event headers are written by the network hubs that also serve as bridge to Gigabit Ethernet. In total this amounts to 27 sub-events for each event.

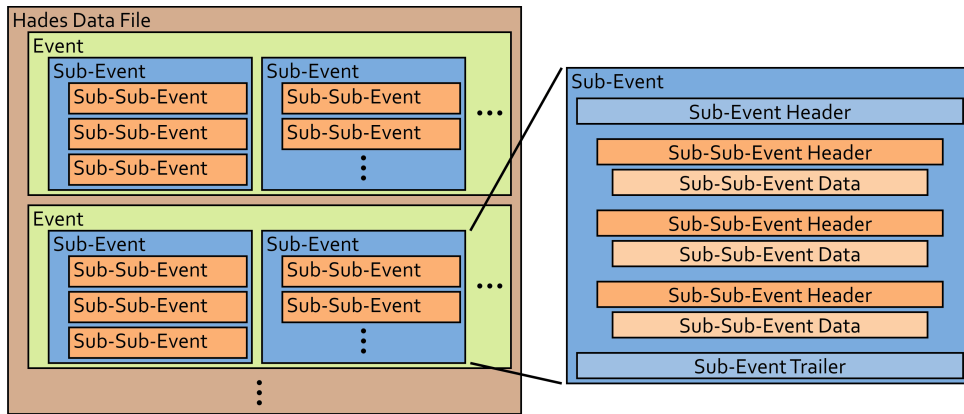


Figure 5.1.: The HADES Data Format. The left figure shows a typical HADES data file (hld-file). It consists of events which are further divided into sub-events and sub-sub-events. A more detailed view of a sub-event containing header, data and trailer words is shown in the right sketch.

The last two words of each Sub-Event are formed by a trailer that contains the status and error information (see table 5.3). The content of the error information is explained in section 5.8.

Nonetheless addresses from each front-end are needed for correct interpretation of the data. This information is given in a compact sub-sub-event header (SSE-Hdr). To simplify the unpacking process every header contains a field with the length of the corresponding data block. It should be mentioned, that this header does not need to be generated by the front-ends by default but could be added later by the sub-event builder. This would reduce the data bandwidth needed within the DAQ network but also makes generation of the sub-event headers much more complicated: The HADES data format requires that the length of each data block is known in advance. Inside network hubs this information can not be generated since data is processed serially and not stored. Hence, length information is only available after the full data stream has already been forwarded to the next network node. Opposed to that, the full event is stored in the front-end modules and can easily be equipped with length information in the header. Data integrity, especially if all sub-events belong to the same event, can be checked based on the trigger number and code provided within each header as well.

5.2. Event Data Interface

All data generated with an event is written to the Endpoint Data Handler through the Event Data Interface (EDI). The layout of this interface is described in table A.6. Writing data is allowed after a valid-trigger strobe signal until the user finalizes the transfer by assigning the

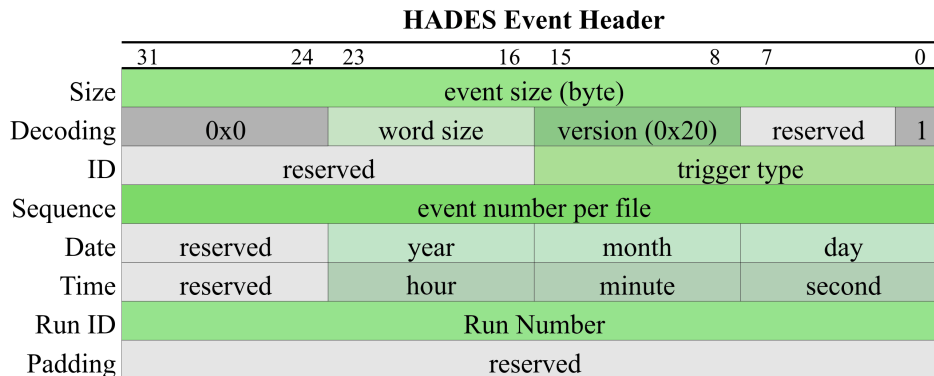


Figure 5.2.: The HADES Event Header. Each event stored on disk is preceded by an event header. It contains all information that is necessary to analyze the data contained, i.e. the size, the exact time of recording and a unique event number.

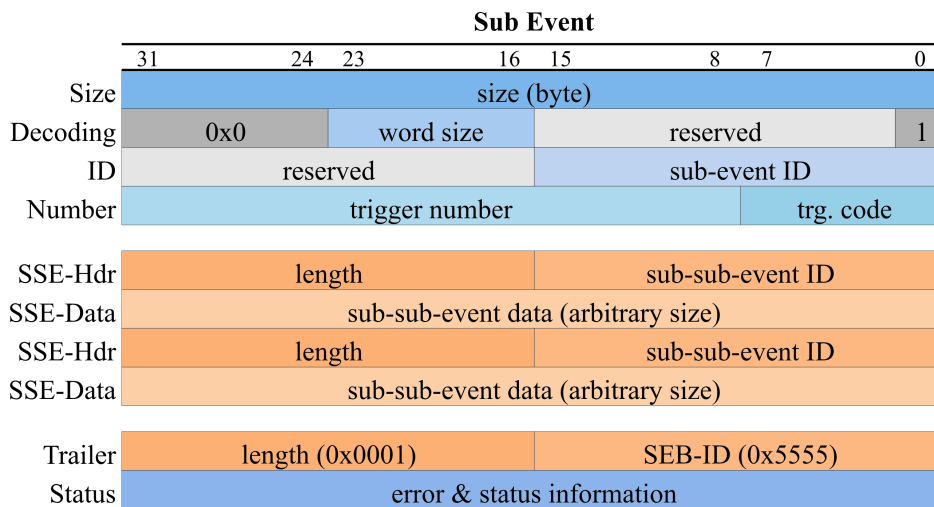


Figure 5.3.: The HADES Sub-Event. The four-word sub-event header precedes each data block sent to Event-Builders and written to storage. The end of a sub-event is formed by a trailer that contains the error information transported in the termination word of the TrbNet protocol. A typical HADES event consists of about 30 sub-events. Each sub-event contains an arbitrary number of sub-sub-events. A sub-sub-event starts with a one-word header (SSE-Hdr) followed by data.

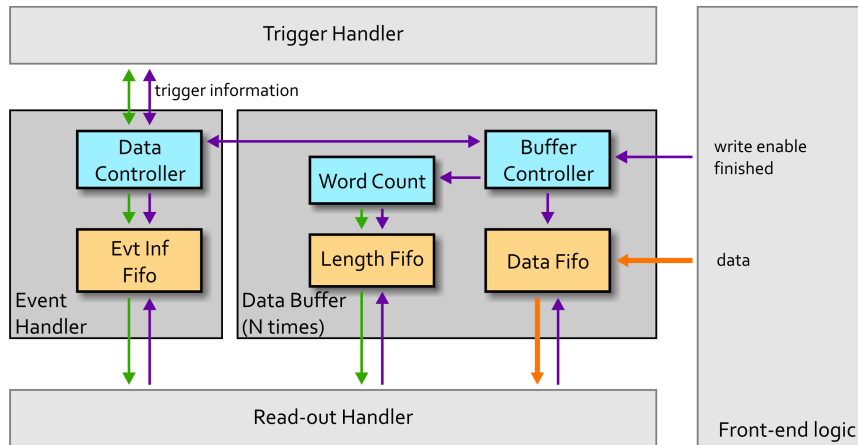


Figure 5.4.: A schematic view of the Endpoint Data Handler with its logical blocks.

finish signal. An example timing diagram of one valid event is shown in figure A.7.

The data format used is not fixed and can be defined arbitrarily based on the requirements of the specific front-end. Nevertheless it is suggested that the data format contains a header defining the data format and forseees to send debug information along with the data if requested by the CTS.

The data handler is able to control a configurable number of Event Data Interfaces in parallel. Thus, in case that the FEE read-out logic consists of several parts running independently, each can fill the data in a separate buffer. A synchronization between the different input channels is not necessary. Each single input can terminate writing data to the buffer whenever it is finished. This operation mode is used e.g. in the pre-Shower FEE were the read-out of the 12 ADC interfaces is done independently from each other.

Depending on the configuration of the data buffer, the LVL1 busy release signal is either controlled by the front-end logic directly or the release is delayed until writing data to the buffers has been finished. In both cases, the busy release is further delayed if the fill-level of one of the buffers is above the critical threshold.

The correct setting for the buffer threshold depends on the operation mode: If the busy is configured to be released after all data has been written to the buffers, the buffers must be able to store at least one more maximum sized event. If the busy is allowed to be released before writing data has been finished, the threshold must be at least two maximal sized events below the buffer size because the fill-level after the current event is finished is not known.

The size of buffers can be configured in the Data Handler. Combined with knowledge about the maximal event size generated by the front-end, all buffer handling can be done internally in the Data Handler. The full list of configuration options is shown in table A.7.

In parallel, hidden from the event data interface, the data handler stores all trigger information necessary to perform the read-out. These data include the trigger number, possible

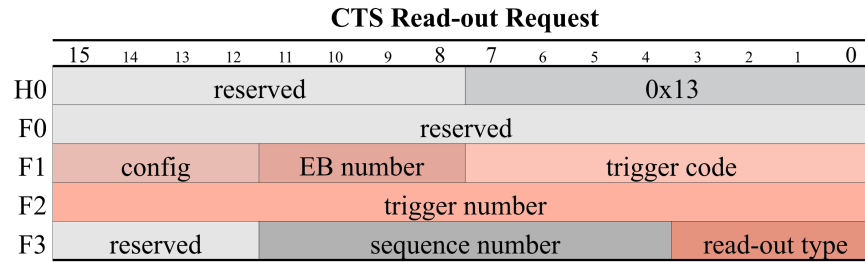


Figure 5.5.: The Central Trigger Systems sends a read-out request packet containing trigger number, read-out information and event random code.

trigger information bits and the random trigger code. The full Endpoint Data Handler is shown in figure 5.4.

5.3. CTS Read-out Request

The information contained in the request is shown in table 5.5. The random code and trigger number sent with the request are used to identify the event in the buffer and to prevent event mixing by checking this information in every network node that transports the data.

The read-out information contains configuration for the read-out chain, e.g. to which Eventbuilder the data are supposed to be sent. The read-out type may be used to select between different read-out methods in later applications, but is not used in the HADES DAQ system.

5.4. Read-out Handler

The data are kept in the buffers until a read-out request from the Central Trigger System arrives. As soon as the Read-out Handler receives the request, it reads event information and data length from the buffer fifos. From these information, two header words are generated. The first one (Event Information) contains the event number, trigger type and the event random code as read from the buffer fifo. Additionally, one bit (“Pack-Bit”) is used to determine how hubs should treat this data (see section 5.5).

The second word is the sub-sub-event header that contains the length of event data counted in 32 bit words and the address of the endpoint. Afterward, data from all event data buffers is retrieved and sent. The full data structure is shown in figure 5.7. The TrbNet termination word sent in the end of each transfer contains additional status and error information about the event as described in section 5.8. The full read-out data stream generation is shown in figure 5.6.

The front-end sends the event information, data header and status information for each

Chapter 5. The Read-out Chain

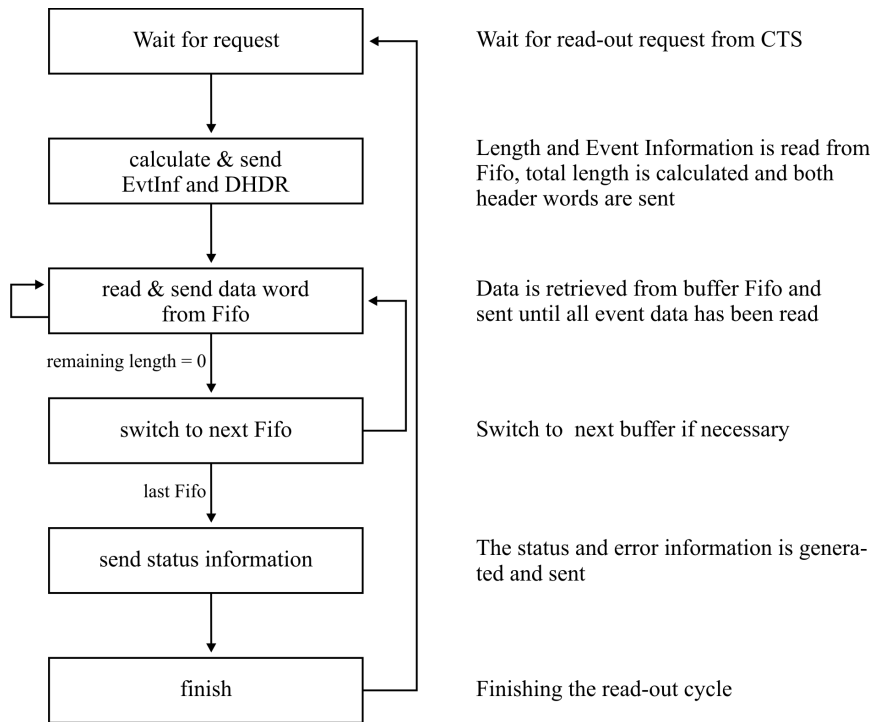


Figure 5.6.: A schematic view of the read-out data stream generation in the Read-out Handler.

event. It should be mentioned that this is done also when there is no event data available, e.g. because no hit was detected.

5.5. Data Read-out Hub

The network hubs merge data from all connected down-links into one stream sent out on the up-link. On the data channel, also a small amount of processing takes place. The event information from all down-links are read and compared against the values in the read-out request. Since during normal operation all event headers are identical, the duplicates do not contain any additional information and can be discarded.

Regarding the sub-sub-event header, there are two possible processing scenarios: First, the SSEhdr from all data streams may be kept and forwarded (non-packing mode). Secondly, the SSEhdr may be processed and removed (packing mode). The first method may be used if the network address contained in this word is necessary to correctly unpack the data during analysis. The second one can be chosen if the address information is not necessary for any further processing steps.

Which of the two operating modes is chosen is decided based on the “pack-bit” received in the event information word: If this bit is set in all received data streams, then the packing is enabled.

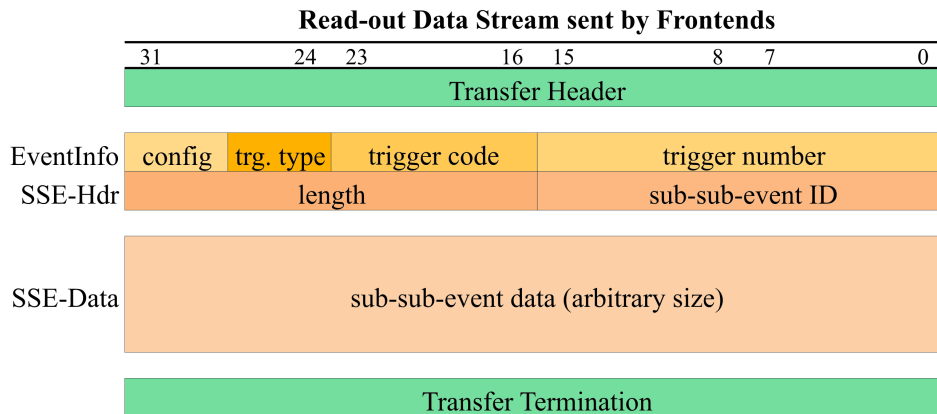


Figure 5.7.: The data stream generated by the Read-out Handler. The first two words are added by the handler based on trigger information and the number of data words written by the front-end. The further data words are directly forwarded from the data buffer.

In the HADES DAQ system, all front-ends send data marked not to be packed. Thus, their address information is kept in the first stage of network hubs. The hub itself adds its own SSEhdr with its own address and the total data length in front of the first front-end SSEhdr. Here, the pack-bit is set to allow all subsequent hubs to merge the headers in subsequent hubs.

The mixing of both operation modes has an advantage: The final data structure during analysis does not change when adding an additional hub the data passes through: Independently from the network setup, the data contains SSEhdr from the front-ends as well as from the last hub that processed the data. All other hubs remain invisible to the the data stream.

After sending the event information and data header, all incoming data is forwarded in a round-robin order. The resulting data stream from a hub with two connected front-ends in non-packing mode is shown in table 5.9, the data structure in packing mode is shown in table 5.10. As a last step, the error and status information from all data streams is merged together with own error bits from the hub logic itself.

If desired, the operation mode may be switched to packing mode already in the first layer of hubs thus removing the SSEhdr sent by front-ends. This can significantly reduce the amount of data. Since the HADES system comprises of close to 500 front-ends each sending a SSEhdr, the amount of data per event might be reduced by up to 2 kiB. This corresponds to about 15% in a high multiplicity event or more than 30% for elementary proton-proton collisions.

In the current data formats, most detectors depend on the information sent in the SSEhdr by front-ends. Only the RICH data format would allow to remove the length/source information from the front-ends but it is kept to keep the data structure identical for all sub-systems.

Chapter 5. The Read-out Chain

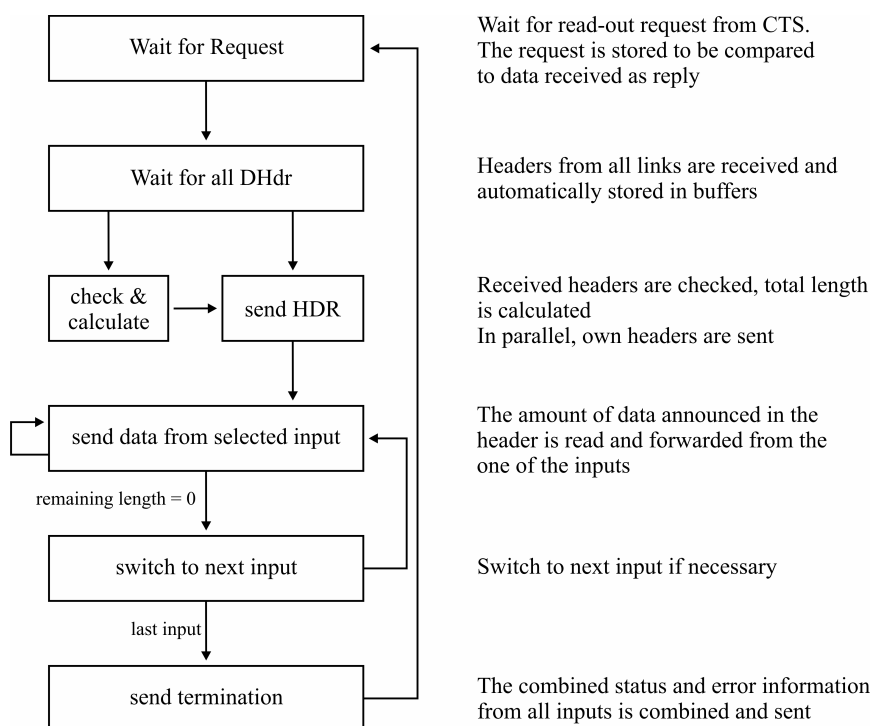


Figure 5.8.: A schematic view of the processing of event data streams in the hub logic.

The first argument is, that the overhead generated by these words is negligible for the RICH detector. A second point is, that the SubEventBuilder has to be able to add its own data words to the end of the data stream and therefore needs to put its own length/source information to the data stream. Without this SSEhdr word it would not be possible to detect the end of event data and the start of SubEventBuilder information.

Debugging Information from Hubs

To send debugging information, the hubs are allowed to add an extra SSEhdr and an arbitrary number of data words to the end of each read-out data stream. This data will then appear as coming from another endpoint but with the network address from the hub. Clearly this feature can only be used when the hub is used in a non-packing mode, since otherwise the hub debug data can not be detected properly. Even though this feature is foreseen in the protocol, it is not in use in the HADES DAQ system.

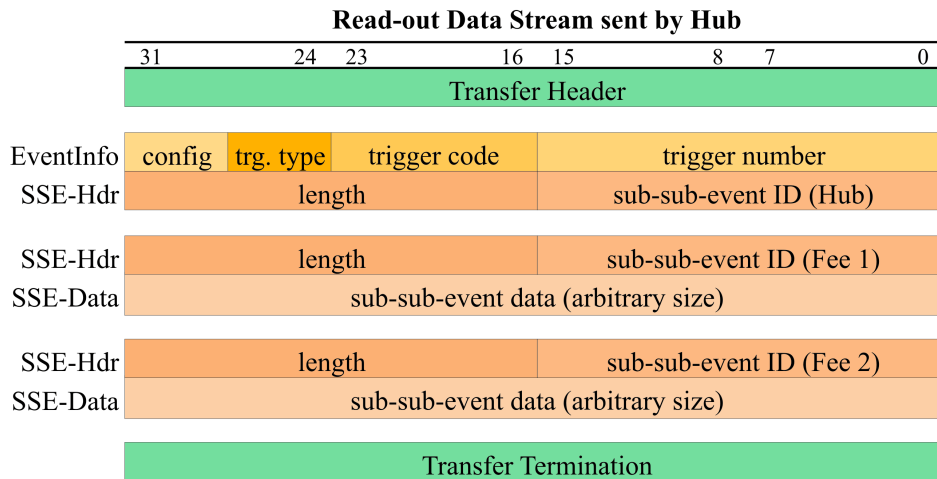


Figure 5.9.: The resulting data stream after a hub merged two incoming data streams in non-packing mode.

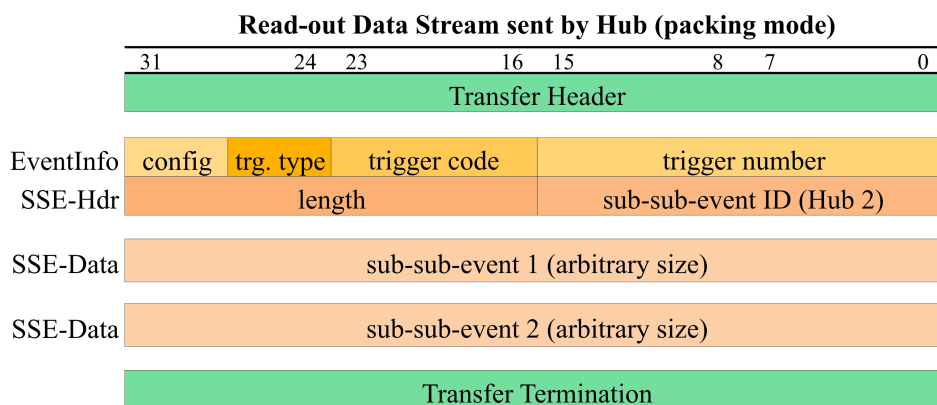


Figure 5.10.: The resulting data stream after a hub merged two incoming data streams in packing mode.

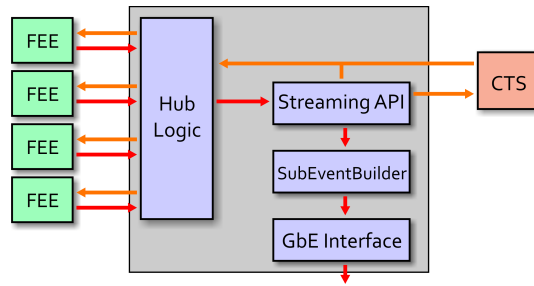


Figure 5.11.: A typical application of the streaming API. Embedded in a network hub (gray box), it provides the interface to extract data from a TrbNet read-out process and to send it over Gigabit Ethernet to the server farm.

5.6. Streaming API, Sub-Event Builder and Gigabit Ethernet Interface

In a certain place, all transported data have to be bridged from TrbNet to another transport medium like Gigabit Ethernet. For this purpose, the Streaming API has been developed. It allows the read-out request to pass through from the CTS to the front-ends. The read-out data stream, on the other hand, is received and forwarded on a data port. The logic connected to this port now can process the data in any desired way. For example, all data can be sent using another medium. A second port allows to send an empty read-out data stream or a full, processed data stream further on to the CTS. A schematic view of the setup is shown in figure 5.11

Typically, this bridge between TrbNet and GbE is placed in the same FPGA as the hub that merges the data. All ports of the interface are described in tables A.10, A.11 and A.12. A typical transaction on this interface is shown in the timing diagrams A.10, A.11 and A.12.

The Sub-Event Builder is the entity which converts data from the internal TrbNet format (see section 5.4) to the standardized HADES Sub-Event format [57] as described in section 5.1.

Both data formats contain essentially the same information and can easily be converted from one format to the other. The rest of event data can be transported without any change. This permits the read-out to be performed in a streaming mode which can be efficiently processed in programmable hardware.

The resulting sub-events are stored in a Fifo buffer before they are further processed. The further transports requires one or more sub-events being packed into a Hades-Transport-Queue of up to 64 kByte size. For transport over Gigabit Ethernet these data packets have to be split into Ethernet frames of up to 1.5 kByte/s each. An IP-core provided by Lattice Semiconductors forms the low-level interface to the GbE network and is used to transport the final data packets. A detailed description of the data processing and transportation steps can

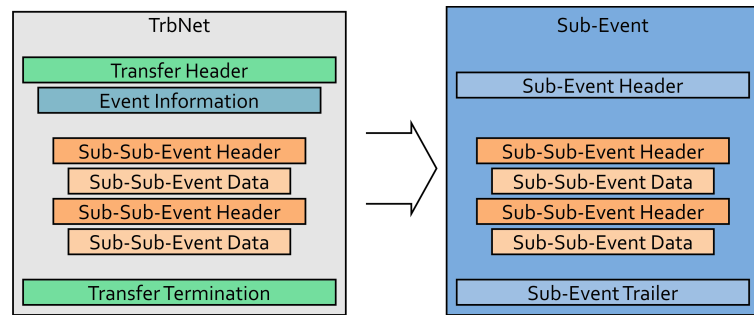


Figure 5.12.: The Sub-Event Builder converts the TrbNet data format to a HADES sub-event. The data is transported without changes while header and trailer words are regenerated.

be found in [58].

The target address the data packets are forwarded is stored in a configuration memory that can be altered via slow-control. A total of 16 different targets can be configured. The CTS decides to which event-builder data are to be sent by using a round-robin logic. Special event types can also be forwarded to selected event-builders, e.g. all calibration triggers can be collected in one data file [54].

5.7. The Event-Builders and Data Storage

The event data are transported using an industry-standard Gigabit Ethernet infrastructure to the Event-Builder server farm as shown in figure 2.15. Each server hosts up to four independent event-building processes [59]. Here, data from all sources are collected in a buffer memory until all sub-events of a given event have been received. Subsequently, the HADES Event Header is generated and data can be written to disk.

The local storage has a capacity of about 160 TB which is sufficient for several days of data recording. In parallel data is forwarded to the GSI computing center and stored on tape or in a distributed network file system (Lustre).

The event-building processes operate synchronously, i.e. one process acts as a master that controls all other instances. That is, all data files are started and closed at the same time which simplifies later data analysis. Information about date, time, file size and number of events is automatically stored in a database.

5.8. Read-out Error and Status Information

As on all TrbNet channels, the end of a transfer is formed by a termination word containing a 32 bit error and status information code (see section 3.2.4 for details). The 16 channel depend

words are selected to encode all information necessary for later analysis:

Bit 16: event number mismatch

In each data transport node (either a hub routing logic or the front-end endpoint), the trigger number is checked for validity by comparison to the event number requested by the CTS. If a mismatch is found, the correct event number is inserted and this Bit is set.

Bit 17: Trigger code mismatch

Like Bit 16, this bit shows if there was a mismatch in the upper 16 bits of the DHdr, i.e. either the event code or the trigger type does not match the one requested by the CTS.

Bit 18: Wrong length

The length of the data stream did not match the length given in the DHdr. The length is checked in both the front-end and every network hub. If the front-end detects a mismatch, this bit is set but data is forwarded without changes. If such a mismatch is detected in a hub, excess words are deleted, but no padding is added if words are missing.

Bit 19: Answer missing

This bit is foreseen to mark events, where one front-end failed to deliver any data or the reply contained no data. This functionality is currently not implemented.

Bit 20: Not found

The event number requested by the CTS was not available in the event data buffer on the front-end. The implementation in the front-end handlers checks the next event in the buffer and sends it in any case. Depending on the capabilities of the data buffer, other modes like a look-up through all events waiting for read-out may also be implemented.

Bit 21: Partially found

This bit is set by the data handler in case of a problem with a data buffer. The event was found and at least some data was sent. Due to an error condition, not all data could be delivered. This may be caused by a buffer overflow or a limitation in the maximum number of data words per events.

Bit 22: Severe problem

A generic error flag showing problems with the data buffer or the read-out. This bit is set by either the read-out logic, the data handler in the front-end or the network hubs in case of a problem that is not likely to vanish with the next read-out. If this bit is set, manual intervention is needed to restart and resynchronize the read-out process.

Bit 23: Single broken event

A generic error flag, marking an event as corrupted. In contrast to bit 22, only one event is affected and the read-out can be continued.

Bit 24: Ethernet link down

The Ethernet link which is configured to transport data to the eventbuilders is currently not working. Data has been either forwarded through TrbNet to the next network hub, or it was lost. Even though this information can not be written to the Hades Subevent it belongs to, the information is generated and transported to both other network hubs, which might have a configured and established ethernet connection, and to the CTS which can store the information. Based on this, an error message can be generated via slow control.

Bit 25: SubEvent buffer almost full

This status flag is used to transport the information about almost filled data buffers. This is not an error since the inherent back-pressure of the network protocol will delay further read-out requests until sufficient space for data is available. Nonetheless, the bit can be used for monitoring purposes.

Bit 26: Ethernet / SubEventBuilder error

Either the Ethernet interface or the SubEventBuilder found corrupted data in the received data stream. The result is either no data being sent for this event or data has been sent partially only.

Bit 27: Timing trigger error

The reference time signal showed some irregularities during the trigger process. Either the signal was missing, had spikes or multiple edges were seen and may have corrupted data.

6. The MDC Read-out System

One of the major parts of the HADES DAQ upgrade was the replacement of the MDC read-out system based on new optical technology. The original MDC Read-out system was designed almost 15 years ago. The reason for a re-design was the combination of the requirement for a higher read-out bandwidth for heavy-ion systems and to avoid the electromagnetic noise produced by the massive parallel read-out busses.

The new read-out systems employs FPGA based boards mounted directly onto the front-end electronics. All configuration and control features as well as voltage regulation are located on these boards. The geometry of the detector puts strong constraints on the size of the boards: Their footprint must be less than $4\text{ cm} \times 5\text{ cm}$ and the amount of cables has to be kept as small as possible. The parallel data read-out busses were replaced by a optical data transport to improve both speed and the electromagnetic noise environment.

This upgrade is fully described in [21]. In 2010 further work was done on the read-out, data reduction and monitoring structure which are described in detail in this chapter.

6.1. MDC DAQ Upgrade

The new MDC DAQ electronics consists of three parts: The old front-end electronics including signal amplifiers (“Daughterboards”) and shapers and the TDC circuitry (“Motherboards”) was kept. Here, a set of ASD-8 [60] ASICs amplify, shape and discriminate the signals of up to 96 detector channels. These signals are routed to TDCs [18] where all transitions are recorded and stored.

The electrical transceiver mounted on top of this board has been replaced by a new board, the OEP (Optical End-Point). The OEP is equipped with a FPGA (Lattice ECP2/M-20), voltage regulators and an optical transceiver. A schematic picture of the full MDC read-out system is given in figure 6.1. A more detailed view of the front-end electronics mounted on the frame of each MDC chamber is shown in figure 6.2.

The improvements of the new, optical read-out system are various:

- The data band-width for each front-end has been increased. The old system transported data from up to three chained front-end boards at a rate of 180 MBit/s while the new system runs at a speed of 250 MBit/s from each single front-end. The further parallelized read-out also caused a significant lower intrinsic dead-time compared to the old system [61].

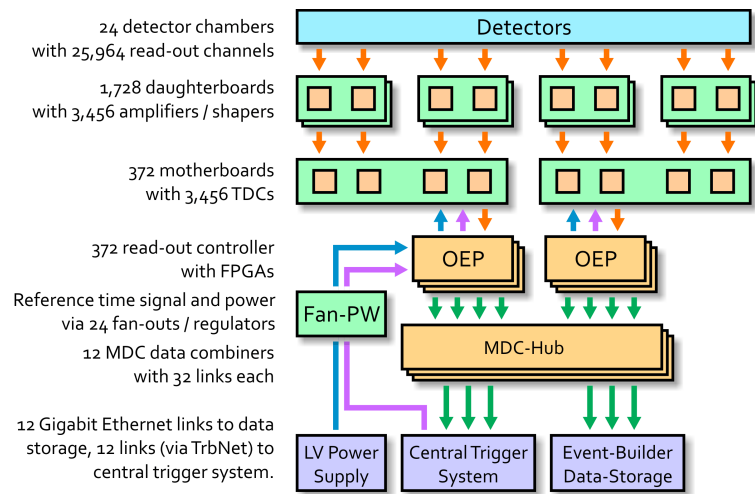


Figure 6.1.: A simplified view of the full MDC read-out system. Detectors are read-out via front-end electronics. Read-out is controlled by Optical Endpoints (OEP) and data are further transported to data combiners and the storage system. In parallel, trigger signals and the power supply is distributed over dedicated fan-out and voltage regulator boards.

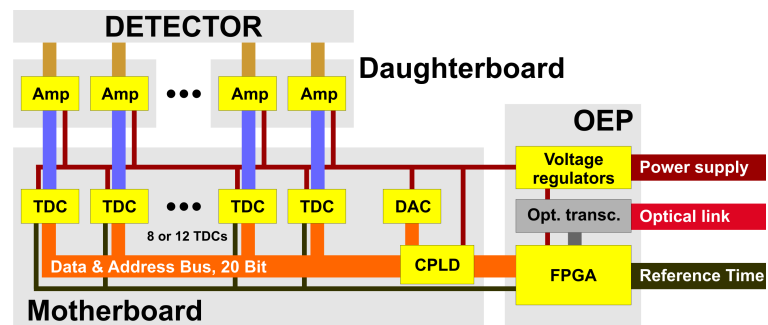


Figure 6.2.: A simplified view of the MDC front-end electronics. The signals from the detector are fed to a amplification and discrimination stage (“Daughterboards”). The resulting signals are measured by TDCs (“Motherboard”). The full setup is configured, monitored and read-out by a FPGA based logic (“OEP”).

- Transmission of data on optical fibers generates no electromagnetic noise. The noise produced in the old system by the wide read-out data busses led to an interleaved operation mode since it was not possible to transport data while the TDC were sensitive for signals. This caused a drop in both the achievable trigger rate and the maximal data rate. With the new system, both operations can be run independently from each other and fully in parallel.
- The amount of conducting connections between different parts of the system has been reduced. With the old system, there were many so-called ground loops, i.e. two connections of two points with the same electrical potential running over different paths. Such setups can act like antennas and as such pick up electromagnetic noise and cause fluctuations on power lines. With an optical read-out only the power distribution itself and the frame of the chamber remain as conductive connections.
- Even though more additional functionality has been packed onto the front-end boards, the current consumption of the whole system did not rise significantly. This is mainly due to the high power consumption of fast drivers of electrical busses contrasting to the quite low power requirements of optical transceivers. Additionally, the power consumption of optical transceivers is constant while the current drawn by a electrical transceiver strongly depends on the amount of data transported. A comparison of the current consumption on the different voltage levels is given in table 6.1. It has to be noted that the measurements from 2003 were done with pulser signals similar to real signals [62] while the 2011 data were taken during a high intensity Au + Au experiment so that some discrepancies have to be expected.

Nevertheless, the total power dissipated on the front-end modules increased. In the old setup, all voltage regulators were placed in separated from the front-end modules and the already adjusted voltages were delivered to the motherboards. To achieve a better voltage stability, some regulators have been put on the new read-out controller boards. The difference between input and output voltage of these regulators causes a power dissipation which is in the order of 20% of the total power consumption. The available cooling system for the MDC read-out system is able to cope with the additional heat produced and keeps the temperatures of all front-ends within acceptable limits.

The read-out chain is completed by the MDC Optical Hub that receives data from two complete MDC chambers (32 OEPs), packs the data into the HADES SubEvent structure and forwards it to the eventbuilder server farm via Gigabit Ethernet as described in section 5.5. The full setup with all components is shown in figure 6.1.

Furthermore, the low voltage power supply system for the FEE was rebuild. Central power supplies generate all voltages required by the front-end systems²². These are distributed to

²²The amplifier and TDC need voltages of 1V, 3V, 5V and -3V; the OEP needs 3.3V additionally.

Voltage	September 2003		August 2011	
	Current [A]	Power [W]	Current [A]	Power [W]
5V	227	1135	185	925
3.3V	–	–	70	231
1V	65	65	158	158
+3V	159	477	158	474
-3V	157	471	153	459
Total		2148		2217

Table 6.1.: The low-voltage power consumption of MDC. Values from 2003 are taken from [62] and were extrapolated to the full MDC setup. The power is calculated using the nominal voltage levels. 2003 data were taken with artificial signals generated by a pulser connected to the chambers while 2011 data were measured during heavy ion beam activity.

all 24 MDC chambers where all voltages are filtered and adjusted by voltage regulators. The OEP houses another set of voltage regulators to keep voltage fluctuations to a minimum.

Further details of the new electronics can be found in [21].

6.2. MDC Read-out Controller

The full read-out controller logic for the MDC TDC is located in the OEPs. Here, all function blocks needed to control the FEE and adapt to the common HADES trigger and read-out network are implemented. In detail, these blocks are: Trigger Handler, TDC Read-out, Data Handler, TDC configuration and the statistics and monitoring. A graphical representation of the blocks is given in figure 6.3.

Trigger Handler

This logic controls the full data taking sequence. For normal triggers, the incoming reference time signal is received by the OEP and forwarded to the TDC chips. If a calibration trigger is requested, the correct configuration is loaded in the TDC. Afterwards, the read-out of data from TDC is requested. Additionally, the configuration sequence is supervised and repeated everytime the motherboard produces an error during read-out.

TDC Read-out and Data Handler

During a trigger or calibration event, a read-out of the data is performed by sending a read-out token to the TDC. This token is handed through the chain of TDCs. The TDC that currently

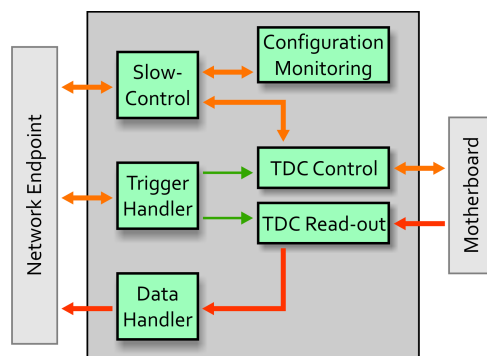


Figure 6.3.: The structure of the logic implemented in the FPGA on the OEP boards can be sub-divided into six main blocks. A detailed description of each block is given in the text.

has the token is allowed to actively sent its acquired data using the common data bus on the motherboard. The CPLD chip reads the data and divides them into smaller words which are sent to the OEP separately²³ and are received by the read-out handler in the OEP.

The data handler analyzes these data words, data cuts are applied (see section 6.4) and the remaining data is packed into the MDC data structure (see section 6.3). If configured, the data handler adds debugging information to the data stream. If a status information trigger is processed, data from status registers is read and sent instead.

TDC Configuration

The configuration component of the read-out controller communicates with the devices located on the motherboard. These devices are a set of 8 or 12 TDC, a CPLD and several digital-to-analog converters.

During the start-up of the DAQ system, they have to be provided with the correct configuration. Here, the data bus on the motherboard used for read-out of the TDC is used in a different mode: The OEP becomes master of the bus and sends addresses and data to the motherboard. Data are received by the corresponding device and stored in the selected memory location. Additionally to the data bus, the TDC are controlled by a set of signals selecting the operation mode. The complete start-up sequence is formed by a defined sequence of changes on these mode signals and writing data to registers.

The settings loaded to the TDC include the configuration of time resolution, spike suppression and operation mode, e.g. the number of signal edges recorded with each measurement. In addition, digital-to-analog converters are adjusted to produce threshold voltages for the discrimination stage.

²³This functionality was needed in the old DAQ system since it was not possible to transport data with the full bus width of 20 bit to the read-out controller.

All necessary settings are loaded from an internal memory. Here, default values have been stored that bring the TDCs to a working mode so that they can perform all necessary options. The specific configuration for each front-end board can be updated at any time using regular slow control accesses. The new settings are loaded with the next reconfiguration of the motherboard which can also be requested by slow control. The typical configuration sequence needs about 4 ms.

When a calibration trigger is being performed, the configuration logic reconfigures the TDC to a special working mode. In this mode, a set of three internal calibration pulses is generated by the TDC. All six edges of the signal are measured by the TDC and read out afterwards.

It may happen that a TDC fails to work. For example, the read-out request is not answered or the proper configuration was lost due to an error in the electronics. This case is detected by the read-out logic, i.e. a time-out is applied while waiting for data from the TDC. If the time limit is reached, a reconfiguration of the motherboard is performed. Only after a successful configuration the busy release packet is sent to the CTS so that no data is lost during the process.

6.2.1. Statistics and Monitoring

In parallel to the configuration and read-out, a number of on-line statistics is generated. The amount of all kinds of errors is being recorded. Among these, the number of failed read-outs from the motherboard and the total amount of data words is recorded. The number of invalid data words which have been rejected data words due to the limits set for TDC measurements is counted.

The time spent by the front-end during different steps of the read-out process are measured to gain detailed information of the dead-time produced by each board as well as its causes. Moreover, different types of errors during operation such as errors on the reference time input or transmission requests are counted. These statistics are available from slow control registers. They are also included in the read-out for each status trigger issued by the CTS. The full list of status information returned is given in table C.11. The status of all state machines dealing with front-end operation is also available from register (see table C.12).

During configuration of the motherboard, the settings stored in the internal registers of the TDC is read back and stored in a memory which can be read out by slow control and processed by a software tool. Several more registers that reflect the current status of each internal logic component are available as well.

The OEP is equipped with an ADC that monitors all supply voltages. These are automatically measured and compared to pre-set minimum and maximum values. The range of fluctuations on the voltage and the current voltage are made available to be read by slow control. The corresponding control and status registers are described in table C.9.

6.3. MDC OEP Data Format

With 372 front-end modules the MDC system forms a major part of the HADES DAQ system. The big amount of individual channels (more than 27,000) and the high number of channels affected by a single particle (up to 30) generates a big amount of data. In a central Au+Au collision the production of about 200 charged particles is expected. In total, this sums up to more than 5,000 active channels per event.

The data delivered by the TDC for each channel consists of two words (“hits”). One hit contains the time of the rising edge of the signal (Hit 1), the other gives the time of the falling edge (Hit 0)²⁴. Each data word is made up by the 11 bit time stamp, a 4 bit TDC number and a 3 bit TDC channel number plus a marker for the hit number. Thus, each hit generates 19 bit of information.

Nonetheless, the two hits generated for a normal input signal contain redundant data since they come from the same TDC channel. The amount of information for one channel sums up to 29 bit. This size can be fit into one 32 bit data word sent on the data channel leaving space for three bit of additional information to further mark the data format.

The other type of data generated by a calibration trigger returns six hits for each TDC channel which can be separated into three individual data words, each containing times from two hits.

The final MDC data format contains three different types of words: A debug format, containing a single hit and transporting all information delivered from the TDC. A packet format with two hits per data word that already is filtered for invalid data from TDC. The third type are status words that contain statistical information or additional words for debugging purposes. The definition for all three types of data words is shown in table 6.2.

The standard data format is the compressed format that results in the lowest possible overhead. The disadvantage of this format is that seemingly corrupted data from TDC is already removed since transporting single hits is not possible in this format. For debugging reasons the extended debug format is available transporting each hit sent by the TDCs separately in a data word.

The status words are used for two purposes: During a normal read-out additional data like the state of the trigger counter can be included. When a status trigger (type 0xE) is performed, only status words are returned. They include all kinds of information and statistical values generated in the read-out logic that can be needed for analysing the data. Each status word contains a payload of up to 24 Bit and a 5 Bit word identifier. The identifier guarantees that data can be interpreted without being dependent on a fixed structure or sequence of status words. All status words are listed in table C.11.

During a status trigger a total of 21 words (i.e. codes 0x00 through 0x14) are sent. If the

²⁴The rising edge gives the precise timing information while the length of the signal encodes the amount of charge deposited.

Bit	Debug Format	Compressed Format	Status Word
31 – 29	000	100	010
28 – 25	TDC number	TDC number	word code
24	TDC channel	TDC channel	code
23 – 22	TDC channel	TDC channel	data
21 – 11	Bit 21: Hit number, rest 0	TDC data - Hit 1	data
10 – 0	TDC data	TDC data - Hit 0	data

Table 6.2.: The MDC data format. Three different types of data words are defined. One contains raw data from TDC, one contains matching hits from the same channel and a third type is defined for status and debug information.

user selected to send dummy data for testing purposes, these data are also marked as status word (i.e. word type 0x1E).

6.4. Data Filtering and Reduction

Despite repacking data with two hits in one word, data received from the TDC can be further filtered to reduce the overall amount of data. On one hand, all relevant data from an event is found within a defined time window. On the other hand, a fraction of data words is corrupted and can not be related to actual signals in the detector.

In an ideal case, the TDC only sends data that contains both Hit 0 and Hit 1 from the same channel of a TDC. The actual data contains a fraction of words with time equals 0 and three hits from the same TDC channel. This is caused by an internal error of the TDC and thus can be filtered from the data stream. In some cases, one of the two hits is not sent. Since the remaining one time value is not sufficient for full analysis, it can be selected to be discarded as well.

The data also contain a number of hits that are not caused by a particle in the detector but by electromagnetic noise. If the logic is able to distinguish this type of hits, they can be filtered. All signals correlated with real particles are located within a time window of 200 ns for the inner chambers and 500 ns for the outer chambers and all show a specific range of pulse widths (i.e. the difference between both hits in one channel). Thus, all hits that lie outside of this window or have a very short or very long pulse width can be filtered. This is especially helpful since the internal spike suppression of the TDC does not reject all short pulses (below 13 ns as selected by the default configuration).

All data filtering can be configured during run-time by a set of registers in the OEP. These registers are described in table C.8.

Lossy Data Reduction

A further compression of data is possible if a loss of information is acceptable. This is possible, since data is not randomly distributed but shows very specific values for real signals.

Each data word contains the measured times and a TDC channel address. This address has to be further supplied with the OEP network address to determine the exact location of the active channel. This information is transported in the SSE-Hdr which are automatically generated in the network endpoint. This 32 bit header word is generated for each of the 372 front-ends in the system, resulting in 1.4 kByte of data per event. This is a significant fraction of the total data volume, especially for low-multiplicity events.

The information needed to attribute a data word to a specific channel is only 4 bit since only the motherboard number on a given chamber is needed. The information about sector and plane is already contained in the SubEventHeader generated by network hubs. If this information can be packed inside a data word, the additional header could be skipped. With the current data format, only 2 bit of each data word remain unused.

From data taken during past experiments, it is clear that all times measured for real particle signals fall within a window of about half the TDC range. Additionally, the time-over-threshold of the signals is always below 500 TDC units (i.e. on quarter of the full range)²⁵. Thus, the data transported can be changed to contain a 10 bit timestamp for the time of the rising edge of the signal (thus saving one bit of information). Additionally, instead of the time of the falling edge the time-over-threshold can be computed.

²⁵The actual pulse width can be longer in rare cases but it can safely be assumed that limiting the range does not affect the efficiency of data analysis.

7. Slow Control and Monitoring

Besides trigger and read-out, the third main component of the data acquisition system is formed by the slow-control. Here, all features that do not directly belong to the trigger and read-out system are integrated. The interface provided within all network nodes is described in section 7.1. All functions are provided based on an address space that spans the full read-out system and thus gives the possibility to directly access each individual information.

The configuration for each individual front-end is loaded and checked during the start-up phase of operation (see section 7.2.1). During operation of the data acquisition system, the slow-control interface is used to monitor the current status of the whole system. Online statistics about data rate and dead-times are generated along with information about possible errors within the front-ends. These features are shown in section 7.3.

An overview of the software available for basic monitoring and control purposes is given in appendix D.

7.1. The Slow Control Interface

Besides basic management functions like assignment of network addresses, all slow control features are implemented based on addressable registers. The interface between the network end-point and the user configurable logic is provided by the RegIO (Register-Input-Output) component. It provides a set of built-in registers and functionality as well as a multi-purpose data and address bus where any other functions can be connected. An overview of the address ranges is given in table 7.1.

The internal functions include a set of status and control registers common to all endpoints and a range of freely usable registers. Basic information about the endpoint, hardware and firmware version are stored in a read-only memory.

The internal data bus is implemented as a typical data/address-interface using a read/write-strobe signal and several feedback strobes. To guarantee a proper function of the slow control channel even in case of a failure of the user logic, the response time on the data bus is limited to 16 clock cycles after which a response must have arrived or an error will be reported. All possible responses to a read or write request are shown in timing diagrams in appendix A.4.

The address range available on the data bus is further divided into the network part (addresses up to 0x7FFF) and the user part (0x8000 and above). The network part is dedicated for all registers provided by the endpoint or hub logic such as information from the read-out buffers. The upper address range is freely configurable.

Addresses	Purpose
0x00 – 0x1F	Common status registers. These registers are identical on all front-ends and are used to collect information about the current operational status of the boards.
0x20 – 0x3F	Common control registers. These registers are identical on all front-ends and provide basic control signals for front-end configuration.
0x40 – 0x4F	Board Information ROM. This memory stores basic information about the hardware and firmware version of the board
0x50 – 0x5F	Timers are used to provide a time stamp and clock ticks for seldom updated logic parts.
0x80 – 0xBF	User defined status registers. Up to 64 status registers can be defined and used to provide any kind of information.
0xC0 – 0xFF	User defined control registers. Up to 64 control registers are used for additional configuration values.
0x0100 – 0xFFFF	These addresses are forwarded to the internal data/address port and can be equipped with any kind of logic.

Table 7.1.: The slow control interface provides a set of different functions based on addressable registers. A full list of common registers is available in appendix C.1

Independent from the data bus, the data from the on-board 1-wire temperature sensor is retrieved and used to allow the slow-control system to assign addresses to all network nodes.

The register interface does not only provide single register accesses but also block-read/write operations. Here, one TrbNet slow control request triggers several accesses on the internal data bus. The request selects between two modes: The first one executes repeated accesses to the same internal register address while the second one uses an auto-increment on the address. Hence, the first mode is suitable to read data from a fifo memory and the second mode can be used to read or write data from an addressable memory block.

Lastly, the reply data for a read access contains an internal time stamp with 16 us resolution²⁶. This timestamp is used to precisely determine the time delay between two read accesses and enables monitoring software to exactly measure change rates of values, e.g. amounts of data sent per second.

Error Information

Similar to the other network channels, the slow control channel features a set of error information bits that are sent with the termination word of a reply. If a request to a non-existent

²⁶This resolution was selected as a compromise between word size (which is limited to 16 bit due to the TrbNet packet size) and rate of overflows of the counter (about 1 Hz).

address is received or the given address can not handle the request type of access, a “unknown address” flag (Bit 16) is set.

If the address corresponds to a fifo or register which is currently empty and does not contain data to read or a write access to a register is currently not possible (e.g. due to a busy handler for this address), the “no more data / retry” flag is set. If the user logic fails to answer a request within the given time-limit, a “timeout” flag is set (Bit 17).

7.2. Slow-Control Features

7.2.1. Start-up and Configuration

The full start-up of the DAQ system is controlled by the startup program. All configuration is provided in the form of script-files or text based database files which are read by the script and executed. These script do not only include TrbNet accesses but also all other operations that have to be carried out on various computer systems.

The custom scripting language contains conditional executions that can be controlled via command line options to select from all available configuration options. E.g. all sub-systems can be activated individually or various thresholds can be selected. The script also automatically checks if all TRB boards and other CPUs are available and accessible. The current DAQ network configuration is determined and a list of all connected front-end boards is written to the main beam-time data base.

Settings that are loaded to registers are provided in a database format as shown in figure D.1. Further information about the start-up scripts and configuration files formats can be found in [63].

The configuration options available are dependent on the front-end type. The configuration and status registers available in the MDC system are listed in section C.3. These include the full setup for the TDC ASICs and reference voltage DAC contained on each MDC motherboard. The settings are stored in an internal memory block and are loaded to the motherboard when a re-configuration is requested; i.e. either during the start-up process or by an explicit request via slow-control. The data reduction algorithm can be configured and certain automatic error recovery functions can be toggled on or off. Network hubs are also widely configurable. A brief description of all features included can be found in section 3.4.2.

7.2.2. Monitoring and Online Statistics

All network nodes and logical function blocks contain a set of monitoring and statistics features. These show the current status of the most important functions and state machines, the state of external signals and many more things. Here, only few examples can be given:

All front-ends provide a memory block in which threshold and pedestals for each individual channel can be stored and automatically applied during data taking. The MDC front-ends

additionally contain a huge set of monitoring and statistics features. Here, times spent for different operations, number of data words and error information can be retrieved. A more complete description is given in section 6.2.1.

The network hubs contain registers for the status of each network link, the amount of data transferred and the received error messages. A more detailed description is given in 3.4.2. The central trigger system contains a huge set of monitoring features that can be found in [54]. It contains records of the count rates on start and veto detectors with configurable time resolution.

7.2.3. Common Slow-Control Features

All front-ends contain a set of pre-defined registers with basic information about the current status of the front-end. Moreover, common control registers are defined for basic accesses, like gating the reference time input.

A set of nine status registers give an overview of the operational status of the front-end. The first register contains error flags and gives direct access to all error conditions. Five registers are assigned to the status of the trigger handler and contain counters for all kinds of erroneous reference time signals. The LVL1 and read-out handler share two further registers which allow to monitor the synchronous operation of all front-ends. Lastly, a register reports the status of the error correction algorithm used on some optical links. Depending on the type of network node, not all registers are in use. I.e. network hubs do not contain information from the trigger handler. The full list of common status registers is given in appendix C.1.

Two common control registers are defined for basic control of the front-ends. One register contains strobe signals which are used to trigger actions in the FPGA such as clearing status registers or reinitializing front-end modules. The second register contains persistent configuration flags, e.g. to select a event data format or to enable sending of debug information. The common control registers are described in detail in appendix C.1.

Three registers are connected to an internal read-only memory that contains information about the compile time and version of the FPGA design as well as a hardware identifier. These information is set during the compilation and allows the slow-control software to identify boards. This method is used for example within the Flash programming routine to prevent loading of a design not matching the actual hardware. A further register provides a microsecond clock that can be used to tag data or status logs.

7.3. System Monitoring

A crucial point is the visualization of all available monitoring data in a way that allows for a fast detection of possible errors in the data acquisition system. At the same time, detailed status information about individual aspects of the system has to be available to a large group of people.

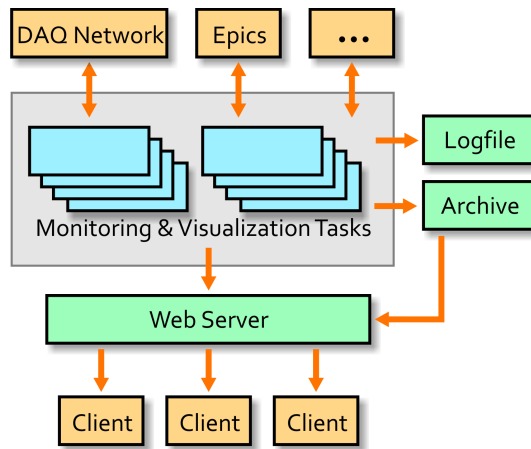


Figure 7.1.: The Monitoring Tool Hmon consists of a set of individual tasks that collect, analyze and visualize data from many sources. The information can be accessed via a web server and is stored in a logfile and an archive for later analysis.

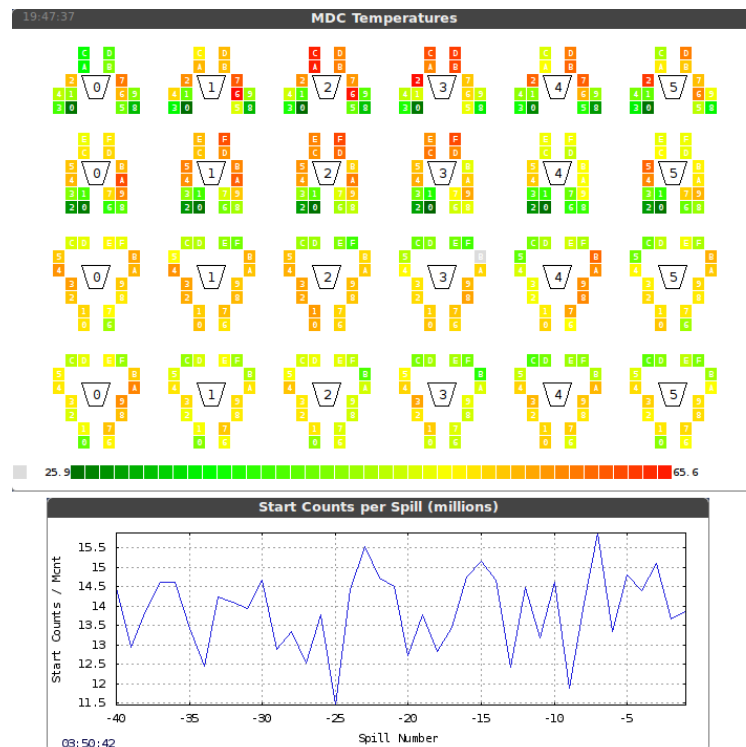


Figure 7.2.: Examples of the information provided by Hmon tools. Top: temperatures of all 372 MDC front-ends. Bottom: beam intensity history.



Figure 7.3.: The tactical overview window provides a condensed overview of the status of various aspects of the system. Errors are shown by changing colors.

A simultaneous access of all users to the DAQ system is not desirable due to the high load generated on front-ends and servers. Hence, a set of independently running monitoring scripts collect and process information continuously. This data is provided to a web-server from which all users can retrieve the data. In this setup, the number of users has only a marginal influence on the load on all systems on the data acquisition system. In parallel to providing online data to all clients, part of the information are stored on disk for later analysis of all errors within the system. A schematic view of the monitoring system is given in figure 7.1.

The tools collect data from various sources, such as TrbNet, EPICS [64] or direct access to other computers over ssh. This data is analyzed and visualized in different ways: the HadPlot tool (see chapter D) provides diagrams while a lot of information can be displayed by pure HTML code [65]. Two examples of information windows are shown in figure 7.2. All data collectors generate a short status information which is shown in an overview window (“Tactical Overview”, see figure 7.3). Here, the current status is shown with a short and a detailed message as well as a color-coded state information. The operator observing the DAQ system can also access further information and possible error handling routines from this window.

8. DAQ Performance

As described in the previous chapters, the complete data acquisition network has been implemented and was successfully used during several commissioning runs. During these experiments, the performance of the new DAQ system was determined which is shown in the following section. Moreover, a comparison to calculations based on simulation, tests and measurements done with laboratory setups is discussed. Here, the most important numbers are the latency of information transport (see section 8.1), the dead-time of the detectors and the read-out time for each event (see section 8.2). Finally, a summary of the latest results of the August 2011 test experiment is given in section 8.4.

The HADES data acquisition system and electronics is also used in various external test-setups and experiments which are briefly shown in section 9.1. Moreover, additional possible improvements of the DAQ system are evaluated and ongoing developments in electronics are described in section 9.2.

8.1. Latency Measurements

One of the most critical parameters for the HADES DAQ network is the latency for data transmission. This time directly influences the dead-time of the whole system since the arrival of the busy-release signal of the slowest sub-system defines the time when the detector is able to record the next event. Table 8.1 shows a set of all relevant latency measurements for parts of the network.

The tables lists the fixed latencies introduced by the serdes²⁷ IP-core within the FPGA, separate for four different configurations (with or without clock tolerance compensation (see below) and 2 GBit/s or 250 MBit/s speed) used in the network. Since this latency directly scales with the clock speed of the serial data stream, the 250 MBit/s link have a latency that is roughly eight times higher than on 2 GBit/s links²⁸.

Additional time is spent for data preparation and data reception within the media interfaces. Here, the 2 GBit/s link is dominated by the time required to transfer data between different clock domains since the serdes block uses a clock source distinct from the internal FPGA

²⁷Serializer/Deserializer. The serialization of data is necessary to transport them over a single wire or an optical fiber. A serdes converts a serial data stream into a data word and vice versa. In FPGAs, the serdes is a dedicated block within the device that provides the necessary functions, including encoding of data and recovery of the clock signal from the data stream.

²⁸Slight deviations are due to different configuration of the width of data interfaces

Measurement	TX [ns]	RX [ns]
Serdes 2 GBit/s CTC (Hub)	60	145
Serdes 2 GBit/s (FEE, Hub Uplink)	60	70
Serdes 250 MBit/s CTC (MDC-Hub)	360	1,040
Serdes 250 MBit/s (OEP)	360	440
Media Interface 2 GBit/s	60	70
Media Interface 250 MBit/s	260	660
Media Interface parallel	20	60
10 m optical cable	55	
Hub Logic (LVL1, SCtrl)	180	
Endpoint Logic (FEE)	120	
CTS reference time to LVL1 generation	550	

Table 8.1.: Calculated Latency Times for various logical blocks within the DAQ network. Serdes latencies are mean values as given in [66], other blocks are derived from code, simulation or measurements.

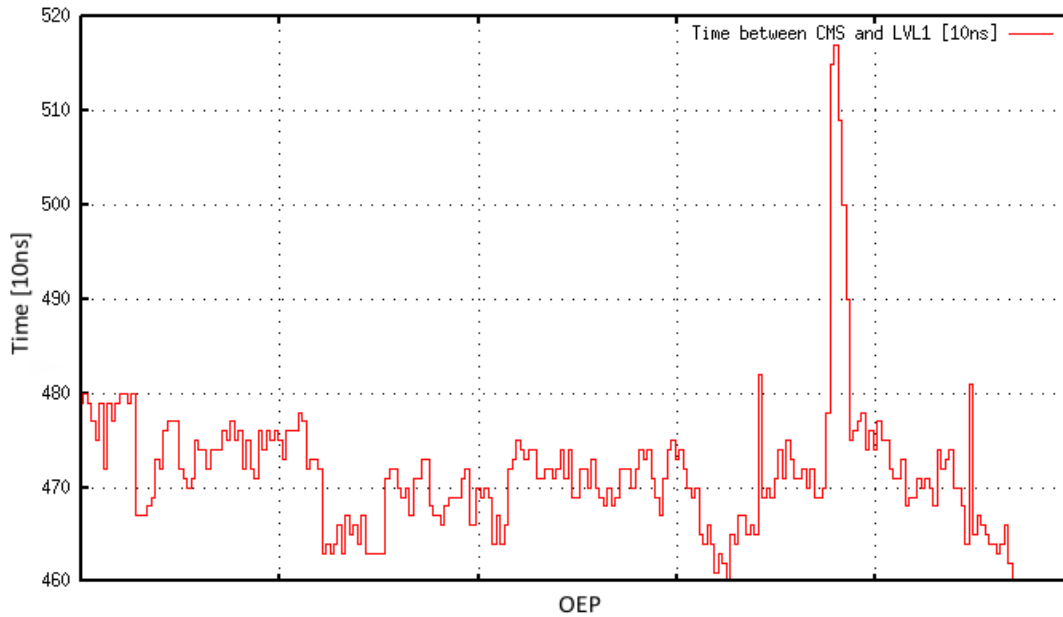


Figure 8.1.: The time difference between the reference time signal and the LVL1 trigger is a measure for the total latency of data transmission in the network. The plot shows the time in units of clock cycles (10 ns) for each of the 372 OEP within the system for a given event. The single peak is caused by a delayed LVL1 trigger packet due to other network traffic.

Component	CTS -> FEE [ns]	FEE -> CTS [ns]
Hub version 2 network hub	710	785
MDC Hub	1,380	2,055
FEE Endpoint	290	270
MDC OEP – CTS	4,270	4,610
Shower FEE – CTS	2,470	2,580
RICH / TOF / RPC FEE – CTS	2,060	2,210

Table 8.2.: The latencies of several important components and data paths of the DAQ system. The values are summed based on table 8.1.

logic. The 250 MBit/s interface latency is to be attributed to the error detection and correction logic. In particular, the transmitter has to pass data through an additional buffer stage while the receiver has to collect and analyze a full network packet before it can be forwarded to the network endpoint.

Last, the contribution of a network hub stage as well as a network endpoint are both well below 200 ns. Table 8.2 sums up the different contributions for the main components in the network: The latency between the input of a network hub and its output is 710 ns and 785 ns, respectively, for up- and down-link. In the front-ends, about 300 ns is needed to receive data and offer it to the application and vice versa. The full network path between CTS and MDC front-ends amounts to 4.3 μ s to 4.6 μ s, while other the latency for other sub-systems are lower by roughly 2 μ s.

The latency for the busy-release packet to be transported from the FEE to the CTS is longer than the latency in the opposite direction. This is due to a different configuration of the serdes module in case several inputs are used on one serdes bank²⁹. Here, the clock tolerance compensation (CTC³⁰) feature of the serdes is activated to reduce the necessary amount of resources in the FPGA. This feature introduces a latency of 15 clock cycles in average (75 ns on 2 GBit/s links, 600 ns on 250 MBit/s links). In total, this sums up to an additional latency of 750 ns for MDC front-ends and 150 ns for the other systems.

The actual latency values can easily be obtained in the detector setup, since all endpoints measure the time interval between receiving a reference time signal and the corresponding

²⁹The internal configuration of the FPGA groups four serdes modules in one serdes quad that shares specific features. Furthermore, the FPGA is not able to handle a setup where 12 serial links are all receiving data at distinct clock speeds. Hence, a special operation mode (Clock Tolerance Compensation - CTC) had to be chosen where the clock for received data is adjusted to the internal clock of the FPGA.

³⁰Clock Tolerance Compensation. Data transmitted by one board usually has a slightly different clock speed than the one used on the receiver side. Hence, care has to be taken to transfer data between the two clock domains without loss. The serdes modules contain a special block, CTC, to adapt the incoming data stream to the internal clock by adding or removing “idle” words

Subsystem	calc. Latency	meas. Latency
MDC	4,820	4,820 – 5,020
pre-Shower	3,020	3,050 – 3,150
RICH / TOF / RPC	2,610	t.b.d.

Table 8.3.: The actual latencies have been measured based on the time difference between the reference time signal and reception of the LVL1 trigger. The time measured includes the time needed by the CTS to generate the LVL1 trigger of 550 ns.

LVL1 trigger. Figure 8.1 shows this value for the MDC front-ends in units of 100 MHz clock cycles. The measured time of 4.6 μ s to 4.8 μ s for all boards matches the theoretical expectation. A small subset of boards report a latency of up to 5.2 μ s. This is caused by other traffic on the optical network that can delay the transportation by up to one packet length. In case of the 250 MBit/s links, this time is 480 ns.

The different cable length depending on the position in the detector is not visible in this plot since the cable lengths for reference time and optical signals are almost identical and the speed of light in both media is comparable. Hence, their contribution to the measured latency cancels out.

The measured latencies include the time the CTS takes to generate the LVL1 trigger after a trigger decision. In the current implementation, this time is 550 ns between sending the differential reference time signal and starting the LVL1 trigger transmission. Table 8.3 shows the measured latencies for the MDC³¹ and pre-Shower system which agree with the calculated values within a small error. The latency of other subsystems could not be determined because in the FPGA firmware the required measurement logic was not implemented.

8.2. Read-out Time and Dead-Time Estimations

The event rate is basically limited by the read-out time. In some sub-systems, the time can vary greatly depending on the number of data words produced. Table 8.4 gives an overview of the minimal and maximal times needed to transfer data from the front-ends. These times can be compared to the theoretical values based on the expected occupancies as given in table 2.3.

The pre-Shower and RICH sub-systems have a fixed time which does not depend on the number of data words per event. Here, the time is defined in the multiplexing stage of front-end channels to ADC input channels. E.g., the pre-Shower front-end has an intrinsic read-out time of 300 ns per pixel row and a total of 32 rows resulting in a total time of 9.6 μ s.

In the TDC based sub-systems, data has to be transferred from TDC to the FPGA. The

³¹The MDC latencies were corrected by an additional offset of 220 ns due to a fixed delay of the reference signal

System	Min. time [μs]	Max. time [μs]	Typ. Au+Au [μs]
RICH	5.2 ¹	6.1 ¹	5.7 ¹
MDC	0.6	38.4	7.6
Shower	9.6	9.6	9.6
TOF TRB	0.5	3.6	1.0
RPC TRB	0.5	3.6	0.7
other TRB	0.5	3.6	0.7

Table 8.4.: Transmission time for the read-out of the front-end electronics. The second and third columns show times for minimal and maximal data words. Times for a typical Au+Au event are given in the last column.

⁽¹⁾ It should be noted that the actual dead-time is lower due to analog buffers within the front-ends.

transmission needs a fixed time for each data word to be sent. The MDC system has a very long read-out time of almost 40 μs in the case that all TDC channels deliver data. This time is reduced to 8 μs for an typical occupancy of about 20%.

The typical read-out time for the full system is given by the slowest sub-system which is not identical to the system with the longest typical read-out time. The amount of data per front-end varies both by event-by-event fluctuations and the position in the detector. This leads to a variation in the read-out time. Especially the MDC front-ends where a single board only reads a very constraint part of the detector show a high fluctuation factor of approximately 4. Hence, the total read-out duration of the MDC system is given by the board with the highest occupancy that has an approximate read-out time of 15 μs .

The lower limit of the dead-time is given by the read-out time of the slowest front-end system plus the latency of the busy-release packet to the CTS. The latency of the LVL1 trigger can be neglected since read-out already starts with the reception of the reference time signal.

The reference time signal arrives about 400 ns after the physical event and the busy-release latency is 2.6 μs as shown in table 8.2. The total dead-time sums up to 12.6 μs , corresponding to a trigger rate of 79 kHz. The typical dead-time is given by the MDC read-out time (15 μs) plus the latency for the busy-release signal (4.6 μs) and the reference time generation delay of 420 ns. The resulting time is 20 μs and the accepted event rate is 50 kHz.

8.3. Read-out Bandwidth

The band-width available for event data transport of the full system is limited by the number of Gigabit Ethernet links available. The bandwidth of the TrbNet links is not relevant since

System	GbE links	Data rate [Mwords/s]		factor
		available	expected	
RICH	3	37.5	9	4
MDC	12	150	90	1.5
Shower	6	75	10	7
TOF TRB	1	12.5	2.5	5
RPC TRB	2	25	11	2
others	2	25	4	6

Table 8.5.: The bandwidth for event data transport is mainly limited by the number of Gigabit Ethernet links connected to each sub-system. The available bandwidth is well above the expected data rates for all sub-systems.

it is, in general, higher: 1.2 GBit/s net data rate compared to 400 MBit/s per Ethernet link. The same holds for the 160 MBit/s optical links within the MDC system, since 32 front-ends share one Gigabit link.

Table 8.5 gives an overview about the available data bandwidth and a comparison to the expected data rates as shown before. The available bandwidth is well above the expected rates for all sub-systems. The MDC sub-system has the lowest free bandwidth of about 33% while all other systems have a bandwidth reserve of 70%. This guarantees that data transport is never a limiting factor for the complete system.

8.4. August 2011 Test Run

In August 2011 an experiment to test the capabilities of the tracking system in heavy ion reactions was conducted. In parallel, all parts of the data acquisition were tested as well. The accelerator delivered a Au beam at 1.25 AGeV and a typical intensity of 1.4×10^6 particles per second. The target was a 15-fold segmented gold foil target with a mean interaction probability of 1%.

MDC Optical Link Error Correction

The efficiency of the retransmission system on optical links of the MDC system (see section 3.3.4) is clearly visible from figure 8.2. In average 40 retransmissions are necessary per minute under high-intensity (1.8×10^6 particles per second) beam conditions. All errors were successfully corrected. The majority of errors was detected by the receiver of the OEP as was expected from the increased noise environment in the vicinity of the detectors.

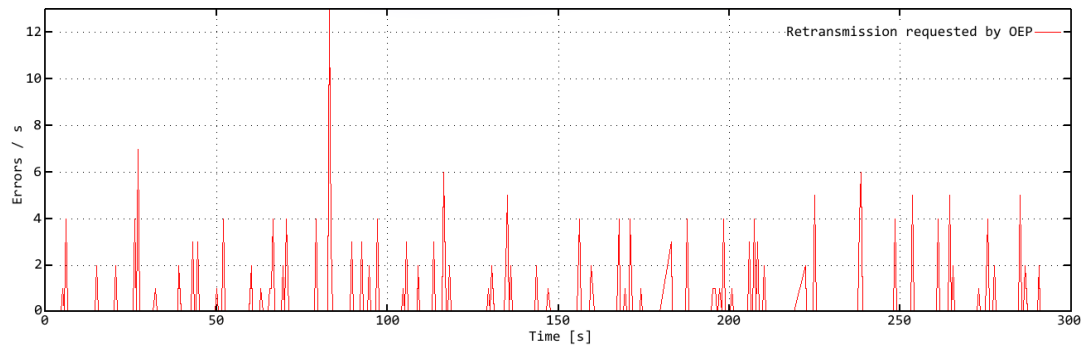


Figure 8.2.: Rate of transmission errors on the MDC optical links. The histogram shows the number of retransmission requested in both link direction during 5 minutes under high intensity beam conditions. Roughly 200 requests have been issued in the complete MDC DAQ system, most errors occurred on data received by the front-ends.

Amounts of Data Reduction in MDC

The MDC front-ends applies selections on the TDC measurements. E.g., all data with a measured time below 140 ns are discarded. TDC channels which miss one time measurement are also not stored. As shown in figure 8.3, the average reduction under beam conditions is about 10% but is much higher on some front-ends. It should be noted that these selections reduce the data but leave the over-all busy time of the detector unchanged. This is due to the fact that the time is mainly fixed by the read-out time of the TDC and the amount of data transferred between motherboard and OEP is not affected by the cuts.

The figure also shows one chamber of MDC plane 3 which was not supplied with high voltage and three front-ends that don't deliver any data due to a hardware failure.

Micro-Structure of the Beam

The start detector in combination with the monitoring features of the CTS provide an ideal tool to examine the changes in beam intensity with very high granularity. The micro-structure of the particle beam extracted from the accelerator shows large fluctuations on the scale of 10 μ s which are visible in figure 8.4. This measurement was done at a beam intensity of 1.8×10^6 particles per second, i.e. a mean rate of 10 particles per data point. The maximum rate summed over the whole start detector can be as high as 60 particles per 10 μ s but also shows intervals of more than 50 μ s without any particle. The effect of these inhomogeneities can be seen in the fraction of events actually recorded as shown below.

Detector Dead-Time

Figure 8.5 shows the dead-times of the different sub-systems. The total dead time is shown in blue (large bars); additionally the exclusive busy time, i.e. the time during which only one

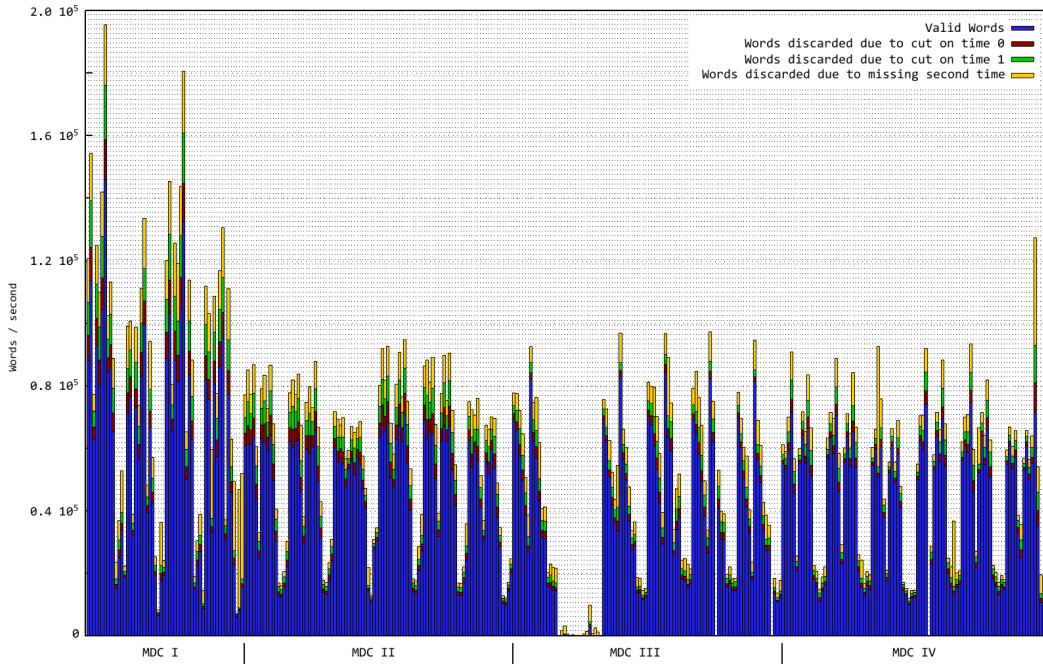


Figure 8.3.: The front-end logic of the MDC OEP apply cuts on the measured time (> 140 ns) and also discards data from TDC channel that do not deliver times for both edges of the signal. In average, 10 to 20 percent of data is filtered by these cuts.

given system was busy, is shown in red. As expected (see section 8.2), the MDC system, separated in an inner and an outer half, shows the longest dead times followed by the Shower detector. The dead-time of the full system is about 5% higher due to additional latency between the point of measurement (i.e. the central TrbNet hub) and the CTS as well as trigger generation delays.

The read-out time of different MDC front-ends shows the expected variations. The read-out time per event averaged over all motherboards is $7.4 \mu\text{s}$. Some boards, especially in plane I, show a much higher time of up to $20 \mu\text{s}$ due to a high occupancy. It has to be noted that these values are averaged over thousand events so that event-by-event fluctuations are not taken into account.

The trigger rate at the time the dead-times were measured was 10 kHz, combined from 8 kHz semi-central (multiplicity 20 or more in TOF) and 2 kHz peripheral (multiplicity 5 or more in TOF) events. Even though the dead times do not scale linearly with trigger rate, especially due to limitations in data transport rates, it can be stated safely that the DAQ system is able to handle the projected 20 kHz trigger rate containing 10 kHz central collisions as the data rates are far lower than any limitation as shown below.

The non-uniform distribution of beam intensity strongly affects the fraction of interactions that can be recorded. Hence, a good micro-structure of the beam is essential to record a

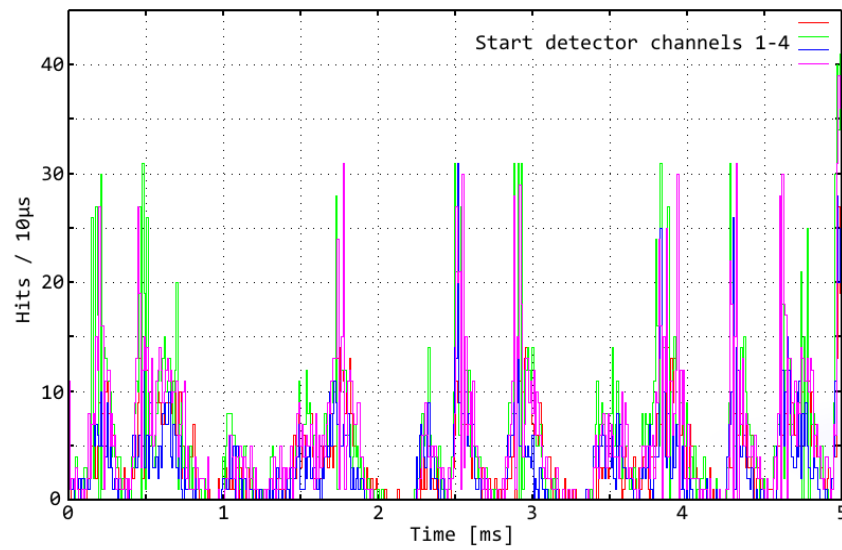


Figure 8.4.: The start detector in combination with the CTS provides a measurement of the fluctuations of beam intensity with high granularity. Here, the number of hits in the start detector is shown with a bin size of $10\mu\text{s}$ measured in the middle of a spill. The different colors represent the four channels of the inner part of the detector.

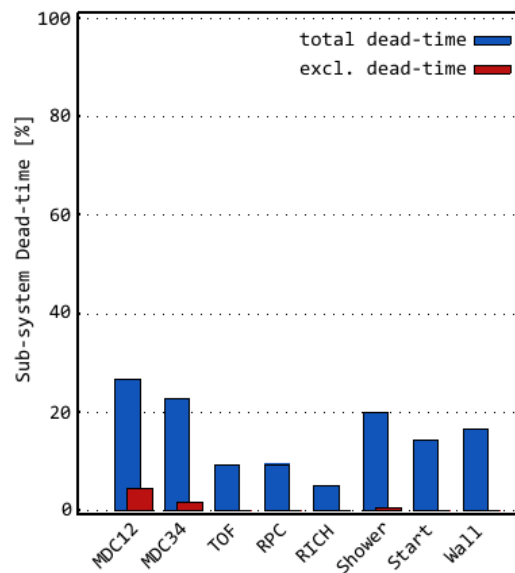


Figure 8.5.: The dead-times of all sub-system at a trigger rate of 10 kHz.

Beam Intensity (particles / s)	Trigger [1000/s]		Ratio
	offered	accepted	
0.5×10^6	2.5	1.75	0.70
0.9×10^6	5	3	0.60
2.0×10^6	13	7.5	0.49

Table 8.6.: The ratio of generated trigger signals over accepted events varies depending on the beam intensity.

decent number of events without increasing the load on detectors too much.

Even though the average dead time of the data acquisition system was less than 40%, the data acquisition system was able to record only less than 50% of triggered events provided at high beam intensity. Naturally, this ratio gets worse when operating at higher interaction rates as shown in table 8.6. A rough estimate shows that in order to reach the proposed event rate of 10 kHz for central events, the beam intensity has to be increased by a factor 3 for two reasons: First, the reduced ration of accepted triggers has to be accounted for. Second, the trigger setting used during the August 2011 test experiment corresponds to a lower multiplicity than proposed and hence to a bigger number of triggers generated.

The different channel occupancy of front-end boards depends on the position on the detector as described above. The strongest deviations were expected for MDC. The actual values are shown in figure 8.6. The boards connected to long sense wires and those located near the beam axis show a pronounced amount of data with rates about two times higher than the average. As is visible in figure 8.3, the fluctuations are stronger in MDC plane I and not evenly distributed. The reason are lower thresholds and spike rejection settings of the front-ends. Nevertheless, the fluctuations are covered by the estimations done in section 2.2.1.

Data Rates and Event Sizes

The data rates of all sub-systems were estimated by simulation in table 2.3. In table 8.7 this estimation is compared to the actual rates measured during the August 2011 experimental run. For most systems, the numbers agree quite well. The much lower data rate from the RICH detector can be attributed to the lower noise contribution of the new read-out electronics.

In total, the measured data rate was slightly lower than the expected data rate. On the other hand, one has to take the different event selection into account. In simulation, a high multiplicity of 100 or more particles was selected while the actual experiment used a trigger on 60 particles in the time-of-flight system and also added low-multiplicity events. From the measured rates it can be concluded, that the actual data rates are significantly higher than the ones expected from simulation. Anyhow, the impact on the data acquisition system is small since all data paths are fast enough as shown in table 8.5.

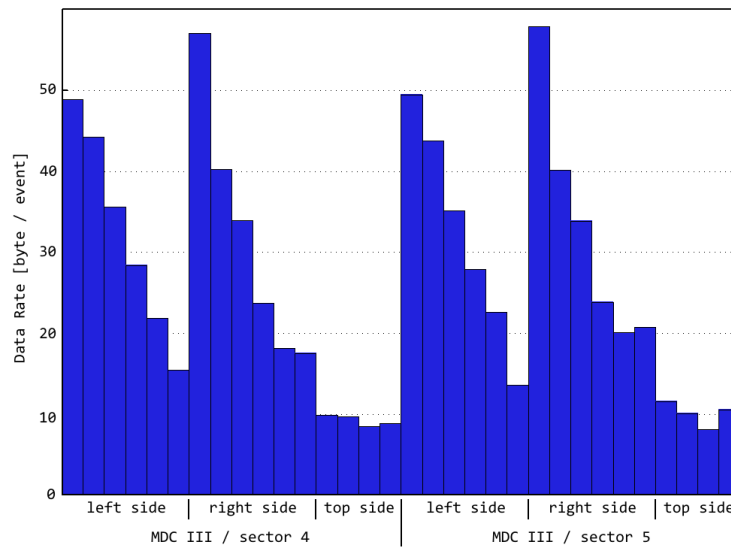


Figure 8.6.: The amount of data sent by each front-end of the MDC system varies depending on the position of the front-end in the detector. Here, two chambers of MDC plane 3 are shown.

Summary

The slow-control system was able to deliver all information needed for a detailed analysis of the system status. The event and data rates of 15 kHz and 330 MByte/s, respectively, are very good and the bandwidth margins allow for further increase. The event size was slightly bigger than estimated due to the contribution of noise, but still well within the limits the system can handle. The data reduction in the front-end systems was able to reduce data rates by up to 10%, zero-suppression not taken into account. The transmission errors seen during earlier runs were all corrected successfully. In conclusion, in the August 2011 run the data acquisition system showed very good performance values.

System	kB/evt (Simulation, 25% most central)	kB/evt (Aug. 2011, 80% mult. 60 + 20% mult. 15)
MDC	19.3	Sum: 18 MDC I: 7.2 MDC II: 4.8 MDC III: 3 MDC IV: 3
RICH	1.75	0.65
TOF	0.65	0.9
RPC	2.4	2.4
Shower	2.1	1.8
Others	0.8	0.85
Total	27 kB/evt	24.6 kB/evt

Table 8.7.: The estimated data rates for all sub-systems as given in table 2.3, including data headers, compared to the actual values measured during the August 2011 experimental run. The MDC data rate has been extrapolated to a full setup with 24 chambers.

9. Conclusions, Synergies and Possible Improvements

In the experimental runs of the HADES experiment since 2002 it became clear that the original DAQ setup is not sufficient to operate in an experiment with a heavy ion beam. Hence, the data acquisition system of the HADES spectrometer was upgraded during the last years. In this scope, almost all digital front-end electronics and the complete data transport chain were exchanged. Since 2010 several commissioning and test experiments of the whole system were conducted and showed a good overall system performance.

The data transport in the inner data acquisition system is handled by a custom network protocol, TrbNet. It was designed as a versatile protocol that can be adapted to all kind of different hardware platforms. Extension to new hardware components is easily possible. The inherent status reporting features allow an efficient reporting of possible errors in the systems. Besides a rough overview of the status of the detector, the user interface features also detailed information about all front-ends. The control interface allows the operator to access all front-ends and introduce reconfigurations on-the-fly.

The new electronics are not only capable of higher data and event rates but also have several other positive effects on data quality. The electromagnetic noise produced by the read-out electronics in the sensitive front-ends was reduced so that signal detecting thresholds could be reduced. Especially, now data acquisition and data transport in MDC are now run in parallel while the old system was operated in an interleaved mode due to generated noise.

In August 2011 a four-day experimental run took place to commission the full system, including analysis software, for heavy-ion experiments. In summary, all measured performance values agree with the expected and required numbers. The total data rate is higher than expected but well within the limits of the network capabilities due to the contribution of noise signals that can not be cut due to negative effects on data quality. Extrapolating all data to higher trigger rates, the DAQ system can be operated at trigger rates of at least 25 kHz even with a big percentage of high multiplicity events included.

9.1. Synergies: Employment in Other Experiments

The hardware and network protocol of the HADES DAQ system and hardware, especially the Trbv2, is also employed in several other experiments. E.g. at LIP³² in Coimbra, Portugal an Positron-Emission-Tomography (PET) scanner for small animals is being developed [67].

The TrbNet protocol will also be used for the read-out of the prototype for the CBM Micro-Vertex-Detector (MVD) [68]. Even though this detector uses a free running read-out system compared to the triggered system of HADES, TrbNet can be employed with small modifications. The detector consists of about 60 CMOS Pixel sensors (Monolithic Active Pixel Sensors / MAPS [69]) with one million pixels, an area of 4 cm² and a frame rate of 30 kHz each. These sensors will be read out by an array of FPGA-based read-out controllers that apply a first level of data analysis and reduction. The data is further transported to servers based on one of the two technologies used in the HADES setup (Gigabit Ethernet or PCI-Express). All front-ends are run synchronously controlled from a central arbiter which replaces the trigger system of HADES.

The arbiter sends requests for each frame recorded by the sensors. The read-out controllers check, process and store the data until a read-out request is received. The reply to each frame-request is used to transport information about the current status of the sensors and buffer fill levels to the arbiter which can start the appropriate actions in case of a problem.

The synchronous read-out of all sensors can be achieved by running all sensors with a centrally distributed clock signal. Here, either the clock recovered from the optical data stream or an independently distributed clock signal can be used. The additional complexity introduced is outweighed by numerous advantages. First, the recording time of each individual pixel is known precisely and can be used to improve all high-level trigger and tracking algorithms. Moreover, the central arbitrated operation allows for efficient error handling, e.g. if data from one sensor is corrupted, the whole frame can be skipped if desired.

Lastly, this operation concept for pixel sensors is closely related to a triggered read-out system. Thus, most components developed for the HADES DAQ system can be reused for the CBM-MVD read-out.

9.2. Further Developments

The current DAQ network implementation increased the total bandwidth and rate capability of the DAQ system substantially. Nonetheless, further improvements can be done. In particular, the event rate for light ion experiments such as proton-proton collisions is still limited by the DAQ system to roughly 50 kHz while the detectors can handle even higher rates in smaller collision systems.

The main source of additional dead-time is the latency of the busy-release information.

³²LIP - Laboratório de Instrumentação e Física Experimental de Partículas

This can be overcome by introducing a modification to the trigger-busy-release architecture as described in section 9.2.1. Higher data-rates and more error correction features can be achieved by modifications of the low-level network protocol. The outline of a proposed second version of TrbNet is given in section 9.2.2.

Last, the time measurement of the detector can be increased by exchanging the TDCs to ones with a better resolution. A new general-purpose read-out platform that is capable of time measurements with a resolution of better than 15 ps is currently under development. The TRB version 3 is briefly shown in section 9.2.3.

9.2.1. Reduced Dead Times: Credits-based Trigger

In the current trigger system implementation a lot of time is spent after read-out is finished to transport this information to the CTS. A possibility is to send the busy-release information already before the actual read-out cycle is finished. This is only feasible for systems with a fixed read-out time and can not be implemented in sub-systems with varying times such as MDC. Unfortunately, this is also the system that has the longest read-out times by default.

A solution is a credits-based trigger system. This means that not every single trigger has to be released, but a set of triggers is sent without a busy release signal. After a given number of triggers, the CTS has to wait for a busy release to prevent buffer overflows in the front-ends. This scheme can be implemented as long as the read-out time of the slowest subsystem can be predicted with a precision high enough. If the estimate is too high, dead-time is increased. If the estimate is too low, the next trigger is sent before all front-ends are able to take the next block of data. Hence, the data from these systems will be lost. In fact, this is acceptable if the number of events with missing data is sufficiently low and the error is reported within the data.

The proposed scheme allows the CTS to send a set of reference time signals corresponding to several events without sending a LVL1 trigger. To distinguish this special trigger sequence from errors on the reference time signal, a special trigger type is used and the number of sent triggers is announced by the CTS. Additionally, the reference time signal can be modified to allow the front-ends to detect the altered trigger sequence. Two different lengths of the timing signal, e.g. 100 ns and 150 ns, are sufficient.

The number of consecutive reference time signals has to be limited due to buffer size requirements in the front-ends. The front-ends have to provide the buffer space to store all events of a sequence because read-out can not take place before the final LVL1 trigger has been sent. Here, the buffer size has to be calculated based on average sized events. Maximum sized events as are used to determine buffer-full conditions in the current system are usually caused by calibration triggers. By definition, these triggers can not be sent in a sequence and hence, the front-ends are able to generate a busy after each of these.

Based on usual event and buffer sizes, a sequence of up to 10 events is reasonable. Assuming a typical dead-time of 16 μ s per event as calculated in 8.2, 10 events can be recorded

within 163 μs compared to 190 μs in the conservative approach. In light collision systems, a reduction from 130 μs to 100 μs can be achieved. Taking the planned further HADES upgrade with a calorimeter and removal of the Shower sub-system into account, the dead-time for small-sized events will not be limited by the fixed front-end timing any more. Hence, an estimated reduction from 90 μs to 53 μs dead-time is possible.

If the MDC front-end electronics is also replaced as noted in 9.2.3, the read-out time can be further reduced to approximately 30 μs corresponding to an event rate of 300 kHz.

The event rate gained by a credits-based trigger system lies between 15% and 40% depending on the experimental setup while keeping data loss due to wrong read-out time estimations at a minimum. With new MDC front-end electronics the total read-out time can be calculated precisely for all sub-systems and all data loss avoided and the total event rate can be increased by a factor of three. From the August 2011 experimental run with 55% accepted high-multiplicity triggers it can be concluded that the number of recorded events could be increased by 80% without changing the beam intensity.

9.2.2. Increased Bandwidth and Control Features: TrbNet v2

The current TrbNet implementation bases on a fixed packet size and no further mechanisms to guarantee that the packet boundary assumed by the receiver is always synchronized to the actual packet position. Moreover, it can not be easily detected if one word is corrupted or lost during transmission. As a result, a corrupted packet is likely to corrupt all data transfer between two network nodes.

The FOT data link already includes a complete error detection and correction mechanism based on a set of link control characters. Unfortunately, this scheme can not be directly adapted for all 2 GBit/s links due to the limited capabilities of the TLK 2501 transceiver used on the TRBv2 board. In fact, this chip, which was used in a major part of the DAQ system during the first construction phase, was also the reason for the current TrbNet implementation.

A new TDC platform is already under development (see section 9.2.3) and can replace the TRBv2 in the future so that this limitation can be overcome.

Main Features

The main features of the planned TrbNet version 2 protocol can be summarized as follows:

- Flexible packet size reduces the overhead during data transmission while keeping low latency trigger messages.
- Inherent data correction algorithms on all network links based on the current FOT link implementation.

- Network links can include a short handshake procedure to test a connection when it is established. During start-up, both link partners send a given pattern of control characters so that link start-up can be done faster and more reliably than in the current implementation.
- Increased data bandwidth at data rates of 2.5 GBit/s. The links should be able to operate at a throttled speed of approximately 2 GBit/s by adding a fixed number of idle words to adapt to slower nodes. A higher link speed may be selected if all hardware platforms are able to support it.
- Internal data busses can be resized to 32 Bit words without increasing the required resources substantially. This increases the internal bandwidth of nodes to 4 GBit/s and matches the increased link data rate. Slower nodes with 16 Bit word size might be supported if necessary.
- The internal logic can be further optimized if the header of each packet is separated from data. In all layers above the multiplexer, the header does not contain any information despite the 3 Bit sized packet type. A 32 + 3 Bit internal data bus can be used to transport this information in parallel to data words. Between the media interface and the multiplexer, a 32 + 6 Bit bus is required.
- Internal data transport can be simplified by using an optimized handshake implementation between implementation blocks.
- The transmission protocol can foresee short messages which do not require a reply from the nodes. The small message size guarantees that each message can be transported in one block of data and the channels are not blocked since no replies have to be sent. In combination, this avoids the possibility of collisions in multiple requester systems.

9.2.3. New Electronics

The currently used TRBv2 has already been used in various experiments but also showed some drawbacks: The embedded Linux platform has a quite low performance that is not sufficient for data acquisition at high rates. The DSP adds much complexity to the board layout but never has been fully employed in any read-out or data processing scheme. The TDC used on the board are not able to handle rates of above 4 MHz per channel. Most components on the board are already out-dated or not available any more.

The embedded Linux system usually only serves as a slow-control interface to program FPGA and TDC. In the meantime, these tasks can also be fulfilled by the DAQ network or by a direct implementation of Gigabit Ethernet within the FPGA. Moreover, powerful FPGA-based TDC have been developed at GSI [70].

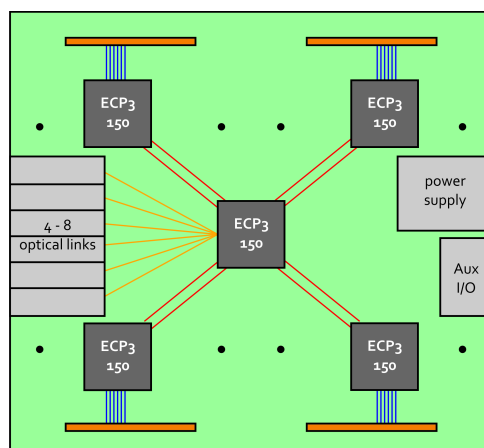


Figure 9.1.: The Trigger and Read-out Board version 3 (TRBv3) is a newly developed platform for data acquisition. Four FPGAs contain the data acquisition logic such as high-precision TDC circuits or front-end control features. Additional input electronics can be added to small AddOn boards. The fifth, central FPGA houses the interconnection to other electronics.

The result is a completely revised TRB version 3 which is currently (June 2011) in production. A schematic view is given in figure 9.1. Four FPGAs contain all front-end control logic needed, including up to 40 TDC channels. Each is supplied with a 200-pin Add-On connector that can be used to attach additional devices to the FPGA. For example, an external ADC board can be connected in order to measure both precise timing and amplitude of an incoming signal. One central FPGA serves as read-out hub. For data transmission it is supplied with 8 SFP transceivers that can be operated with various network protocols, including TrbNet and Gigabit Ethernet. The GbE implementation is bi-directional to transport event data as well as allow slow-control accesses to be made from an external computer.

Based on the experience to be collected with the TRB3, also new front-end electronics for the MDC system seems feasible. FPGA based TDC instead of the currently used custom ASIC-TDCs have several advantages: A much lower and less varying power consumption helps reducing the noise environment, the TDC have more configuration options, read-out can be accomplished in about 95% less time (10 ns instead of 200 ns per word), and finally dead-times are reduced due to multi-hit capabilities.

10. Kurzfassung

Ein zentraler Aspekt der modernen Kern- und Teilchenphysik ist die Erforschung des Phasendiagramms stark wechselwirkender Materie (Abbildung 1.1). Neben normaler Kernmaterie wird ein Phasenübergang in das sogenannte Quark-Gluon-Plasma bei hohen Temperaturen erwartet. Erste Anzeichen für einen solchen Phasenübergang wurden bereits experimentell beobachtet. Bei höheren Netto-Baryondichten werden weitere exotische Zustände vorhergesagt [2, 3]. Die einzige Möglichkeit, Materie bei hohen Temperaturen und Dichten im Labor herzustellen und zu untersuchen, stellen Experimente an Teilchenbeschleunigern dar, bei denen Atomkerne mit relativistischen Geschwindigkeiten zur Kollision gebracht werden. Bei ultrarelativistischen Geschwindigkeiten werden Zustände, wie sie kurz nach der Entstehung des Universums existiert haben, für kurze Zeit (weniger als 10^{-23} s) experimentell erschließbar.

Die hierbei entstehenden Teilchen können beispielsweise mithilfe eines Spektrometers nachgewiesen und charakterisiert werden. Ein typisches Spektrometer kombiniert verschiedene Detektoren und ein Magnetfeld und erlaubt es, Geschwindigkeit, Impuls, Energie, Ladung und Trajektorie jedes Teilchens präzise zu vermessen. In vielen Experimenten ist eine bestimmte Teilchenart von besonderem Interesse. Oft entstehen diese jedoch nur äußerst selten, sodass für eine spätere Analyse eine große Menge an Ereignissen aufgezeichnet werden muss. Aus diesen Gründen ist das Design des Datenaufnahmesystems (DAQ) für solche Experimente eine besondere Herausforderung.

Ein Detektorsystem, das speziell auf die Analyse von Elektron-Positron-Paaren bei Strahlenergien von 1 bis 3 GeV pro Nukleon ausgerichtet ist, ist HADES (High Acceptance Di-Electron Spectrometer, siehe Abbildung 10.1). Es befindet sich seit 2002 am GSI Helmholtzzentrum für Schwerionenforschung in Darmstadt. Das Spektrometer ist aus mehreren Detektorsystemen aufgebaut: Die Spurverfolgung der Teilchen wird durch vier Ebenen von Driftkammern (MDC) ermöglicht. Gleichzeitig erfolgt die Impulsbestimmung durch die Messung der Krümmung der Teilchenspuren in einem Magnetfeld. Die Flugzeit jedes Teilchens wird durch eine Flugzeitwand (TOF) und Resistive Plate Chambers (RPC) gemessen. Die Identifikation von Elektronen wird unterstützt durch einen Cherenkov-Detektor (RICH) sowie einen Pre-Shower-Detektor. Teilchen und Kernfragmente die das Target unter kleinen Winkeln verlassen, werden mit Hilfe eines Hodoskops nachgewiesen. Zwei Diamant-Detektoren (Start und Veto) registrieren alle Strahlteilchen vor beziehungsweise hinter dem Target. So kann der Zeitpunkt einer Kollision genau bestimmt und die Aufzeichnung von Daten, bei denen keine Reaktion im Target stattgefunden hat, unterdrückt werden.

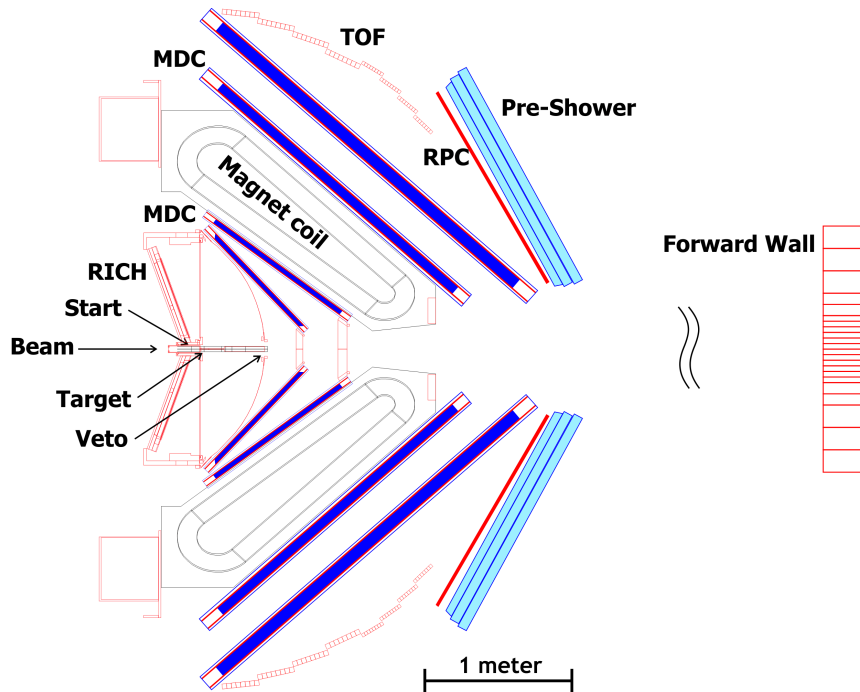


Abbildung 10.1.: Ein Querschnitt (Seitenansicht) durch das HADES Detektorsystem. Zur Vereinfachung sind nur zwei der sechs Sektoren gezeigt.

Auf Grund der Anforderungen geplanter Experimente mit Schwerionenstrahlen wurde im Jahr 2007 begonnen, das ursprüngliche Datenaufnahmesystem komplett zu ersetzen, um höhere Daten- und Ereignisraten zu ermöglichen. Hier sollen Raten von 20 kHz und Datenvolumina von 400 MByte/s erreicht werden, was einer Steigerung der Kapazität gegenüber dem alten System um etwa einen Faktor 30 entspricht. Für Experimente mit leichten Kernen soll die Ereignisrate bis zu 50 kHz betragen.

Das neu entworfene HADES DAQ-System basiert auf mit programmierbaren Logikbausteinen (Field Programmable Gate Array, FPGA) ausgestatteten Modulen. Diese bestehen aus den gleichen Grundbausteinen, die jeweils um detektorspezifische Bauteile ergänzt werden. Insgesamt sind über 500 solcher Module über das gesamte Detektorsystem verteilt. Die Kommunikation zwischen diesen Modulen erfolgt über optische Glasfaserverbindungen. Gegenüber herkömmlichen elektrischen Signalen hat dies mehrere Vorteile: die benötigten Kabel sind kleiner und sind unempfindlich gegenüber elektromagnetischen Störungen.

Das System wird durch ein zentrales Modul (Central Trigger System, CTS) gesteuert. Es empfängt analoge Informationen von verschiedenen Sub-Detektoren, auf deren Basis eine Entscheidung getroffen wird, ob ein Ereignis, das aufgezeichnet werden soll, stattgefunden hat. Diese wird über ein dediziertes Referenzzeitsignal an alle Submodule weitergeleitet, welche daraufhin die Detektordaten digitalisieren und zwischenspeichern. Das CTS versendet zur gleichen Zeit ein Datenpaket über das optische Netzwerk in dem genauere Informationen

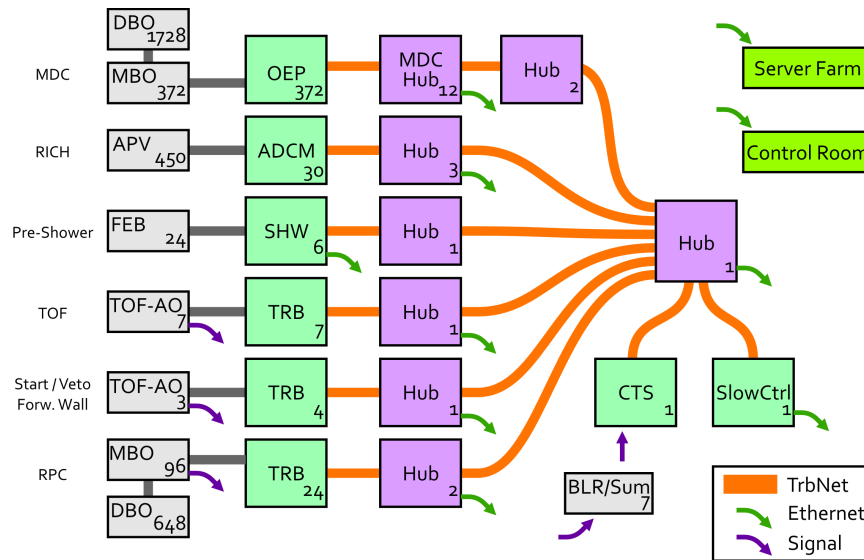


Abbildung 10.2.: Eine Ansicht des kompletten HADES DAQ Netzwerkes: Netzwerkknoten (Hubs) sind in violett dargestellt, alle Auslesemodule in grün, zusätzliche Frontend-Module in grau. Die Zahl der jeweils vorhandenen Module ist durch kleine Zahlen gekennzeichnet. Alle Module benutzen das gleiche Netzwerkprotokoll und ähnliche Bausteine. Untereinander sind die Module durch ein dediziertes Netzwerkprotokoll (TrbNet) verbunden während die Datenübertragung zu verschiedenen Servern über Gigabit Ethernet erfolgt.

zum Ereignis und die Verarbeitung der Daten enthalten sind. Während dieses Vorgangs ist das Versenden weiterer Trigger blockiert bis alle Module die Aufnahme der Daten des Ereignisses bestätigt und somit ihre Bereitschaft weitere Daten aufzunehmen signalisiert haben.

In einem zweiten Schritt werden die zwischengespeicherten Daten über das optische Netzwerk ausgelesen. In den Netzwerkknoten (Hubs) werden die individuellen Datenpakete zu größeren Datenströmen zusammengefasst, so dass aus über 500 einzelnen Paketen 30 Datenblöcke entstehen. Diese können dann an ein Rechnercluster ("Event Builder") weitergeleitet werden, wo sie zu kompletten Ereignissen kombiniert und gespeichert werden. Die Analyse der Daten läuft dann unabhängig von der Datenaufnahme zu einem späteren Zeitpunkt ab. Die Struktur des kompletten Netzwerkes ist in Abbildung 10.2 dargestellt.

Das Netzwerkprotokoll, das zum Datenaustausch verwendet wird, muss neben dem Transport von großen Datenmengen einige besondere Anforderungen erfüllen. Auf Grund des kontinuierlichen Datenaustauschs zwischen Triggersystem und Detektormodulen für jedes aufgezeichnete Ereignis muss die Latenz äußerst gering sein, das Versenden der Triggerinformation zu allen Modulen darf nicht mehr als $7 \mu\text{s}$ in Anspruch nehmen; gleiches gilt für den Rücktransport der Bestätigung zum Triggersystem. Wegen räumlicher Beschränkungen im Inneren des Detektors muss zudem jedwede Kommunikation über dieselbe physikali-

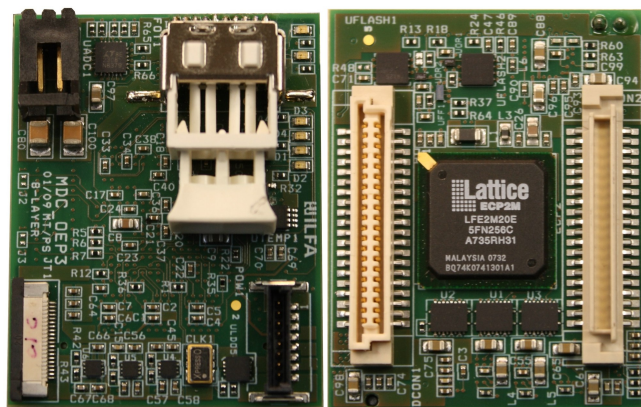


Abbildung 10.3.: Das Auslesemodul für die HADES Driftkammern. Die Größe des Boards ist auf 4 cm mal 5 cm beschränkt. Neben einem FPGA und optischen Transceiver enthält es Spannungsregler, 2 Flash Speicher, Signalkonverter und einen ADC zur Spannungsüberwachung.

sche Verbindung stattfinden, sodass aufgenommene Daten, Triggerinformationen sowie alle Kontroll- und Statusmeldungen miteinander kombiniert werden müssen. Für Netzwerkknoten muss es auf einfache Art möglich sein, mehrere Datenströme zu einem einzigen zusammenzufassen. Auf diesen Anforderungen basierend wurde ein dediziertes Protokoll, TrbNet, entwickelt. Für die zeitunkritische Datenübertragung zwischen Netzwerkknoten und Servern kann hingegen ein herkömmliches Netzwerk verwendet werden. Hier kommt eine Ethernet-Infrastruktur mit 1 bis 10 GBit/s zum Einsatz.

Um die verschiedenen Anforderungen der Datentypen zu berücksichtigen, verfügt das Netzwerkprotokoll über drei mit unterschiedlichen Prioritäten versehene virtuelle Kanäle. So ist es möglich, die Daten in jedem Netzwerkknoten voneinander getrennt zu verarbeiten und nur für die Übertragung auf den optischen Links in einen gemeinsamen Datenstrom zu bündeln. Die Abhängigkeiten der Datenströme untereinander wird so minimiert. Da die zeitkritischen Triggerinformationen den Kanal höchster Priorität benutzen, können sie praktisch verzögerungsfrei transportiert werden, auch wenn parallel erhöhte Datenmengen transportiert werden. Hierzu werden alle Daten in kleine Pakete von nur 80 Byte Größe aufgeteilt, wodurch ein Kanalwechsel nach spätestens 50 ns (bei 2 GByte/s Übertragungsgeschwindigkeit) möglich ist.

Die Ausleseelektronik für die HADES Driftkammern (MDC) stellt zusätzliche Herausforderungen auf Grund des stark beschränkten Platzangebots und der Nähe einzelner Module zu Strahl und Detektor. Es musste ein sehr kompaktes Modul von nur 4 cm auf 5 cm Größe entworfen werden (siehe Abbildung 10.3). Da die Analogelektronik des alten Auslesesystems beibehalten werden sollte, musste das Modul kompatibel zur vorhandenen Elektronik ausgelegt werden. Da die Module zudem im zusammengebauten Zustand des Detektors nicht

mehr zugänglich sind, waren zusätzliche Sicherheitsvorkehrungen zu treffen: Damit bei einem Update der Konfiguration keine Fehler entstehen können, verfügt das Modul über zwei getrennte Speicher, von denen nur einer neu beschrieben werden kann. Wegen ihrer hohen Empfindlichkeit benötigt die Analogelektronik präzise Versorgungsspannungen, die auf dem Modul geregelt werden können.

Die Nähe zum Detektor und anderen elektromagnetischen Störquellen verursacht zudem Übertragungsfehler in den optischen Transceivern. Um diese zu korrigieren, werden alle Daten vor der Übertragung mit Prüfsummen versehen, die auf Empfängerseite überprüft werden. Im Falle eines Fehlers werden die Daten verworfen und das Paket erneut vom Sender angefordert. Die Strahlung, die besonders im zentralen Bereich des Detektors auftritt, sorgt für zusätzliche Fehler, da sie die Konfiguration der Logikbausteine verändern kann. Das System muss also bei einer Fehlfunktion eines Frontend-Moduls dies erkennen und das Modul selbständig aus dem System entfernen, damit die Datenaufnahme des restlichen Systems weiterlaufen kann, bis das fehlerhafte Modul wieder einsatzbereit ist. Diese Funktion übernimmt der dem Modul nächstgelegene Netzwerk-Hub.

Ein weiterer wichtiger Aspekt der Erneuerung des Auslesesystems war die Integration zusätzlicher Kontroll- und Überwachungsfunktionen für alle Detektorsysteme. Diese wurden zunächst dadurch erleichtert, dass nun alle Systeme über ein gemeinsames Netzwerk miteinander verbunden sind und somit auch dieselben Programme verwendet werden können. Zudem ist es nun möglich, jedes Frontend-Modul individuell anzusprechen und seinen Zustand abzufragen. Jedes Modul verfügt über einen Satz an global standardisierten Registern, die eine Auswertung der Informationen vereinfachen. Daneben hat jedes System zusätzliche Funktionen implementiert, um beispielsweise Parameter für die Detektorelektronik zu setzen oder über verschiedene Busprotokolle mit anderen Bausteinen zu kommunizieren. Auf Softwareseite existiert eine Reihe von Werkzeugen, mit denen der Zugriff auf diese Funktionen vereinfacht wird.

Das komplette DAQ-System wurde während mehrerer Testexperimente in den Jahren 2010 und 2011 in Betrieb genommen und erfolgreich validiert. Die Latenzzeiten der Übertragung von Triggerinformationen liegt mit maximal $5 \mu\text{s}$ innerhalb der Spezifikationen. In einem Leistungstest mit künstlich erzeugten Daten erreichte das System Ereignisraten von bis zu 68 kHz und Datenraten von über 800 MByte/s. Diese Werte liegen weit über den Anforderungen während eines Experiments. Im August 2011, während des Gold-Testexperiments bei 1.25 AGeV Strahlenergie und typischen Strahlintensitäten von $1,4 \cdot 10^6$ Ionen pro Sekunde, konnten Datenraten von über 300 MByte/s und Ereignisraten von 15 kHz erreicht werden. Diese Werte liegen leicht unter den geplanten Raten, obwohl die Datenaufnahme mit einer Totzeit von nur etwa 15% nicht saturiert war. Zum einen konnte auf Grund der Last in den Detektoren die Strahlintensität nicht weiter erhöht werden, zum anderen konnten nur etwa 50% aller Ereignisse aufgezeichnet werden. Der Grund hierfür ist die Mikro-Struktur des Ionenstrahls mit Intensitätsspitzen auf einer Skala von bis zu 1 ms. Der Prozess der Extraktion

aus dem Beschleuniger erlaubt es nicht, einen kontinuierlichen Strahl zu erzeugen; stattdessen entstehen abwechselnd Phasen hoher und sehr niedriger Intensität. Hierdurch treten Ereignisse vermehrt innerhalb kurzer Zeitabstände auf und können aufgrund der Totzeit von etwa $30\ \mu\text{s}$ pro Ereignis nicht verarbeitet werden. Die erwartete Übertragungsfehlerrate von durchschnittlich einem Fehler pro Sekunde und die Ausfallrate der Frontends von etwa einem Ausfall pro Stunde konnten sämtlich durch die Fehlererkennungs- und Fehlerkorrektur-Mechanismen abgefangen werden, die auf Grund der Erfahrungen in früheren Tests implementiert wurden.

Von der für HADES entwickelten Elektronik und dem universellen und performanten Auslesenetzwerk profitieren inzwischen auch zahlreiche weitere Experimente bei FAIR und anderen Forschungseinrichtungen. Das zentrale Ausleseboard für alle zeitmessenden Detektoren (TRB2) wird vielfältig einsetzbar und wird unter anderem am LIP in Coimbra (Portugal) zur Auslese eines PET-Scanners eingesetzt. Weitere Module sind für Testaufbauten und Detektortests im Einsatz. Der in Frankfurt entwickelte Micro-Vertex-Detektor für das CBM-Experiment bei FAIR benutzt beispielsweise die gleiche Netzwerk- und Softwareinfrastruktur wie HADES. Aus den Erfahrungen mit dem TRB2 wurde ein Nachfolgemodul gebaut, das eine höhere Zeitauflösung und höhere Flexibilität bringt. Im Gegensatz zu dedizierten Bausteinen zur Zeitmessung wird hier ein Konzept zur Zeitmessung direkt in FPGAs verwendet. Weiterhin kann das Board als universelle Logik-Plattform für beliebige andere Funktionen dienen. Die Erweiterbarkeit um notwendige Zusatzfunktionen ist durch Aufsteckmodule gewährleistet.

A. TrbNet Definitions

All boards within the DAQ network are accessible individually and in groups depending on their type. These accesses are done using network addresses that also reflect the position of a front-end within the detector. The full set of addresses is described in section A.1.

The TrbNet nodes have a defined application interface and are configurable by various settings. The application interface to the network in endpoints consists of three parts: The trigger interface, the event data interface and the slow control interface. These interfaces and their configuration options are described in sections A.2, A.3 and A.4, respectively. Inside network hubs, the streaming interface is used to receive event data from the network and forward it to the servers. This interface is shown in section A.5.

A.1. Network Addresses

All network nodes are individually addressable by a 16 Bit network address. This address is assigned during the start-up process as described in section 3.3.1. Table A.2 shows the addresses assigned to all network nodes while table A.1 gives an overview of all available broadcast addresses that are used to address a given sub-set of nodes at once. All broadcasts are located in the address range between 0xFE00 and 0xFFFF.

Appendix A. TrbNet Definitions

Address	Broadcast to...	Number of Nodes
0xFFFF	All nodes (global broadcast)	approx. 550
0xFFFE	Network hubs	90
0xFFFFD	MDC front-ends (OEP)	372
0xFFFFB	RICH front-ends (ADCM)	30
0xFFFF7	Shower front-ends (Shower-AddOn)	12
0xFFEF	TOF front-ends (TRB)	7
0xFFDF	RPC front-ends (TRB)	24
0xFF7F	Nodes with Ethernet Bridges	25
0xFE11	MDC-Hub Nodes with FOT links	48
0xFE15	MDC-Hub central data combining hubs	12
0xFE21	Shower Front-end read-out handler for ADC 1 – 6	6
0xFE22	Shower Front-end read-out handler for ADC 7 – 12	6
0xFE23	Shower Front-end central hub FPGA	6
0xFE31	Hub AddOn FPGA 1 (with 12 optical links)	11
0xFE33	Hub AddOn FPGA2 (with GbE interface)	11

Table A.1.: The network addressing scheme foresees a set of broadcast addresses to access all or a sub-set of network nodes at once.

Address	Broadcast to...
0x0002	CTS, read-out of event data
0x0003	CTS, configuration interface
0x2000 – 0x235F	MDC front-ends (OEP). 2 nd digit: MDC plane (0 – 3). 3 rd digit: sector (0 – 5). 4 th digit: front-end board number.
0x3000 – 0x3055	RICH front-ends. 3 rd digit: sector (0 – 5). 4 th digit: front-end board number (0–5).
0x3200 – 0x3252	Shower front-ends. 3 rd digit: sector (0 – 5). 4 th digit: FPGA number (0 – 2).
0x4000	Start / Veto detector
0x4400 – 0x4420	Forward Wall front-ends. 3 rd digit: front-end board number (0 – 2).
0x4800 – 0x4853	RPC front-ends. 3 rd digit: sector (0 – 5). 4 th digit: front-end board number (0 – 3).
0x4C00 – 0x4C50	TOF front-ends. 3 rd digit: sector (0 – 5).
0x1000 – 0x1154	MDC-Hub boards. 2 nd digit: inner (0) or outer (1) MDC planes. 3 rd digit: sector (0 – 5). 4 th digit: FPGA number (0 – 3).
0x8000 – 0x8001	Central Network Hub. 4 th digit: FPGA number (0 – 1)
0x8100 – 0x8111	MDC Hubs. 3 rd digit: inner (0) or outer (1) MDC planes. 4 th digit: FPGA number
0x8300 – 0x8321	RICH Hubs. 3 rd digit: hub number (0 – 2). 4 th digit: FPGA number.
0x8400 – 0x8411	RPC Hubs. 3 rd digit: hub number (0 – 1). 4 th digit: FPGA number.
0x8500 – 0x8801	Other network hubs: 2 nd digit: sub-system: 0x5 Shower, 0x6 TOF, 0x7 Forward Wall, 0x8 Start/Veto and CTS. 4 th digit: FPGA number.

Table A.2.: The network addresses are assigned in a way that directly identifies the sub-system and the physical position within the detector system.

Appendix A. TrbNet Definitions

Port	Width	Description
LVL1VALIDTIMINGTRGOUT	1	The LVL1 Handler sets this signal for one clock cycle after a valid reference time signal has been identified
LVL1VALIDNOTIMINGTRGOUT	1	The LVL1 Handler sets this signal for one clock cycle when a valid RTL trigger has been received
LVL1INVALIDTRGOUT	1	This signal is set for one clock cycle if a LVL1 trigger is received that has not been preceded by a reference time signal

Table A.3.: Part 1 of the Trigger Interface: Trigger Validation Ports. Depending on the trigger type, either a valid timing trigger or a valid RTL trigger is announced. In case of an error, an invalid trigger may be reported as described in section 4.4.

A.2. Trigger Interface

The Trigger Interface is part of the connection between the read-out logic and the network. Here, trigger information is offered to the read-out logic and the busy release is accepted. Table A.3 shows the ports for trigger validation and handshakes. Table A.4 shows the ports for trigger information and table A.5 shows the ports for error information.

All trigger information ports (see table A.4) are valid as long as `LVL1TRGDATAVALIDOUT` is set. This signal is high after the LVL1 trigger has been received until the busy release has been sent.

The behavior of the LVL1 Trigger Handler is described in section 4.4. Here, 8 different input scenarios are described. The corresponding timing diagrams are shown below.

Port	Width	Description
LVL1TRGDATavalidOUT	1	The trigger type, number, information and code is valid. This signal rises after the LVL1 Trigger has been received and goes low after the busy release has been sent
LVL1TRGTYPEOUT	4	Trigger Type. See table 4.1
LVL1TRGNUMBEROUT	16	Running Trigger Number
LVL1TRGCODEOUT	8	Random Trigger Code
LVL1TRGINFORMATIONOUT	24	Trigger Information. See table 4.2
LVL1INTTRGNUMBEROUT	16	Internally generated trigger number. The value is valid when the trigger logic is not busy and shows the next expected trigger number.

Table A.4.: Part 2 of the Trigger Interface: Trigger Information. All ports are valid while LVL1TRGDATavalidOUT is set.

Port	Width	Description
LVL1TRGMULTIPLETRGOUT	1	If a second valid reference time signal is detected on the input, this bit is set.
LVL1TRGTIMEOUTDETECTEDOUT	1	The delay after the reference time signal until the LVL1 trigger information is received exceeded the maximal time limit
LVL1SPURIOUSTRGOUT	1	A reference time signal was detected but the LVL1 trigger information announced a RTL-trigger
LVL1MISSINGTMGTRGOUT	1	No reference time signal but a valid LVL1 trigger has been received
LVL1TRGSPIKEDETECTEDOUT	1	A short spike on the reference time input has been received
LVL1LONGTRGOUT	1	The reference time signal was much longer than expected.

Table A.5.: Part 3 of the Trigger Interface: Error Information. On these ports, all different error conditions are reported. See section 4.4 for details. All bits are set when the error condition is detected and are valid until the busy release has been sent.

Appendix A. TrbNet Definitions

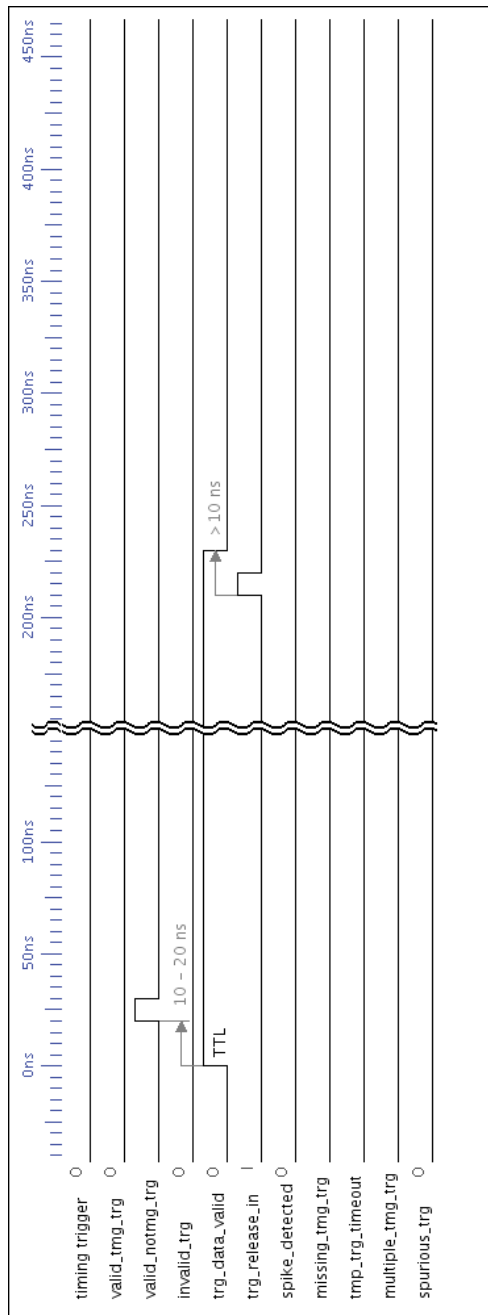
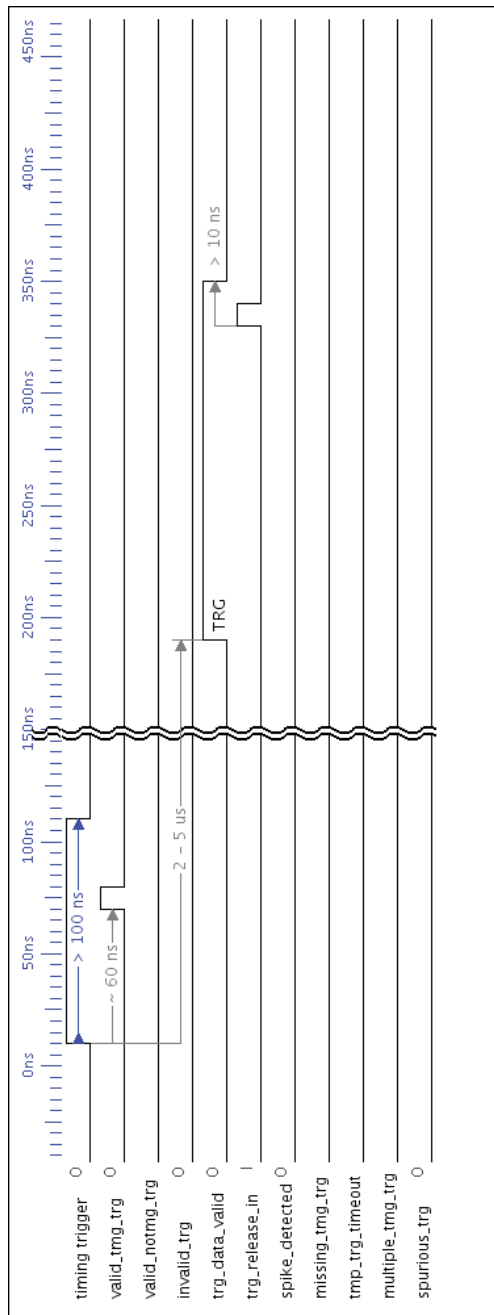


Figure A.1.: Timing Diagram: Behavior of the LVL1 Trigger Handler. Input case 1 (see section 4.4).

Figure A.2.: Timing Diagram: Behavior of the LVL1 Trigger Handler. Input case 2 (see section 4.4).

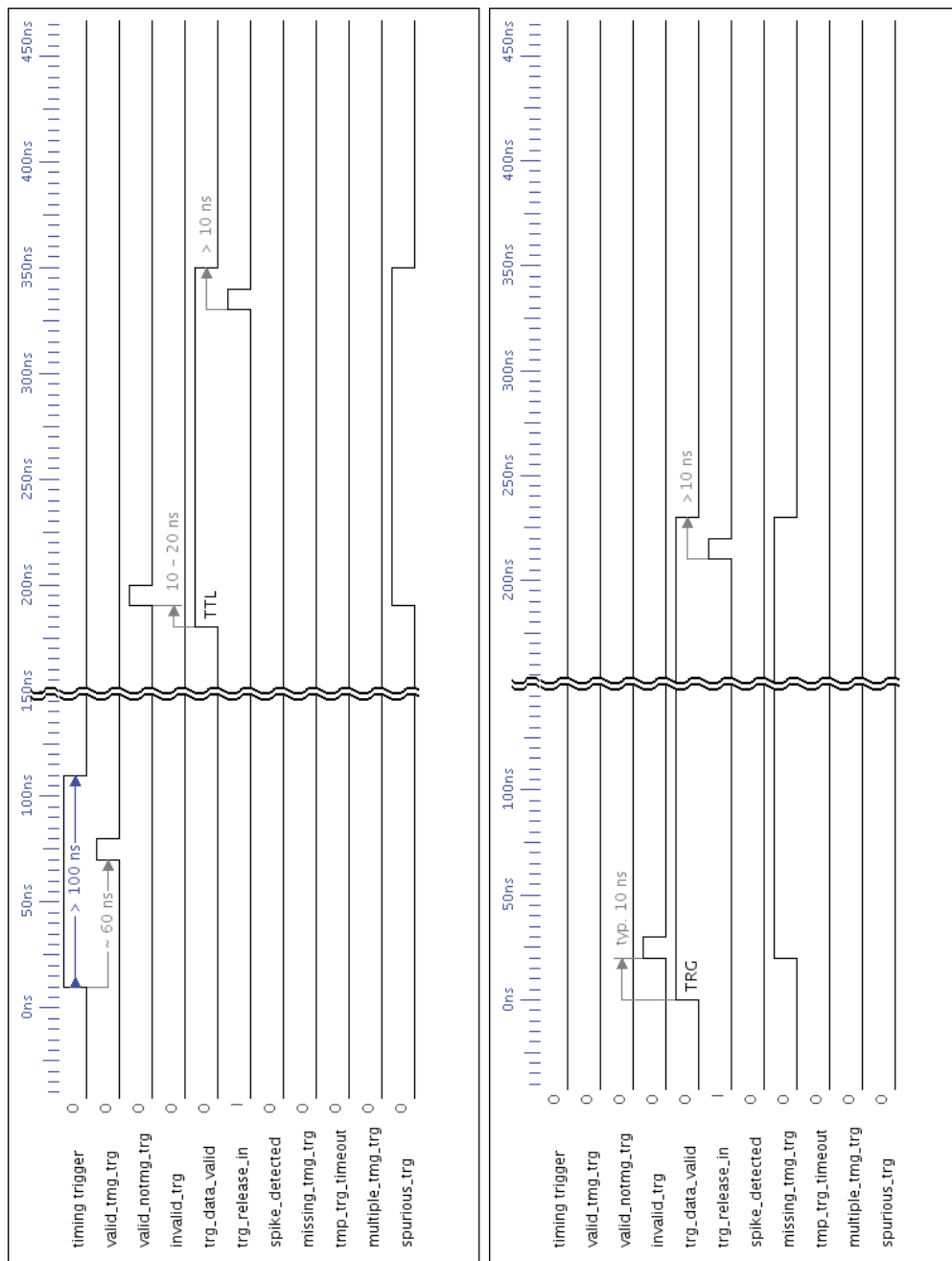


Figure A.3.: Timing Diagram: Behavior of the LVL1 Trigger Handler. Input case 3 (see section 4.4).

Figure A.4.: Timing Diagram: Behavior of the LVL1 Trigger Handler. Input case 4 (see section 4.4).

Appendix A. TrbNet Definitions

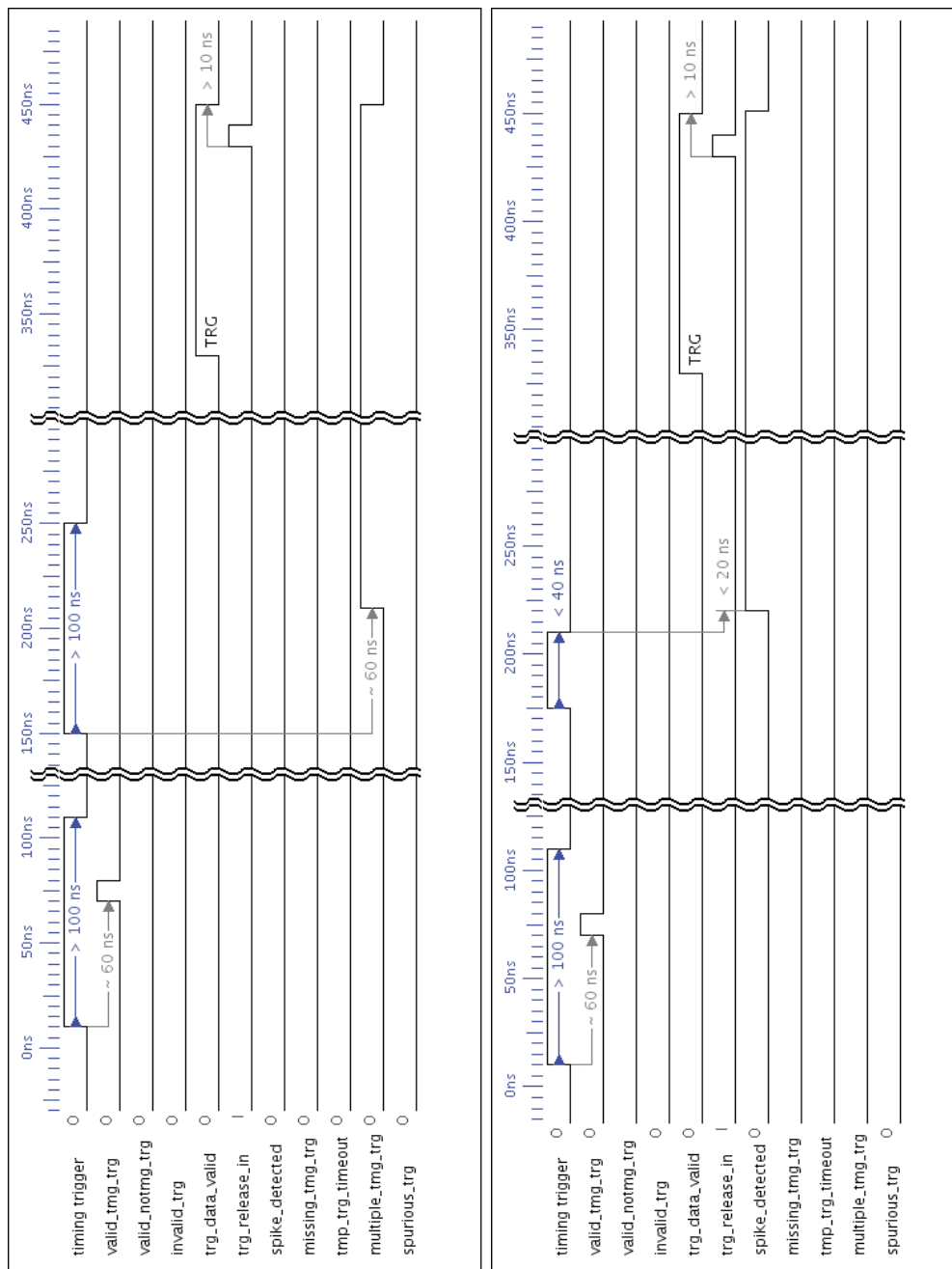


Figure A.5.: Timing Diagram: Behavior of the LVL1 Trigger Handler. Input case 5 (see section 4.4).

Figure A.6.: Timing Diagram: Behavior of the LVL1 Trigger Handler. Input case 6 (see section 4.4).

A.3. Event Data Interface

The Event Data Interface is used to transport data between the front-end logic and the TrbNet endpoint logic as described in section 5.2. Figure A.7 shows a generic example of a transaction on this interface. The interface consists of four signals which are described in table A.6. The configuration of the interface regarding bus widths, buffer depths and general behavior is shown in table A.7.

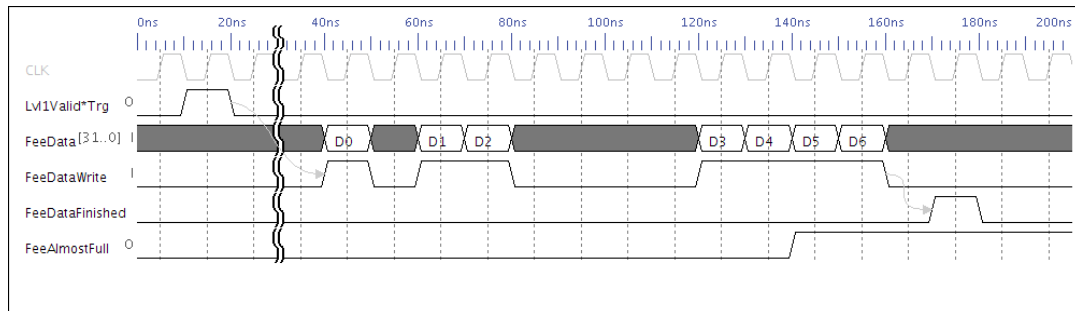


Figure A.7.: A transaction on the Event Data Interface. After the trigger, the front-end sends a total of 7 data words before it finishes writing data. During writing of the event, the buffer raises its almost-full flag indicating that the busy release will be delayed until the fill-level is again below a critical mark.

Port	Width	Description
FEEDATAIN	32	Data input for raw event data
FEEDATAWRITEIN	1	Strobe signal to validate event data
FEEDATAFINISHEDIN	1	Strobe to signal the end of event data
FEEDATAALMOSTFULLOUT	1	Status of the data buffer. This signal is for informational purposes only. If the endpoint is configured correctly it handles buffer full states automatically

Table A.6.: The Event Data Interface is part of the connection between the read-out logic and the network. Here, event data is transported to the endpoint after a valid trigger.

Appendix A. TrbNet Definitions

Generic	Type	Description
TIMINGTRGRAW	bool	Configuration of the LVL1 handler, if set to <code>CNO</code> , a synchronized strobe signal is expected as trigger signal. Otherwise, the input should be a 100 ns long raw reference time signal
DATAINTERFACENUMBER	int	The number of independent data inputs to the endpoint data handler
DATABUFFERDEPTH	int	The depth of one data buffer as calculated by $value = \log_2(\text{number of words})$ The supported values range from 9 to 14
DATABUFFERWIDTH	int	Width of the data port in bits. The maximum width is 32 Bit
DATABUFFERFULLTHRESH	int	The full threshold for the buffer handling. Depending on the setting of <code>TRGRELEASEAFTERDATA</code> , the value must be either one maximal frame size (if set to <code>CYES</code>) or two maximal frame sizes (if set to <code>CNO</code>) below the total size of the data buffer
TRGRELEASEAFTERDATA	bool	The default behavior (when set to <code>CYES</code>) is, that the trigger busy release is sent to the CTS only after the the data finished signal from each data port has been received. If set to <code>CNO</code> , the busy release is controlled by <code>FEETRGRELEASEIN</code>
HEADERBUFFERDEPTH	int	Depth of the buffer for event headers. Available depths are 9 to 14, calculated in the same manner as <code>DATABUFFERDEPTH</code>
HEADERBUFFERFULLTHRESH	int	The full threshold for the event header buffer. This value should be set to a few events less than the full buffer depth

Table A.7.: The Event Data Interface is part of the connection between the read-out logic and the network. In the table, all available configuration options are shown.

A.4. Slow-Control Interface

The slow control endpoint can be configured by a number of generic settings as shown in A.8. The I/O ports not related to the internal data bus are given in table A.9.

All busses of the RegIO register interface are shown in timing diagrams A.8 and A.9. These show all possible combinations of read or write requests and matching reactions of the front-end logic. If the front-end logic does not respond within few clock cycles, a time-out is generated and the slow-control request is completed with an error message. If a request can not be fulfilled because the address does not correspond to any defined functionality, the request is cancelled by a strobe on the “unknown address” signal. This can also be used e.g. if the addressed register of a write access is read-only.

If a request is possible but can not be fulfilled at the time of the request, the “no more data” signal is set. A read access may fail if a register without valid data or an empty Fifo is addressed. Vice versa, a write access fails if the register is blocked, for example in case the register is a control register for a logic block which is busy and can not accept new commands.

If a write access succeeds, the front-end logic performs the write and sets an acknowledge signal. A read access is successfully ended by providing the requested data on the data port and setting the “data valid” flag for one clock cycle.

The block access command is not visible on the interface. RegIO converts any block access to a sequence of single read/write operations with properly adjusted addresses. This is feasible since the data bus is faster than transmission of slow-control data via TrbNet by a factor 5, i.e. each access can spend 5 clock cycles without slowing data transport.

Appendix A. TrbNet Definitions

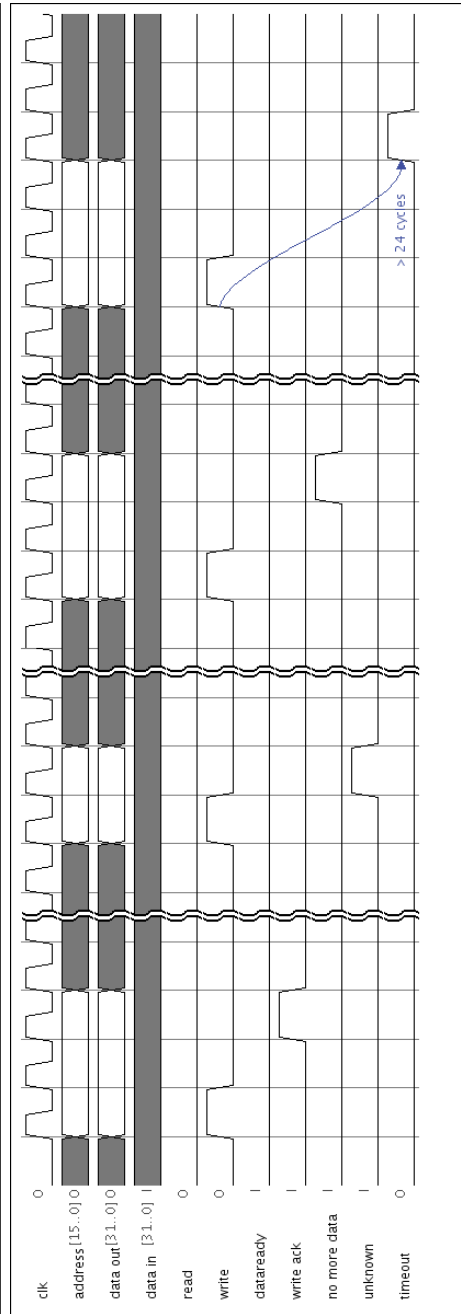
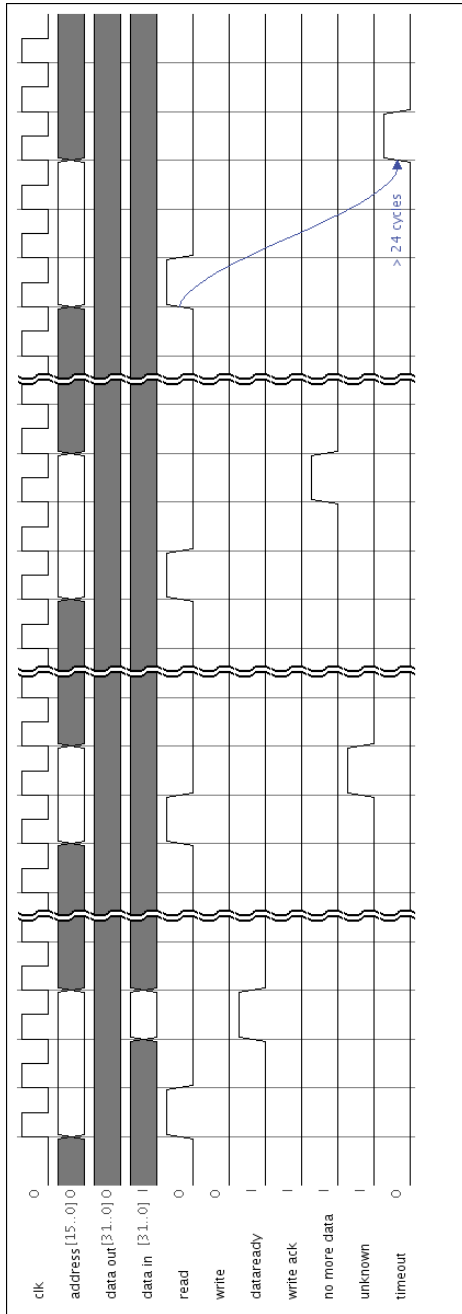


Figure A.8.: RegIO Data Bus - Possible read cycles

Figure A.9.: RegIO Data Bus - Possible write cycles

Generic	Type	Description
ADDRESSMASK	15 – 0	A bit mask to be applied when checking the address of a request. Can be used to have one front-end accessible by several addresses. Default 0xFFFF.
BROADCASTBITMASK	7 – 0	Bitmask to be applied on lower 8 bits of hexFxx broadcast addresses
BROADCASTSPECIALADDR	7 – 0	Lower 8 bits of hexFExx special broadcasts
REGIOINITADDRESS	15 – 0	Initial network address
REGIOINITENDPOINTID	31 – 0	Endpoint ID to distinguish several FPGA with identical unique IDs
REGIOCOMPILETIME	31 – 0	Unix time stamp of compilation time of the design (register hex40)
REGIOCOMPILEVERSION	31 – 0	Value for board design register hex41
REGIOHARDWAREVERSION	31 – 0	Value for board information register hex42
REGIONUMSTATREGS	int	Log base 2 of number of user defined status registers
REGIONUMCTRLREGS	int	Log base 2 of number of user defined control registers
REGIOINITCTRLREGS	511 – 0	Initial values of user defined control registers
REGIOUSE1WIRE	int	Select 1-wire interface: <code>CNO</code> : No 1-wire, endpoint ID is written manually. <code>CYES</code> : Normal 1-wire interface. <code>CMONITOR</code> : 1-wire interface in listen-only mode - a 1-wire master that issues the commands exists in a different place
REGIOUSEVARENDPOINTID	bool	Endpoint ID is not given by generic but by a variable input port
CLOCKFREQUENCY	int	Clock frequency in MHz. Used to adjust internal timers

Table A.8.: The slow control interface can be configured by a set of generic values.

Appendix A. TrbNet Definitions

Port	Width	Description
REGIOCOMMONSTATREG	288	The common status registers
REGIOCOMMONCTRLREG	96	The common control registers
REGIOCOMMONSTATSTROBE	9	Read strobes for the common status registers
REGIOCOMMONCTRLSTROBE	3	Write strobes for the common control registers
REGIOSTATREG	var.	The user status registers
REGIOCTRLREG	var.	The user control registers
REGIOSTATSTROBE	var.	Read strobes for the user status registers
REGIOCTRLSTROBE	var.	Write strobes for the user control registers
ONEWIREINOUT	1	Bi-directional I/O to 1-wire sensor
ONEWIREMONITORIN	1	Input to the 1-wire monitor entity
ONEWIREMONITOROUT	1	Output to an optional 1-wire monitor entity in a different device
REGIOVARENDPOINTID	16	Variable endpoint id. Used if corresponding generic is enabled
TIMEGLOBAL	32	Global timing registers, counting microseconds
TIMELOCAL	8	Local fine timing, running with chip frequency, reset every microseconds
TIMESINCELASTTRG	32	Clock cycles since reception of last reference time signal
TIME TICKS	2	Clock strobe signals send every microsecond (bit 0) or every 1.024 milliseconds (bit 1)

Table A.9.: The slow control interface has a number of ports of simple registers and further information.

Port	Width	Description
CTSSTARTREADOUTOUT	1	This flag is set while a read-out transaction is going on. The signal rises a few clock cycles after the read-out request has been forwarded toward the front-ends. The falling edge of the signal follows the CTSFINISHEDIN signal from the attached logic
CTSNUMBEROUT	16	On these ports, the information transported with the read-out request is shown. These outputs are valid while CTSSTARTREADOUTOUT is set
CTS CODEOUT	8	
CTSINFORMATIONOUT	16	
CTSREADOUTTYPEOUT	4	

Table A.10.: The ports of the Streaming API interface on which information from the CTS is given. This interface is used to transport data out of TrbNet to another medium like Gigabit Ethernet

A.5. Streaming API

The Streaming API is used to transfer event data from TrbNet to another destination or network protocol is described in section 5.6. The ports of this interface can be divided into three parts: Ports connection to the CTS for read-out request (see table A.10), the interface to send an acknowledge and the remaining event data to the CTS (see table A.11 and the interface on which data from the front-ends is received (see table A.12). Figures A.10 and A.11 show the typical procedure of a read-out request received from the CTS followed by event data sent by the front-ends. The remaining empty event and termination sent to the CTS is shown in figure A.12.

Appendix A. TrbNet Definitions

Port	Width	Description
CTSDATAIN	16	The Data that is to be sent to the CTS. The first word for each event must be a valid Event Information word, the subsequent words are written in 16 Byte words, MSB first.
CTSDATAREADYIN	1	Data is being offered on CTSDATAIN. The data is read by the interface in the same clock cycle as CTSDATAREADYIN and CTSREADOUT both are set.
CTSREADOUT	1	The interfaces currently reads the data offered on CTSDATAIN. See CTSDATAREADYIN
CTSLengthIN	16	The length of data, counted in 32 bit words. This word must be valid while the Event Information word is offered to the interface
CTSREADOUTFINISHIN	1	Strobe signal to show that the full read-out process has been finished
CTSSTATUSBITSIN	32	The status and error information to be passed to the CTS. See section 5.8 for details. This information must be valid when CTSREADOUTFINISHEDIN is high

Table A.11.: The ports of the Streaming API interface to send data to the CTS. This interface is used to transport data out of TrbNet to another medium like Gigabit Ethernet

Port	Width	Description
FEEDATAOUT	16	Data from the front-ends
FEEDATAREADYOUT	1	Data on FEEDATAOUT is valid
FEEREADIN	1	Attached logic currently reads data from front-end. See also CTSDATAREADYIN
FEESTATUSBITSOUT	32	Error and status information from front-ends. Data is valid for 50 ns after falling edge of FEEBUSYOUT
FEEBUSYOUT	1	High while waiting for data from front-ends

Table A.12.: The ports of the Streaming API interface where data from front-ends is offered. This interface is used to transport data out of TrbNet to another medium like Gigabit Ethernet

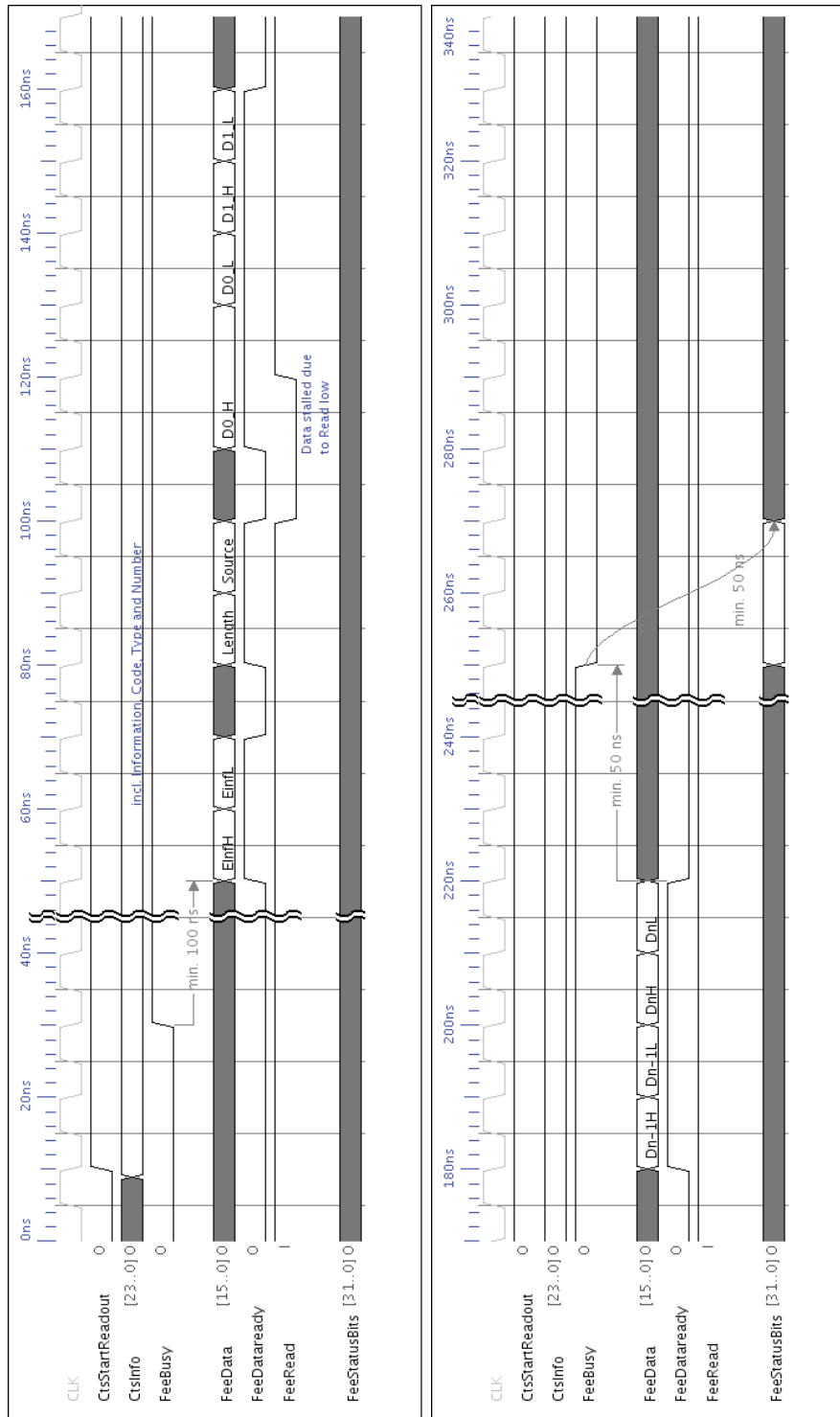


Figure A.10.: Streaming interface I: Start of readout and data receiving from FEE

Figure A.11.: Streaming interface II: End of readout data from FEE

Appendix A. TrbNet Definitions

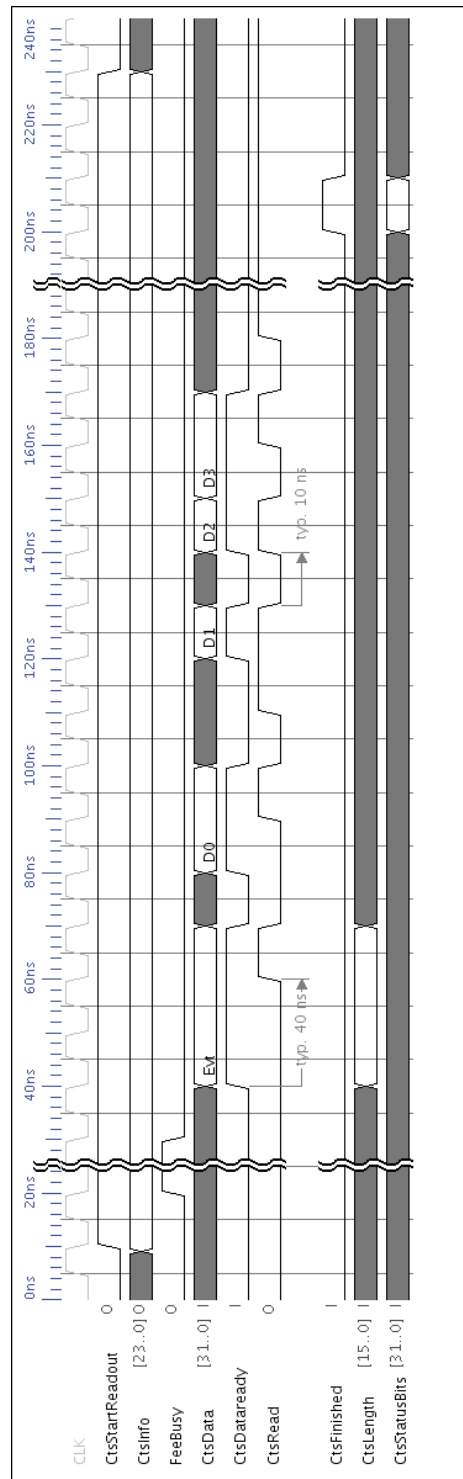


Figure A.12.: Streaming interface III: End of transaction, sending answer to CTS

B. Trigger Distribution

The Central Trigger System is able to generate a set of different reference time signals depending on the detector they are dedicated for. The signalling standard used is either LVDS for short cables or PECL for longer signal distances. In general, the last part of each reference time cable is a LVDS signal for compatibility reasons. The signals output by the CTS are distributed using twisted pair cables, either with flat-cables or with shielded RJ-45 cables. The finer distribution to all 500 front-end boards is done on Fan-out boards with up to ten outputs each which are mounted within the detector.

Most detectors have a special requirement on the arrival time of the reference signal and cable length has been adjusted accordingly. For some signals an additional delay can be generated inside the CTS. All cables, signals and requirements for the different subsystems are listed in figure B.1.

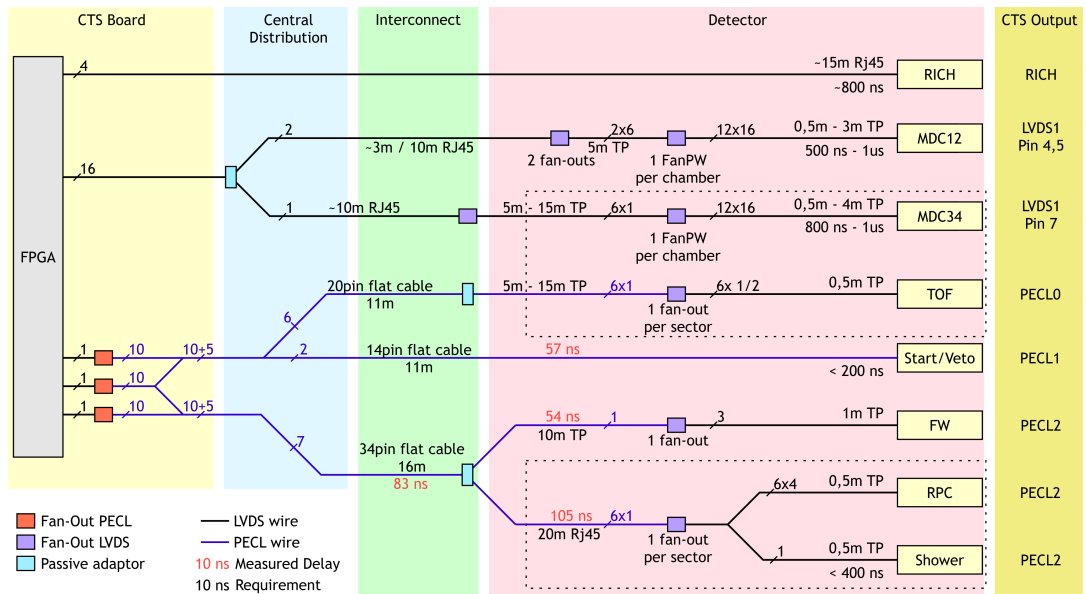


Figure B.1.: The trigger signals are generated in the Central Trigger System and distributed to all detectors. The type and length of wires and all intermediate electronics are depicted.

C. Slow Control Registers

C.1. Common Slow Control Registers

The TrbNet slow-control channel features a rich set of registers common to all network nodes. This simplifies the effort needed for monitoring. The common features can be sub-divided into status (tables C.1 and C.2) and control registers (tables C.3 and C.4) as well as further information registers (see table C.5).

Bits	Description
31 – 20	temperature (in 1/16th degree)
19 – 16	reserved
15	link data error (e.g. code violation)
14	single Event Upset detected
13	timing Trigger Input
12	last event sent on read-out channel is broken
11	severe problem in event data buffer / IPU request handler
10	IPU requested event partially not found / data missing
9	IPU Event not found
8	timing trigger missing
7	frontend error
6	frontend not configured
5	IPU channel counter mismatch
4	LVL1 trigger counter mismatch
3	note flag
2	warning flag
1	error flag
0	serious error flag

Table C.1.: The Common Status Register 0 contains several flags to signal the over-all status of the front-end. Additionally, the temperature of the board is reported.

Register	Bits	Description
1	15 – 0	Trigger: Internal LVL1 trigger number
1	31 – 16	Read-out: Number of last event read out
2	13 – 4	Trigger: Error flags related to reference time
2	26 – 16	Trigger: Delay between reference time and LVL1 trigger information
3	15 – 0	Trigger: Number of pulses (i.e. rising edges) seen on the reference time input
3	31 – 16	Trigger: Length of last reference time signal (in 10 ns units)
4	3 – 0	Media Interface: Counter for reset signals received via optical link
4	23 – 16	Media Interface: Number of retransmission requests received
4	31 – 24	Media Interface: Number of retransmission requests sent
5	15 – 0	Trigger: Lower 16 Bit of trigger information received with last LVL1 trigger
5	19 – 16	Trigger: Type of last LVL1 trigger
5	23 – 20	Trigger: Lower 4 Bit of last LVL1 trigger number
5	31 – 24	Trigger: Code of last LVL1 trigger
6	15 – 0	Trigger: Number of invalid triggers, i.e. LVL1 triggers not preceded by a reference time signal
6	31 – 16	Trigger: Number of occurrences of multiple valid reference time signals before one LVL1 trigger
7	15 – 0	Trigger: Number of spikes seen on the reference time input (sampled with system clocks)
7	31 – 16	Trigger: Number of spurious triggers, i.e. reference time signals preceding a RTL trigger
8	15 – 0	Trigger: Number of spikes on the reference time input using asynchronous detection

Table C.2.: The Common Status Registers 1 to 8 contain various information about the trigger logic and status of the media interface error correction.

Appendix C. Slow Control Registers

Bits	Description
23	Send front-ends to stand-by mode (“End Run”)
22	Initialize front-ends (“Begin Run”)
16	Dummy reference time signal generated internally
15	Reboot FPGA
9	Reinitialize front-ends with next trigger
5	Reset status and error counters
4	Reset persistent error flags
3	Master reset for front-end logic
2	Empty IPU chain / reset IPU logic
1	Reset trigger logic
0	Reset frontends

Table C.3.: The Common Control Register 0 (CCR0) contains strobe signals that trigger certain functions in the network nodes, such as partial resets, status register erasing and initialization.

Bits	Description
31	Enable reference time input
30	Enable sending debug information
29	Invert reference time input signal
28	Single Event Upset Detection enable
27	Enable error correction in media interface
23 – 20	Select event data format
15 – 0	Enable for individual frontends

Table C.4.: The Common Control Register 2 (CCR2) contains configuration for basic functionalities. Settings for the reference time input and the data to be sent for each event can be selected.

Register	Bits	Description
40	31 – 0	Compilation time of the FPGA design. The value is a Unix timestamp
41	31 – 0	FPGA design version. The version number can be assigned freely for each design
42	15 – 0	User defined, minor hardware version number. This value can be freely chosen, e.g. to mark designs for slightly modified FPGA boards.
42	31 – 16	Global hardware identifier. The number identifies the type of hardware the design is intended to be used on. The value is globally assigned as listed in [63].
50	31 – 0	Global clock. All designs contain a clock, counting microseconds since the last clock reset. Clocks can be adjusted to any value by writing to this register.
51	31 – 0	Time since last reference time signal, counting clock cycles of the FPGA.

Table C.5.: The Board Information and Time Registers provide additional information about the hardware and FPGA design version as well as a global clock.

C.2. Hub Registers

Hub Configuration Registers

0xC0, 0xC1, 0xC3: Port Enable The three registers are used to switch ports of the hub on or off individually for each channel (in the given order: LVL1 trigger, read-out and slow-control). Each bit of the registers corresponds to one port of the hub. If set to 1, the port is logically enabled inside the hub logic and disabled otherwise. If a port is off, the link is still established, but no data is sent to this port nor an reply is expected from this port.

0xC5: Time-out Configuration The time-outs (i.e. how long the hub logic waits for a reply from each port) can be adjusted for each channel. The available options are: 0: off, 1: 128 ms, 2: 256 ms, 3: 512 ms, 4: 1 s, 5: 2 s, 6: 4 s, 7: 8 s. In the network, only the hub connected to a failing node is supposed to generate a time-out. Hence, on a central hub the time-out delay has to be higher than on a network hub deeper inside the network structure. Followingly, the time-out delay can be decreased in steps of 2 ms according to the hub offset setting. The bits of this register are defined corresponding to the following list:

Bit 2 – 0 LVL1 channel

Bit 6 – 4 Read-out channel

Bit 14 – 12 Slow-control channel

Bit 19 – 16 Hub offset

Hub Monitor and Status Registers

0x80, 0x81, 0x83: Waiting for Reply The three registers show the current status of each port of the hub on each of the network channels (in the given order: LVL1 trigger, read-out and slow-control). Each bit of the registers corresponds to one port of the hub. If set to 1, the hub logic is currently waiting for a reply on this port.

0x84: Link Status The current status of a physical hub port is shown. Each bit corresponds to one hub port. If the link on this port is established and able to transmit and receive data, the bit in this register is set.

0x85: Uplink Configuration Configuration of the hub. For each link that is configured to accept requests, the corresponding bit is set.

0x86: Downlink Configuration Configuration of the hub. For each link that is configured to accept replies, the corresponding bit is set.

0x87: Read-out Handler Status Status of the read-out handling logic. The description of the bits of this register can be found in [63].

0x88, 0x89, 0x8B: Time-out The three registers show if a time-out was detected for each hub port and network channel. The registers are set as soon as a time-out is detected and are cleared after reading via slow-control.

0x8C, 0x8D, 0x8F: Hand-shake Error The three register report errors in the hand-shake done in the link layer (IO-Buffers). For each network channel and hub port the corresponding bit is set if data transmission had to be stopped due to a missing acknowledge.

0x90: Transmission Error On ports with transmission error correction, all hub ports that encountered transmission errors are indicated by a set bit in this register.

0xA0, 0xA1, 0xA3: Error Information For each of the three network channels, the status and error information sent by the hub with a reply is stored.

0xA4: Slow Control Error Serious errors on the slow-control channel are summarized for each port of the hub.

0xA5: Board Tracking The register provides the information needed for tracking a boards connection through the network. See section D.2 for details.

0x4000 – 0x400F: Read-out Packet Counter A register for each port of the hub with a counter of network packets (i.e. 8 Byte of payload) received on the read-out channel.

0x4010 – 0x401F: Slow-control Packet Counter One register for each port of the hub contains a counter of network packets received on the slow-control channel.

0x4020 – 0x402F: Error Information Each register corresponds to one port of the network hub and contains a part of the status and error information received on this port with the last transfer. The registers contents are:

Bit 7 – 0 Bit 7 – 0 of the error information on the LVL1 channel

Bit 15 – 8 Bit 23 – 16 of the error information on the LVL1 channel

Bit 23 – 16 Bit 7 – 0 of the error information on the read-out channel

Bit 31 – 24 Bit 23 – 16 of the error information on the read-out channel

0x4030 – 0x43F: Inclusive Busy Counter The total busy time (i.e. the time the hub waits for a reply on this port on the LVL1 channel) is monitored for each hub port. The value is given in microseconds. Counters can be cleared by writing to 0x4030.

Appendix C. Slow Control Registers

0x4040 – 0x404F: Exclusive Busy Counter The exclusive busy time (i.e. the time the hub waits for a reply on this port on the LVL1 channel and no other port is busy as well) is monitored for each hub port. The value is given in microseconds. Counters can be cleared by writing to 0x4040.

0x4060 – 0x406F: Transmission Error Correction The number of retransmission requests received (Bit 23 – 16) and requests sent (Bits 31 – 24) are shown for each port of the network hub. The lower 16 Bit contain information about the status of the link state machines (see [63] for details).

C.3. MDC Configuration and Status Registers

The MDC subsystem features a rich set of status and configuration registers. The basic control signals and values are shown in table C.7. The data reduction and filtering is configured using the settings shown in table C.8. The values of the ADC measuring the most important voltage levels on the OEP are available in registers shown in table C.9, the corresponding voltages for each ADC channel are given in table C.10.

The status of each internal state machine controlling the communication to the Motherboard is available from the registers listed in table C.12. Status information and statistics are available in registers and sent in the read-out data stream for each status trigger event. These registers and data identifiers (compare table 6.2) are listed in table C.11.

The configuration for each TDC and the DAC on the motherboard is shown in table C.6.

Name	short MBO	long MBO
TDC Reg. 0 (Mode 0)	$0xA001 + 2 \times \text{TDC}$	$0xA061 + 2 \times \text{TDC}$
TDC Reg. 1 (Mode 1)	$0xA011 + 2 \times \text{TDC}$	$0xA079 + 2 \times \text{TDC}$
TDC Reg. 2 (Cal. Mask)	$0xA021 + 2 \times \text{TDC}$	$0xA091 + 2 \times \text{TDC}$
TDC Reg. 3 (Inp. Mask)	$0xA031 + 2 \times \text{TDC}$	$0xA0A9 + 2 \times \text{TDC}$
Thresholds	$0xA049 + 2 \times \text{DBO}$	$0xA0CD + 2 \times \text{DBO}$

Table C.6.: The MDC Motherboards can be configured by several registers in each TDC and the on-board DAC. The position in the internal memory depends on the type of motherboard (short or long). The variables DBO and TDC have to be replaced with the corresponding DBO or TDC number, respectively. Note that both are counted from 0. Addresses in the range above 0xA000 not listed must not be written since they contain essential control information.

Appendix C. Slow Control Registers

Reg.	Bit	Description
0x20	22	Executes a begin run trigger to initialize MBO and TDC
0x20	16	Simulates a timing trigger to test the read-out in a setup without an external common stop signal
0x20	15	Reloads the firmware of the FPGA from Flash
0x20	9	Force a reinitialization of the motherboard. It is executed after the next trigger sent by the CTS has been processed
0x20	5	Reset status registers and error counters
0x20	4	Reset persistent error flags
0x20	1	Resets the TDC Read-out, Data Handler and Trigger Handler logic blocks.
0x20	0	Resets the motherboard control logic and disables TDC. A begin run trigger is necessary to re-configure the motherboard.
0x22	31	Enable common stop input
0x22	30	Enable sending debug data
0x22	21	Enable sending dummy data words instead of real TDC data
0x22	20	Select data format: 0: compressed, 1: long data format
0xC0	5 – 4	1: short MBO, 2: long MBO
0xC1	0	Switch on blinking of LED to identify a board
0xC1	10	Enable automatic reconfiguration in case of no token return
0xC1	27 – 16	Number of dummy data words (if enabled)

Table C.7.: The control registers for configuration of the MDC front-end

Register	Bits	Description
0xC0	25 – 16	Maximum number of data words per event
0xC1	8	Suppress sending of a hit 0 without preceding hit 1 from one channel. Note that this also suppresses double or triple hit 0.
0xC1	9	Suppress sending of a hit 1 without subsequent hit 0 from one channel
0xC2	10 – 0	Lower Threshold of TDC data for hit 0. Only data words with a time greater or equal this value are transported.
0xC2	26 – 16	Lower Threshold of TDC data for hit 1. Only data words with a time greater or equal this value are transported.

Table C.8.: MDC Control Registers for Data Filtering

C.3. MDC Configuration and Status Registers

Address	Description
0x8000	Control Register. Bit 0: ADC enable - measurements are performed continuously, 1: perform a single measurement for all voltages, then switch off again, 2: reset the ADC logic, 3: Enable the comparison with thresholds and update the voltage status overview
0x8001	Voltage status overview. Each hex digit of the register value represents one voltage. Values: 0: voltage ok, 1: too low, 2: too high or combinations thereof. Too low / too high flags are stored until ADC reset.
0x8010 – 0x8017	Current voltage level. One register for each channel. Bit 11 – 0 show the current voltage, measured in millivolt.
0x8018 – 0x801F	Threshold Settings. One register for each channel. Bit 11 – 0 show the low threshold, Bit 27 – 16 contain the high threshold in millivolt.
0x8020 – 0x8027	Min/Max values. One register for each channel. Bit 11 – 0 contain the lowest measured value, Bit 27 – 16 show the highest measured value. Register are reset on ADC reset.

Table C.9.: Memory map for MDC OEP voltage monitoring ADCs. The voltages assigned to each channel are shown in table C.10

Channel	Voltage	Nominal Voltage
0	5V input	5.8V
1	5V regulated	5.0V
2	3.6V input	3.8V
3	3.3V regulated	3.35V
4	1.8V input	1.8V
5	1.2V regulated	1.20V
6	3.0V input	3.0V
7	-3.0V input	-3.0V

Table C.10.: The address offset in registers 0x8010 to 0x8027 of the OEP corresponds to the ADC channels used for voltage monitoring. Both 5V channels are connected using a voltage divider, thus reading 2.5V at nominal voltage.

Appendix C. Slow Control Registers

Name	Code	Description
Basic Info.	0x00	Bit 15 – 0: Internal trigger number, Bit 16: Short MBO, Bit 17: Long MBO, Bit 18: CMS active
Token Missing	0x01	Number of missing token
Phys. Triggers	0x02	Number of received triggers
Calib. Triggers	0x03	Number of received calibration triggers
Discarded Hit 1	0x04	Number of discarded hit 1 words from TDC due to threshold setting
Discarded Hit 0	0x05	Number of discarded hit 0 words from TDC due to threshold setting
Discarded Words	0x06	Number of discarded words due to limit per event
Truncated Evt	0x07	Number of truncated events
Single Hit 1	0x08	Number of single, double or triple hit 1
Single Hit 0	0x09	Number of single, double or triple hit 0
Retransmit Req.	0x0A	Bit 11 – 0: Number of retransmit requests sent, Bit 23 – 12: Number of retransmit requests received
Words	0x0B	Number of words given to data handler
Invalid Trg.	0x0C	Bit 15 – 0: Number of invalid triggers received
Multiple Trg.	0x0D	Bit 15 – 0: Number of multiple triggers received
Spikes Trg.	0x0E	Bit 15 – 0: Number of spikes on CMS received
Spurious Trg.	0x0F	Bit 15 – 0: Number of spurious triggers received
Idle Time	0x10	Idle time of the trigger handler state machine in μ s
Init Time	0x11	Time the OEP spent for reinitializations in μ s
Calib Time	0x12	Time the OEP spent with calibration in μ s
Readout Time	0x13	Time while reading data from TDC in μ s
Waiting Time	0x14	Time spent with various small waits in μ s
Dummy Word	0x1E	Dummy data word. Sent in every event when selected by CCR2 Bit 22. Bit 23 – 16: Lower 8 bit of trigger number. Bit 11 – 0: Word counter
Debug Word	0x1F	Debug word. Sent in every event when selected by CCR2 Bit 30. Bit 15 – 0: Trigger number

Table C.11.: MDC Status words are used to transport supplemental information along with event data. All words are identified by a 5 Bit code as listed above. The values are also accessible via slow control in registers 0x9100 to 0x9114.

Address	Description
0x9000	Main control. Bit 2: Common Stop Active
0x9001	Trigger Handler. Bit 3 – 0: Status of the state machine. 0: idle, 1: processing begin run trigger, 2: processing timing trigger, 3: processing calibration trigger, 4: read-out, 5 – 6: releasing LVL1, 7 – 8: reinitialization
0x9002	Data Handler. Bit 3 – 0: Status of the state machine, Bit 21 – 12: counter for data words
0x9003	TDC Readout. Bit 3 – 0: Status of the state machine, Bit 6: DST signal from MBO, Bit 7: AOD signal from MBO, Bit 8: Reserve signal from MBO
0x9005	Begin Run Trigger. Bit 3 – 0: Step of the mode settings sequence, Bit 7 – 4 step of the configuration sequence, Bit 19 – 16 status of the state machine
0x9006	Mode Line Setting. Bit 7 – 0: status of the state machine, Bit 16: GDE, Bit 17: MOD, Bit 18: RES, Bit 19: TOK, Bit 20: WRM, Bit 21: RDM
0x9007	TDC Setup. Bit 7 – 0: status of the state machine

Table C.12.: The MDC endpoints contains registers that reflect the current status of all internal components. The table shows only an excerpt of the most important values. The full list is available in [63].

D. Control and Monitoring Software

The main interface between the DAQ network and all software used for slow-control purposes is provided by the libtrbnet library. This library was developed at TU Munich and is available for different platforms. It can either be run on the Etrax CPU on a TRB and controls the interface to the on-board FPGA. A second version runs on any PC Linux platform and connects to the Pexor PCI-Express card.

The software can either be used locally on the CPU that is connected to the FPGA interface but also contains server that allows access from anywhere within the network. Here, a Remote Procedure Call (RPC) approach is used to set up a server and client system that can execute any kind of TrbNet access [71].

In this chapter, only few tools used by DAQ experts are shown. The general monitoring interface to all DAQ sub-systems is described in 7.3.

D.1. Control Software

TrbCmd TrbCmd is the main tool that provides the possibility to send manually every kind of TrbNet access. In test setup it can be used to send trigger and read-out requests, but mainly is used to issue slow-control accesses. This software is the basis for most high-level DAQ control tools. TrbCmd features several scripting capabilities and an easily parseable output data format for further data analysis [71].

TrbFlash Most DAQ FPGA are equipped with a Flash ROM that is able to initialize the FPGA after power-up with the correct program. This Flash ROM can be rewritten with updated firmware versions using the DAQ network. TrbFlash contains all features necessary for this procedure, including identification of the front-end type and Flash ROM version as well as erasing, programming and verifying the Flash content. For the Pexor card a specialized version, PexorFlash, is available to program the ROM [71].

TrbDHCP One of the first steps during the Start-up process is to assign the network addresses to all nodes. TrbDHCP takes a list of unique id to network address assignments and executes the necessary commands to configure the network [71].

TrbI2C / TrbRichCmd These are specialized tools to issue commands to configure and monitor the RICH front-end electronics. TrbI2C combined with a VHDL block in the ADCM


```

!Register table
# Type # C0 # C1 # C2 # C3 # C4 # C5 #
  1     0xA049 0xA04B 0xA04D 0xA04F
  2     0xA0CD 0xA0CF 0xA0D1 0xA0D3 0xA0D5 0xA0D7

!Value table
# Board # Type # T0 # T1 # T2 # T3 # T4 # T5 #
0x2000  1     0x60 0x58 0x48 0x80
0x2001  2     0x50 0x50 0x38 0x40 0x40 0x48

```

Figure D.1.: Example of a database file used to load values to a set of registers in the front-ends. In the upper part two sets of registers are defined. The lower part contains front-end addresses, a register set selector and the values to be loaded.

board is able to interface to the I2C interface of the pre-amplifier chips. Other features of the front-end cards such as temperature monitoring and ID reading is provided by TrbRichCmd [63].

Startup The start-up script is used to execute all commands necessary for starting the DAQ system. E.g. the main configuration is loaded to all front-ends, data sources are defined and a list of all front-ends available within the network is generated. A more detailed description is given in 7.2.1.

The data to be loaded is defined in database files like the one shown in figure D.1.

D.2. Monitoring Software

GbE Monitor

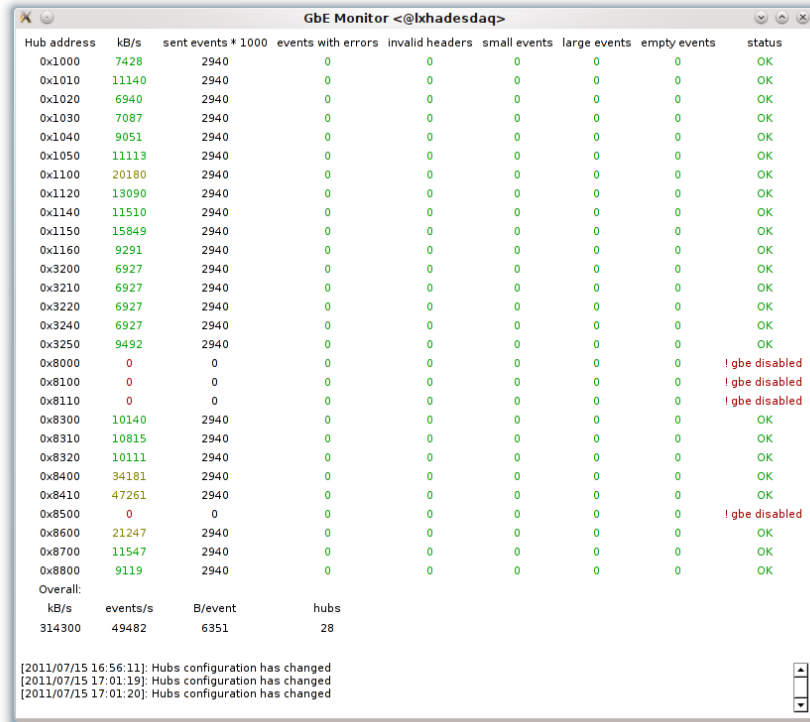
The Gigabit Ethernet Monitor shows a list of all network hubs with GbE interface and displays the amount of data sent and possible errors from these hubs [72]. A screenshot of the software is shown in figure D.2.

CTS Monitor & Control

The CTS Monitor [73] gives an overview of the current configuration of the Central Trigger System and provides the interface to alter all settings. In the same screen the most important statistic values are shown as well. These include scaler values from all inputs and the current trigger rate.

Several configuration sets can be saved and restored. A given setup can be selected to be loaded during the start-up phase of the DAQ network.

Appendix D. Control and Monitoring Software



The screenshot shows a window titled "GbE Monitor <@lxhadesdaq>". It contains a table with the following columns: Hub address, kB/s, sent events * 1000, events with errors, invalid headers, small events, large events, empty events, and status. The table lists 28 network hubs. Most hubs show a status of "OK". Hubs 0x8000, 0x8100, 0x8110, 0x8500, and 0x8800 show a status of "! gbe disabled". At the bottom of the table, there is an "Overall:" section with summary statistics: kB/s: 314300, events/s: 49482, B/event: 6351, hubs: 28. Below the table, there are three log messages: "[2011/07/15 16:56:11]: Hubs configuration has changed", "[2011/07/15 17:01:19]: Hubs configuration has changed", and "[2011/07/15 17:01:20]: Hubs configuration has changed".

Hub address	kB/s	sent events * 1000	events with errors	invalid headers	small events	large events	empty events	status
0x1000	7428	2940	0	0	0	0	0	OK
0x1010	11140	2940	0	0	0	0	0	OK
0x1020	6940	2940	0	0	0	0	0	OK
0x1030	7087	2940	0	0	0	0	0	OK
0x1040	9051	2940	0	0	0	0	0	OK
0x1050	11113	2940	0	0	0	0	0	OK
0x1100	20180	2940	0	0	0	0	0	OK
0x1120	13090	2940	0	0	0	0	0	OK
0x1140	11510	2940	0	0	0	0	0	OK
0x1150	15849	2940	0	0	0	0	0	OK
0x1160	9291	2940	0	0	0	0	0	OK
0x3200	6927	2940	0	0	0	0	0	OK
0x3210	6927	2940	0	0	0	0	0	OK
0x3220	6927	2940	0	0	0	0	0	OK
0x3240	6927	2940	0	0	0	0	0	OK
0x3250	9492	2940	0	0	0	0	0	OK
0x8000	0	0	0	0	0	0	0	! gbe disabled
0x8100	0	0	0	0	0	0	0	! gbe disabled
0x8110	0	0	0	0	0	0	0	! gbe disabled
0x8300	10140	2940	0	0	0	0	0	OK
0x8310	10815	2940	0	0	0	0	0	OK
0x8320	10111	2940	0	0	0	0	0	OK
0x8400	34181	2940	0	0	0	0	0	OK
0x8410	47261	2940	0	0	0	0	0	OK
0x8500	0	0	0	0	0	0	0	! gbe disabled
0x8600	21247	2940	0	0	0	0	0	OK
0x8700	11547	2940	0	0	0	0	0	OK
0x8800	9119	2940	0	0	0	0	0	OK
Overall:								
kB/s	events/s	B/event	hubs					
314300	49482	6351	28					

[2011/07/15 16:56:11]: Hubs configuration has changed
[2011/07/15 17:01:19]: Hubs configuration has changed
[2011/07/15 17:01:20]: Hubs configuration has changed

Figure D.2.: The Gigabit Ethernet monitor shows a list of all network hubs with GbE interface and displays the amount of data sent and a list of all errors encountered.

Hub Monitor

The hub monitor [73] gives an overview on the current status of the DAQ network. For each network hub the number the number of network links available and the number of links connected to another node are shown. Additionally the busy time on the LVL1 trigger channel is given for each link. This allows to easily determine which front-end is limiting the DAQ rate or how big the total dead time of the system is.

The script also monitors the number of boards available in the network and compares them to a list generated during DAQ start-up to detect missing boards.

LogMon / LogMonEB

All changes in the DAQ system and error messages are logged in a central file. This file is visualized by the LogMon program [73]. For better usability, the log file monitor is split in two parts: One shows only errors from the Eventbuilders, the other one covers all other messages. A screenshot of the main log monitor is shown in figure D.4. [73]

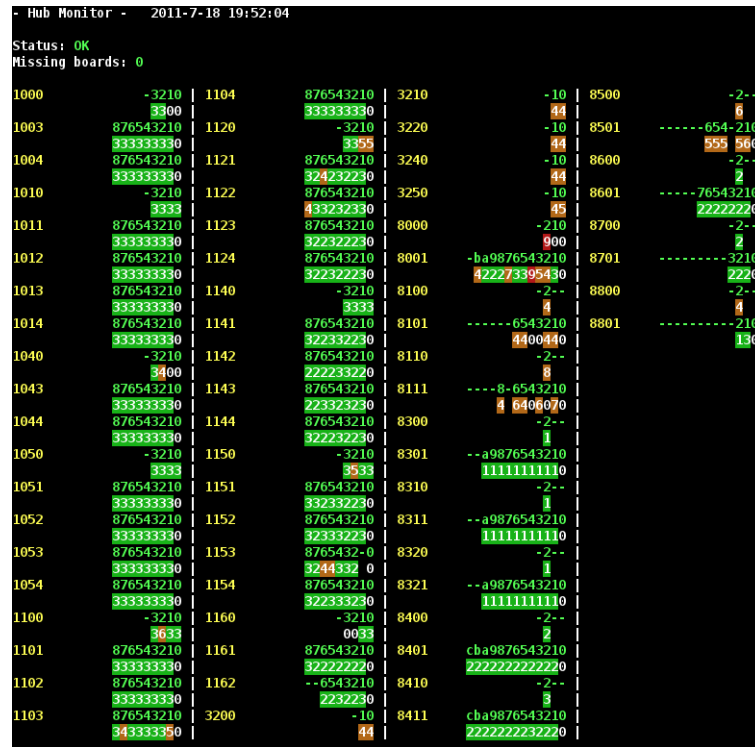


Figure D.3.: The Hub Monitor displays the busy times of each link of each of the network hubs within the system. Active network links are shown as green numbers (corresponding to the input the hub - hub addresses are displayed in yellow), busy times are shown as multiples of 10% busy time (in white, with colored background). The script also monitors the number of boards available in the network and compares them to a list generated during DAQ start-up to detect missing boards.



Figure D.4.: The Logfile Monitor shows a log of the most important actions done and errors found within the DAQ system.

Appendix D. Control and Monitoring Software

```
1 $ ./nettrace.pl 2154
2   Hop   Board   Port
3     0    8000    2
4     1    8001    3
5     2    8100    2
6     3    8101    6
7     4    1050    2
8     5    1053    5
9     6    2154
```

Figure D.5.: Example of a nettrace looking up the path to front-end board 0x2154. The access is routed through six layers of network hubs. In this example, the boards is connected to hub 0x1053, port 5 which in turn is connected to hub 0x1050, port 2 and so on.

NetTrace - Board Tracing

In many situations the exact knowledge about the network path to a given front-end module is vital. To disconnect a board from the network, either physically by removing a cable or logically by changing the network hub configuration, it is necessary to know which port the board is connected to. Vice versa, if a hub reports a problem with the connection on one port, one needs to know which board is connected to this port.

Since the TrbNet setup does not contain inherent topological information, on-line tracing has to be performed. This is implemented in a simple two-step process: First, a slow control access is done to a specific board³³. The network hubs store information about the port from which the transfer was answered in a register which can be read out by a subsequent access.

The result from this access contains all information to reconstruct the full data path through the network. An example is shown in figure D.5. By looping over all connected boards the tree structure of the full network can be deduced.

HadPlot

HadPlot is a versatile tool to visualize any kind of information available from within TrbNet. It consists of a Perl script that issues and evaluates commands in the DAQ network using TrbCmd. After processing, data is fed to gnuplot [74] which displays the data in various different plot formats. Examples are given in section 8.4.

³³This access is required to produce a minor error in the front-end, namely an “unknown command” error which can be caused by issuing a memory read operation on register 0. The reason is the protection from other slow control accesses which might occur inbetween the two steps of the tracking.

The tool has a set of predefined plots built-in. Additionally, the user can define any data source accessible in the DAQ network to be displayed.

The command options are as follows:

```
hadplot [-d delay] [-n samples] [-o downscaling]
        [-g geometry] [-z style] [-c windowtitle]
        [-a address -r register -w regwidth -p regoffset [-t title]]*
        plotname
```

The behavior of the plot window are determined by the delay between two data acquisitions (-d) given in milliseconds. For histogram plots, the number of samples to take between two updates of the plot window (-o) and the number of samples to display (-n) can be defined as well. In general, the update rate should not be above 10 Hz for graphically simple plots and 1 Hz for plots with many data points. The sampling frequency can be up to 200 Hz when using the PCI-e interface.

The geometry of the plot window can be given in standard X format (-g, e.g. 640x480+50+100). Note that this string can also be used to set background and font color. The title of the plot window can be set as well (-c). The style option (-z) is used to select between different plotting styles and labels on the axes: Style 0 shows dots for each value, style 2 and 4 show bargraphs. Style 2 places bars next to each other, style 4 on top of each other if more than one dataset is to be displayed. Style 12 is similar so style 4, but the y-axis values are divided by 1000 to represent seconds or milliseconds instead of microseconds. Adding 1 to each style changes the x-axis labels to network addresses instead of arbitrary numbers. Note that not all styles are available for all plot types.

For generic plots, a set of network address (-a), register address (-r), width of the value (-w) and position in the register (-p) can be given to select arbitrary data sources in the network. All four options are mandatory and can not be left out. The full set can be repeated to define several data sources to be shown in one plot. Currently, four generic plot types are defined:

reg (“register”) shows the content of registers.

regdiff (“register differences”) shows the differences of register values between two samples.

hist (“histogram”) shows a histogram of register values. Here, all values returned by the slow control access are summed up.

histdiff (“histogram differences”) shows a histogram of register values, but only the relative values for each single register is summed and shown.

Often used plots are predefined and directly accessible via their name. Internally, most are based on generic plots but some which require preprocessing of data use individual functions.

Bibliography

- [1] Alexander Schmah. private communication, 2011.
- [2] Berndt Müller. Signatures of the quark-gluon plasma. *Nuclear Physics A*, 544(1–2):95–108, 1992.
- [3] F. Weber and R. Negreiros. QCD in neutron stars and strange stars. *AIP Conf.Proc.*, 1354:13–18, 2011.
- [4] A. Andronic, D. Blaschke, P. Braun-Munzinger, J. Cleymans, K. Fukushima, et al. Hadron Production in Ultra-relativistic Nuclear Collisions: Quarkyonic Matter and a Triple Point in the Phase Diagram of QCD. *Nucl.Phys.*, A837:65–86, 2010.
- [5] J.M. Heuser, M. Deveaux, C. Muntz, and J. Stroth. Requirements for the silicon tracking system of CBM at FAIR. *Nucl.Instrum.Meth.*, A568:258–262, 2006.
- [6] M. Anderson, J. Berkovitz, W. Betts, R. Bossingham, F. Bieser, et al. The Star time projection chamber: A Unique tool for studying high multiplicity events at RHIC. *Nucl.Instrum.Meth.*, A499:659–678, 2003.
- [7] J. Alme, Y. Andres, H. Appelshauser, S. Bablok, N. Bialas, et al. The ALICE TPC, a large 3-dimensional tracking device with fast readout for ultra-high multiplicity events. *Nucl.Instrum.Meth.*, A622:316–367, 2010.
- [8] O. Bourrion, R. Guernane, B. Boyer, J. L. Bouly, and G. Marcotte. Level-1 jet trigger hardware for the alice electromagnetic calorimeter at lhc. *Journal of Instrumentation*, 5(12):C12048, 2010.
- [9] Claudia Borer. Overview of the atlas data acquisition system operating at the tev scale. *IEEE Trans. Nucl. Sci.*, 17th Real Time Conference, 2010.
- [10] Walter F J Müller for the CBM collaboration. The cbm experiment @ fair—new challenges for front-end electronics, data acquisition and trigger systems. *Journal of Physics: Conference Series*, 50(1):371, 2006.
- [11] Martin Purschke. Upgrades for the phenix data acquisition system. *IEEE Trans. Nucl. Sci.*, 17th Real Time Conference, 2010.

- [12] IEEE. Standard 1076 - very high speed integrated circuit description language. Technical report, IEEE, 1987.
- [13] G. Haefeli, A. Bay, A. Gong, H. Gong, M. Muecke, et al. The LHCb DAQ interface board TELL1. *Nucl.Instrum.Meth.*, A560:494–502, 2006.
- [14] B.G. Taylor. Timing distribution at the LHC. *LECC 2002 Proceedings*, pages 63–74, 2002.
- [15] M. Frank, J. Garnier, C. Gaspar, R. Jacobsson, B. Jost, Liu Guoming, and N. Neufeld. The LHCb Eventbuilder: Design, Implementation and Operational Experience. *Nuclear Science, IEEE Transactions on*, 58, Issue: 4:1877 – 1884, 2011.
- [16] The HADES Collaboration. The high-acceptance di-electron spectrometer hades. *European Physics Journal A*, 41:243–277, 2009.
- [17] Jerzy Pietraszko. Diamonds as timing detectors for minimum-ionizing particles: The hades proton-beam monitor and start signal detectors for time of flight measurements. *Nuclear Instruments and Methods in Physics Research A*, A 618:121 – 123, 2010.
- [18] Matthias Sylvester. Funktionsbeschreibung für den hades drift chamber tdc. Technical report, MSC ASIC Design Center, November 1998.
- [19] Wolfgang König. private communication. July 2011.
- [20] Karl Zeitelhack. The hades rich detector. *Nuclear Instruments and Methods in Physics Research A*, 433:201 – 206, 1999.
- [21] Attilio Tarantola. *Dielectron analysis in p+p collisions at 3.5 GeV with the HADES spectrometer: ω -meson line shape and a new electronics readout for the Multi-wire Drift Chambers*. PhD thesis, Goethe-University Frankfurt, 2010.
- [22] Jerzy Pietraszko. The hades pre-shower detector. *Nuclear Instruments and Methods in Physics Research A*, 531:445 – 458, 2004.
- [23] C. Müntz et al. The hades tracking system. *Nucl.Instrum.Meth.*, A535:242–246, 2004.
- [24] Thomas Bretz. Magnetfeldeigenschaften des spektrometers hades. *Diploma Thesis*, 1999.
- [25] Jochen Markert. Estimated data rates. private communication, March 2010.
- [26] Paolo Finocchiaro. The hades time-of-flight wall. *Nuclear Instruments and Methods in Physics Research A*, 492:14–25, 2002.
- [27] Paulo Fonte. The hades rpc inner tof wall. *Nuclear Instruments and Methods in Physics Research A*, 602:687 – 690, 2009.

Bibliography

- [28] Alberto Blanco. In-beam measurements of the hades-tof rpc wall. *Nuclear Instruments and Methods in Physics Research A*, 602:691 – 695, 2009.
- [29] Diego Gonzalez-Diaz. Performances of multi-gap timing rpcs for relativistic ions in the range $z=1-6$. *JINST*, 4:P11007, 2009.
- [30] Erik Lins. *Entwicklung eines Auslese- und Triggersystems zur Leptonenidentifizierung mit dem HADES-Flugzeitdetektor*. PhD thesis, Justus-Liebig-Universität Gießen, 2001.
- [31] The HADES Collaboration. Study of dielectron production in c+c collisions at 1 agev. *Phys.Lett. B*, 663:43 – 48, 2008.
- [32] The HADES Collaboration. Dielectron production in c-12 + c-12 collisions at 2-agev with hades. *Phys. Rev. Lett.*, 98:052302, 2007.
- [33] R.J. Porter and others (DLS Collaboration). Dielectron cross section measurements in nucleus-nucleus reactions at 1.0 agev. *Phys. Rev. Lett.*, 79:1229–1232, 1997.
- [34] The HADES Collaboration. Dielectron production in ar+kcl collisions at 1.76a gev. 2011.
- [35] The HADES Collaboration. Origin of the low-mass electron pair excess in light nucleus-nucleus collisions. *Phys. Lett. B*, 690:118–122, 2010.
- [36] K. Beckert, P. Beller, W. Bourgeois, B. Franczak, B. Franzke, F. Nolden, U. Popp, A. Schwinn, and M. Steck. ESR Operation and Development. *GSI Scientific Report*, 2001, 2001.
- [37] Jerzy Pietraszko. Technical design report for hades-100. investigation of dilepton production at sis100 with hades. 2009.
- [38] The HADES Collaboration. Investigation of baryon rich dense nuclear matter by means of rare probes with hades. 2010.
- [39] Marcin Kajetanowicz. pre-shower daq, private communication.
- [40] Jan Michel. Development of a real-time network protocol for hades and fair experiments. *Diploma Thesis*, 2008.
- [41] *Optoway SPM-8100WG 3.3V / 850 nm / 4.25 Gbps SFP LC Multi-Mode Transceiver*.
- [42] *Firecomms, 650 nm 250 Mbps Fiber Optic Transceiver FDL-300I*.
- [43] *Maxim IC DS18S20 1-Wire Digital Thermometer*.
- [44] Ingo Fröhlich. A general purpose trigger and readout board for hades and fair-experiments. *IEEE Trans.Nucl.Sci.*, 55:59–66, 2008.

- [45] *Xilinx Virtex-4 FPGA Family Data Sheet*.
- [46] Texas Instruments. *TLK 2501 - 1.5 to 2.5 Gbps Transceiver*. <http://www.ti.com/lit/gpn/tlk2501>, 2003.
- [47] J. Christiansen. *HPTDC*. Digital Microelectronics Group, CERN, 2002.
- [48] Axis Communications. *Axis Etrax FS System-on-Chip*.
- [49] Lattice Semiconductor. *Lattice Semiconductor, HB1003 LatticeECP2/M Family Handbook*.
- [50] Texas Instruments *ADS5271 8-Channel, 12-Bit, 50MSPS Analog-to-Digital Converter with Serial LVDS Interface*.
- [51] Lattice Semiconductor, *DS1004 LatticeSC/M Family Datasheet*.
- [52] Hans Zimmermann. Osi reference model — the iso model of architecture for open systems interconnection. *IEEE Transactions on Communications*, 28:425 – 432, 1980.
- [53] Michael Böhmer. Rich daq system, private communication. 2010.
- [54] Marek Palka. Φ Meson Production in pp Reactions at 3.5 GeV with the HADES Detector. PhD thesis, Jagiellonian University Krakow, 2011.
- [55] Marek Palka. private communication. May 2011.
- [56] Alberto Blanco. private communication. May 2010.
- [57] Mathias Münch. Hades raw data format, 2002.
- [58] Grzegorz Korcyl. Implementacja gigabitowego ethernetu na układach fpga firmy lattice dla eksperymentu hades. Master's thesis, Jagiellonian University Krakow, 2010.
- [59] Sergey Yurevich. The hades event building system. *GSI Scientific Report*, 2010.
- [60] F. M. Newcomer. A fast low power, amplifier-shaper-discriminator for high rate straw tracking systems. *IEEE Trans. Nucl. Sci.*, 40:630, 1993.
- [61] Christian Sturm. Internal documentation, March 2004.
- [62] Jörn Wüstenfeld. *Auslese und Qualitätskontrolle der HADES - Driftkammern*. PhD thesis, Goethe-Universität Frankfurt, 2005.
- [63] Jan Michel, Michael Böhmer, Grzegorz Korcyl, and Marek Palka. *A Users Guide to the HADES DAQ Network*, 2012.

Bibliography

- [64] *Experimental Physics and Industrial Control System*, <http://www.aps.anl.gov/epics/about.php>.
- [65] World Wide Web Consortium. *XHTML 1.1*, <http://www.w3.org/TR/2010/REC-xhtml11-20101123/>.
- [66] *Lattice Semiconductor, Technical Note TN1124 LatticeECP2M SERDES/PCS Usage Guide, 2010.*
- [67] Alberto Blanco et al. Rpc-pet: A new very high resolution pet technology. *IEEE Trans. Nucl. Sci.*, 53:2489 – 2494, 2006.
- [68] Michael Deveaux. Design considerations for the micro vertex detector of the compressed baryonic matter experiment. *Proceedings of Science*, 2008.
- [69] R. Turchetta, J.D. Berst, B. Casadei, G. Claus, C. Colledani, et al. A monolithic active pixel sensor for charged particle tracking and imaging using standard VLSI CMOS technology. *Nucl.Instrum.Meth.*, A458:677–689, 2001.
- [70] Eugen Bayer. A high-resolution 48-channel time-to-digital converter (tdc) implemented in a field programmable gate array (fpga). *IEEE Trans. Nucl. Sci.*, 99, 2011.
- [71] Ludwig Maier. private communication.
- [72] Grzegorz Korcyl. private communication.
- [73] Sergey Yurevich. private communication. May 2011.
- [74] *Gnuplot graphing tool* - <http://www.gnuplot.info/>.