Das Datenerfassungssystem für das Crystal-Barrel/TAPS-Experiment an ELSA

Dissertation
zur
Erlangung des Doktorgrades (Dr. rer. nat.)
der
Mathematisch-Naturwissenschaftlichen Fakultät
der
Rheinischen Friedrich-Wilhelms-Universität Bonn

von
Philipp Hoffmeister
aus
Essen

Bonn, Dezember 2018

Dieser Forschungsbericht wurde als Dissertation von der
Mathematisch-Naturwissenschaftlichen Fakultät der Universität Bonn angenommen und
ist auf dem Hochschulschriftenserver der ULB Bonn
http://hss.ulb.uni-bonn.de/diss_online elektronisch publiziert.

1. Gutachter:	Prof. Dr. R. Beck
2. Gutachter:	Prof. Dr. K. Brinkmann
Tag der Promotion:	26.03.2019
Erscheinungsjahr:	2019

Zusammenfassung

Die Hadronenspektroskopie untersucht die Kräfte zwischen den Quarks in Hadronen. Dazu werden die Anregungsspektren der Hadronen untersucht und mit Modellen, wie zum Beispiel Konstituenten-Quarkmodellen, oder Gitter-QCD Rechnungen verglichen. Beim Vergleich zeigen sich in den Messungen fehlende Resonanzen bei höheren Energien. Die Suche nach weiteren Resonanzen, sowie die genauere Identifikation der physikalischen Eigenschaften der bisher gefundenen Resonanzen ist die Hauptaufgabe der Hadronenspektroskopie. Schwierigkeiten bereiten dabei die breiten und überlappenden Resonanzen. Dadurch können Resonanzen mit kleiner Kopplung nicht direkt erkannt und gemessen werden.

Die Anregungen der Nukleonen können durch Streuexperimente mit Strahlteilchen (Sonden) verschiedener Art erfolgen. Dies können stark oder elektromagnetisch wechselwirkende Teilchen sein. Im Falle der starken Wechselwirkung können zum Beispiel π-Mesonen oder Kaonen benutzt werden. Im Falle der elektromagnetischen Wechselwirkung kann zum Beispiel mit Photonen oder Elektronen gestreut werden.

Das Crystal-Barrel/TAPS-Experiment am ELSA-Teilchenbeschleuniger in Bonn hat sich auf die Messung der Anregungsspektren von Protonen und Neutronen spezialisiert, wobei in diesem Experiment die Anregung durch Beschuss mit reellen Photonen erfolgt. Zur Bestimmung von kleinen Resonanzbeiträgen ist es dabei notwendig Polarisationsobservablen zu messen. Dazu werden polarisierten Nukleonen und Photonen benutzt oder das rückgestreute Nukleon gemessen. Um diese Polarisationsmessungen effektiv durchführen zu können, wurden die Detektoren des Crystal-Barrel/TAPS-Experiments umgebaut, erweitert und in einer andere Experimenthalle aufgebaut. Dadurch wurde es notwendig, eine neue Datenerfassung zu entwickeln, welche die Daten des umgebauten Experiments aufnehmen und speichern kann. Im Rahmen dieser Arbeit wurde diese Datenerfassung entwickelt und getestet. Die Datenakquisition wurde dabei in der Programmiersprache C++ implementiert, wobei die Boost-Erweiterung benutzt wurde. Die einzelnen Komponenten der Datenakquisition kommunizieren über das TCP/IP-Protokoll. Es wird im Rahmen dieser Arbeit gezeigt, dass die im Experiment entstehenden Daten gespeichert werden können, ohne dass dadurch eine zusätzliche Totzeit entsteht. Dabei wurde auf Zukunftssicherheit geachtet, so dass das Crystal-Barrel/TAPS-Experiment um weitere Komponenten erweitert werden kann, ohne eine signifikante Totzeit zu

erzeugen. Zusätzlich wurde im Rahmen dieser Arbeit ein System entwickelt, mit dem die Integrität der Daten überprüft wird. Dadurch wird sichergestellt, dass die vom Experiment gemessenen Ereignisse korrekt zusammengesetzt werden. Außerdem wurde ein neues Trigger-Modul entwickelt, mit dem charakteristische Signale der Detektoren auf gewünschte Signaturen untersucht werden, um die Auslese und Speicherung der Daten zu starten.

Mit der neuen Datenerfassung war es möglich, in zahlreichen Messperioden Daten zu speichern, aus denen dann in unterschiedlichen Reaktionskanälen Wirkungsquerschnitte und Polarisationsobservablen bestimmt wurden. Die Daten, die mit der im Rahmen dieser Arbeit entwickelte Datenakquisition gespeichert wurden, bilden damit die Basis für zahlreiche Doktorarbeiten und Veröffentlichungen. Besonders zu erwähnen sind hier die folgenden Veröffentlichungen:

- A. Thiel u. a., *Well-established nucleon resonances revisited by double-polarization measurements*, Phys. Rev. Lett. **109** (2012) 102001, arXiv: 1207.2686 [nucl-ex]
- R. Milner u. a., *The OLYMPUS Experiment*, Nucl. Instrum. Meth. **A741** (2014) 1–17, arXiv: 1312.1730 [physics.ins-det]
- M. Gottschall u. a., *First measurement of the helicity asymmetry for* $\gamma p \to p\pi^0$ *in the resonance region*, Phys. Rev. Lett. **112** (2014) 012003, arXiv: 1312.2187 [nucl-ex]
- J. Hartmann u. a., *The polarization observables* T *,* P *, and* H *and their impact on* $\gamma p \to p\pi^0$ *multipoles*, Phys. Lett. **B748** (2015) 212–220, arXiv: 1506.06226 [nucl-ex]
- B. S. Henderson u. a., *Hard Two-Photon Contribution to Elastic Lepton-Proton Scattering: Determined by the OLYMPUS Experiment*, Phys. Rev. Lett. **118**.9 (2017) 092501, arXiv: 1611.04685 [nucl-ex]

Abstract

Hadron spectroscopy analyses the forces between the quarks within hadrons. For this purpose, the excitation spectra of hadrons are probed, analysed and compared to Constituent Quark Models or to Lattice QCD calculations. The comparison to measurements shows missing resonances at higher energies. Therefore, the search for new resonances and the improved identification of the physical properties of the already established resonances is the main goal of hadron spectroscopy. However, difficulties arise due to the broad and overlapping resonances. This means that resonances with small coupling cannot be directly detected and measured with ease.

The excitation of nucleons can be achieved via scattering experiments using beam particles (probes) of different nature. The latter can be either strongly or electromagnetically interacting particles. In the case of strongly interacting particles, the experiments can use for example pion or kaon beams. Experiments with electromagnetically interacting particles use for example photons or electrons.

The Crystal-Barrel/TAPS experiment at the ELSA accelerator in Bonn specializes on measuring the excitation spectra of protons and neutrons while using real photons as impinging particles. In order to obtain sensitivity to small resonance contributions, it is necessary to measure polarization observables by using polarized nucleons and photons and by measuring the polarization of recoil nucleons. To enable effective polarization measurements, the detectors of the Crystal-Barrel/TAPS experiment were modified, expanded and rebuilt at a different experimental area. This required a new data acquisition, which had to be designed and developed to read out and save the data of the modified experiment. In the scope of this work a data acquisition system was developed and tested. It was implemented using the programming language C++ with the boost-extension. The components of the data acquisition communicate to each other via the TCP/IP protocol. In this work it is demonstrated, that the data which is measured with the experimental setup can be stored without significant additional dead time. To be safe for future expansions of the experiment, a main goal was to also ensure easy expandability and enough margins in the data transport. An additional component of this work was the development of a system which assures the integrity of the data before storing it. This ensures that the data packages measured by different parts of the experiment are assembled correctly. A new trigger module was also developed in

the course of this work. The trigger module checks for given signatures in characteristic signals sent by the detectors. If these selected structures are found, the readout of the detectors is started.

The new data acquisiton made it possible to save data in numerous data taking periods. From this data, different analyses were conducted to obtain cross sections and polarization observables for multiple reaction channels. The data acquisition, which was developed in the course of this work, thereby provided the basis for the release of multiple dissertations and publications. The following publications deserve special mention here:

- A. Thiel u. a., *Well-established nucleon resonances revisited by double-polarization measurements*, Phys. Rev. Lett. **109** (2012) 102001, arXiv: 1207.2686 [nucl-ex]
- R. Milner u. a., *The OLYMPUS Experiment*, Nucl. Instrum. Meth. **A741** (2014) 1–17, arXiv: 1312.1730 [physics.ins-det]
- M. Gottschall u. a., *First measurement of the helicity asymmetry for* $\gamma p \to p\pi^0$ *in the resonance region*, Phys. Rev. Lett. **112** (2014) 012003, arXiv: 1312.2187 [nucl-ex]
- J. Hartmann u. a., *The polarization observables* T , P , *and* H *and their impact on* $\gamma p \to p\pi^0$ *multipoles*, Phys. Lett. **B748** (2015) 212–220, arXiv: 1506.06226 [nucl-ex]
- B. S. Henderson u. a., *Hard Two-Photon Contribution to Elastic Lepton-Proton Scattering: Determined by the OLYMPUS Experiment*, Phys. Rev. Lett. **118**.9 (2017) 092501, arXiv: 1611.04685 [nucl-ex]

Inhaltsverzeichnis

1. Einleitung

In der Atomphysik wird der Aufbau der Atome und die Bindung der Elektronen an den Atomkern durch Betrachtung von Absorptions- und Emissionsspektren der Atome untersucht. Diese Anregung kann zum Beispiel durch Erwärmung oder auch durch beschleunigte Elektronen erfolgen (z. B. beim Frank-Hertz Versuch). Durch die Abregung von angeregten Zuständen endlicher Lebensdauer in energetisch tiefer liegende Zustände entsteht für jedes Material ein charakteristisches Emissionsspektrum aus den abgestrahlten Photonen (siehe zum Beispiel Abbildung 1.1). Alternativ kann auch weißes Licht durch ein Material hindurch gesendet werden. Es kann dann gemessen werden, welche spektralen Teile des Lichts absorbiert werden. Daraus entsteht ein Absorptionsspektrum. Anhand dieser Emissions- und Absorptionsspektren können Schlüsse über den Aufbau der Atome gezogen werden, sowie auch auf die Kräfte zwischen dem Atomkern und der Elektronenhülle. Letztendlich führten diese Messungen in Verbindung mit anderen Experimenten zur Entwicklung und einem tieferen Verständnis der Quantenelektrodynamik.

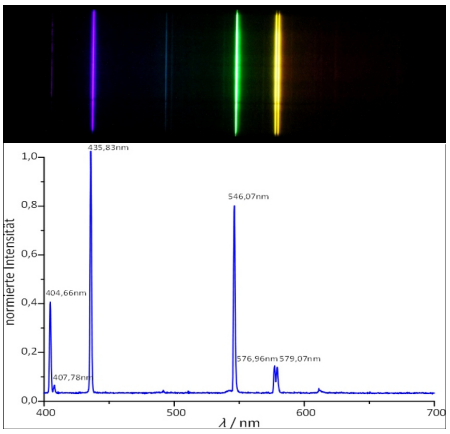


Abbildung 1.1.: **Emissionsspektrum eine Quecksilberdampflampe** [She11]**.** Die Quecksilberdampflampe sendet bei definierten Energien Licht aus. Da die Linienbreite meist deutlich kleiner ist als der Abstand der Linien, lassen sich die Linien leicht voneinander trennen. Nur die gelben Linien liegen dicht beieinander, lassen sich jedoch trotzdem noch trennen.

Um den Aufbau von und die intern herrschenden Kräfte in Nukleonen zu erkunden, kann ein analoges Verfahren angewandt werden. In der Atomspektroskopie wird das

gebundene System aus Kern und Elektronen angeregt. In der Hadronspektroskopie wird das Bindungssystem der Quarks in den Protonen und Neutronen angeregt. Während in der Atomspektroskopie die Anregungsenergien im optischen Bereich von einigen eV liegen, muss die Energie zur Untersuchung der kleineren Strukturen in den Nukleonen erhöht werden und liegt meist im Bereich von mehreren hundert MeV bis zu einigen GeV. Um diese Energien zu erreichen, ist es notwendig Teilchenbeschleuniger zu benutzen bei denen Teilchen wie zum Beispiel Elektronen beschleunigt und dann an den Nukleonen gestreut werden, um letztere energetisch anzuregen.

In der Atomspektroskopie entstehen bei der Abregung der Atome scharfe, voneinander getrennte Linien (siehe Abbildung 1.1). Dies ist in der Hadronspektroskopie jedoch nicht der Fall. Die Breite Γ der Linien hängt durch die Heisenbergsche Unschärferelation direkt mit der Lebensdauer τ der angeregten Zustände zusammen ($\Gamma = \hbar/\tau$) [Pov+06]. Bei der Hadronspektroskopie ist die Lebensdauer der Zustände aufgrund der höheren Kopplungskonstante deutlich kleiner, daher besitzen die Zustände eine höhere Breite von bis zu mehreren Hundert MeV [Pat+16]. Damit überlappen die einzelnen Anregungszustände in Resonanzbereichen [Wun+17] und sind nur schwer voneinander zu unterscheiden.

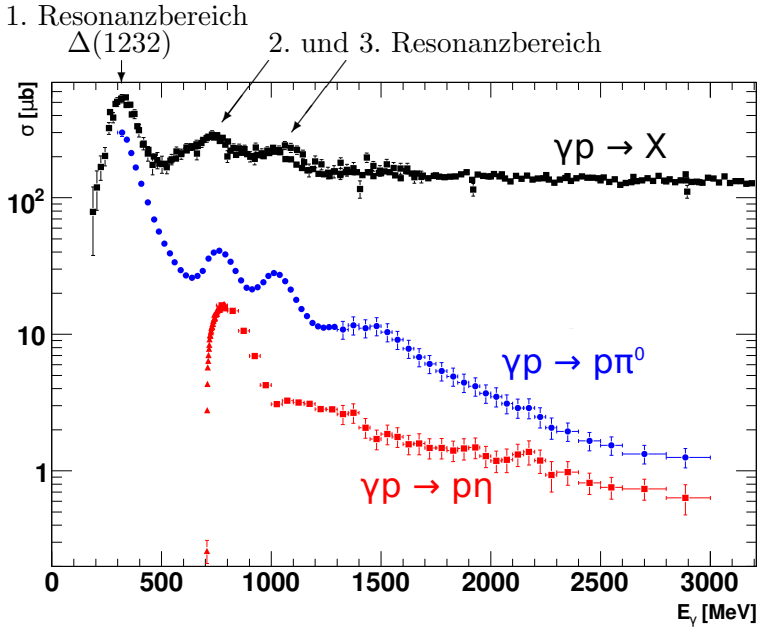


Abbildung 1.2.: **Wirkungsquerschnitte in der Photoproduktion am Proton.** Aufgetragen ist der totale Absorptionsquerschnitt für die Photoproduktion am Proton [Nak+10] als Funktion der Photonenergie in schwarz. Zusätzlich dazu die totalen Wirkungsquerschnitte der Photoproduktion von π^0-Mesonen [Pee+07] (blau) und η-Mesonen [Bar04; Kru+95] (rot).

In Abbildung 1.2 ist der totale Absorptionsquerschnitt der Streuung von Photonen am Proton gezeigt. Dort sind vor allem bei niedrigen Energien die Resonanzbereiche erkennbar, jedoch keine scharf getrennten Linien wie in der Atomspektroskopie. Eine Möglichkeit dies zu verbessern besteht darin anstatt den totalen Wirkungsquerschnitt zu betrachten, nur Reaktionen zu untersuchen bei denen bestimmte Endzustände auftreten, wie zum Beispiel $\gamma p \to p\pi^0$ oder $\gamma p \to p\eta$.

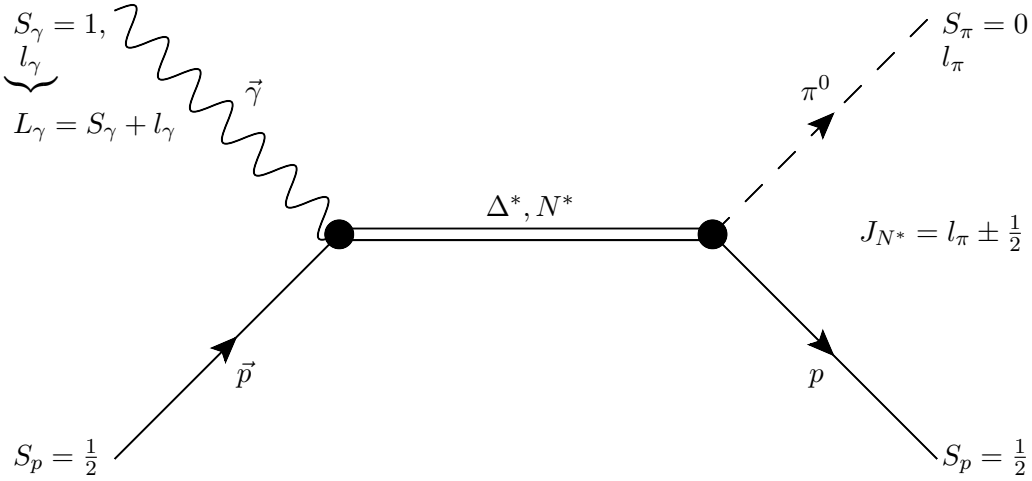


Abbildung 1.3.: **Schematische Darstellung der Reaktion $\gamma p \to p\pi^0$.** Im Anfangszustand reagiert ein Photon mit einem Proton. Dabei entsteht ein Anregungszustand N^* oder Δ^*. Dieser zerfällt dann in ein Proton und ein neutrales Pion (π^0). Bild modifiziert aus [Thi12].

Als Beispiel ist in Abbildung 1.3 die Reaktion $\gamma p \to p\pi^0$ gezeigt. Das Photon regt dabei das Proton an, wobei der angeregte Zustand als Resonanz bezeichnet wird. Dieser angeregte Zustand ist jedoch nicht stabil und kann in ein Proton und ein π^0 zerfallen. Eine andere Reaktion wäre $\gamma p \to p\eta$. Für diese beiden Reaktionen sind die Wirkungsquerschnitte ebenfalls in Abbildung 1.2 aufgetragen. Es lassen sich dabei mehr und andere Strukturen als beim kompletten totalen Wirkungsquerschnitt erkennen. Bei der Reaktion $\gamma p \to p\eta$ ist zum Beispiel klar zu erkennen, dass die erste Struktur des totalen Wirkungsquerschnitts fehlt. Dies liegt daran, dass die Energie dieser Resonanz unterhalb der kinematischen Produktionsschwelle der Reaktion $\gamma p \to p\eta$ liegt. Zusätzlich sind im Absorptionsquerschnitt der Reaktion $\gamma p \to p\eta$ keine Δ-Resonanzen (Isospin $I = 3/2$) enthalten, da diese aufgrund der Isospinerhaltung nicht in den Endzustand $p\eta$ zerfallen können ($I_p = 1/2, I_\eta = 0$)[Pat+16]. Durch den Vergleich unterschiedlicher Reaktionen können also weitere Informationen über die beitragenden Resonanzen gewonnen werden. Die durch die totalen sowie differentiellen Wirkungsquerschnitte erhaltenen Informationen reichen aber nicht aus, um alle beitragenden Resonanzen zu erkennen. Abbildung 1.4

zeigt zur Verdeutlichung für die Reaktion $\gamma p \to p\pi^0$ den totalen Wirkungsquerschnitt mit Breit-Wigner Kurven von möglichen Resonanzen. Die Massen und Verzweigungsverhältnisse der angepassten Resonanzen wurden dabei entnommen aus [Ber+12]. Es sind viele teilweise sehr schwach beitragende Resonanzen erkennbar, welche sich in sogenannten Resonanzgebieten zu den erkennbaren Strukturen überlagern. Durch die Messung des totalen Wirkungsquerschnitts allein können diese schwach beitragenden Resonanzen nicht identifiziert werden. Nur die dominanten Resonanzen lassen sich ermitteln. Zusätzliche Informationen über die schwach beitragenden Resonanzen können gewonnen werden, wenn Messungen mit polarisiertem Strahl und polarisiertem Target durchgeführt werden. Außerdem kann noch die Polarisation der auslaufenden Teilchen gemessen werden. Die so zugänglich gemachten Polarisationsobservablen werden im nächsten Kapitel beschrieben.

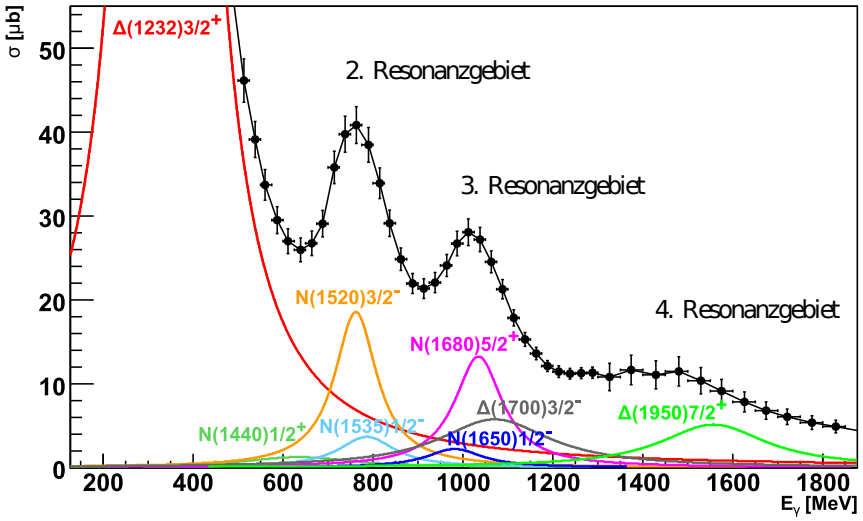
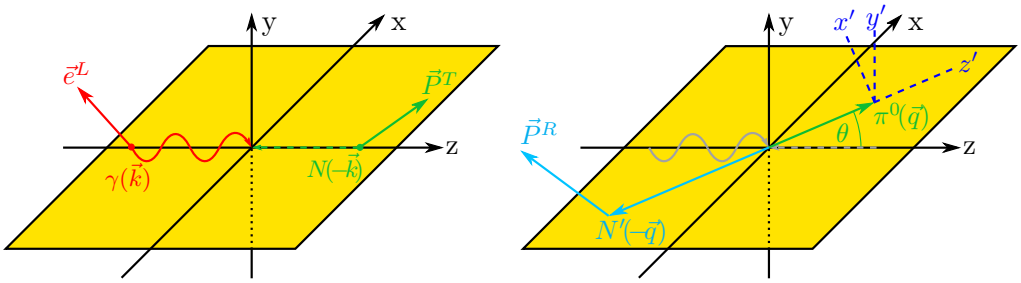


Abbildung 1.4.: **Totale Wirkungsquerschnitt mit Breit-Wigner Verteilungen der Resonanzen in $\gamma p \to p\pi^0$.** Für die möglichen Resonanzen in der Reaktion $\gamma p \to p\pi^0$ wurden Breit-Wigner Verteilungen mit den gemessenen Massen und Verzweigungsverhältnisse aus [Ber+12] erstellt. Bild aus [Got13].

1.1. Polarisationsobservablen



(a) Anfangszustand (b) Endzustand

Abbildung 1.5.: **Koordinatensystem der Reaktion $\gamma N \to N'\pi^0$.** Auf der linken Seite ist der Anfangszustand im Schwerpunktssystem dargestellt. Dabei trifft das polarisierte Photon γ auf ein polarisiertes Nukleon N. $\vec{k}$ ist dabei der Impuls des einlaufenden Photons. Auf der rechten Seite ist der Endzustand dargestellt. Es entstehen ein Nukleon N' und ein π^0. $\vec{q}$ ist hier der Impuls des auslaufenden Mesons.

Bei unpolarisierten Streuexperimenten können lediglich totale und differentielle Wirkungsquerschnitte gemessen werden. Besteht die Möglichkeit die an der Reaktion beteiligten Teilchen zu polarisieren, lassen sich weitere Observablen messen. Diese werden Polarisationsobservablen genannt [San+11]. Bei der Beispielreaktion $\gamma p \to p\pi^0$ können das einlaufende Photon und das Proton polarisiert werden. Zusätzlich kann die Polarisation des Protons im Endzustand gemessen werden. Das Photon kann dabei zirkular oder linear polarisiert werden. Das Proton kann in allen drei Raumrichtungen (x, y, z) polarisiert werden. Bei der Messung der Rückstoßpolarisation des auslaufenden Protons können ebenfalls alle drei Raumrichtungen (x', y', z'), hier notiert im "gestrichenen" Koordinatensystem (siehe Abbildung 1.5), gemessen werden. In Tabelle 1.1 sind alle Größen notiert, welche aus den möglichen Kombinationen der Polarisationsarten der Reaktionsteilchen erhalten werden können und unabhängig voneinander sind. Für den einfachen Fall ohne Rückstoßpolarisation lässt sich der Wirkungsquerschnitt nach [BDS75] aufschreiben als:

$$\begin{aligned}\left.\frac{d\sigma}{d\Omega}\right|_{\text{polarisiert}} = \left.\frac{d\sigma}{d\Omega}\right|_{\text{unpolarisiert}} \{1 &- P_T \Sigma \cos(2\phi) \\ &+ P_x[P_T H \sin(2\phi) + P_\odot F] - P_y[-T + P_T P \cos(2\phi)] \\ &- P_z[-P_T G \sin(2\phi) + P_\odot E]\}\,.\end{aligned} \tag{1.1}$$

Dabei sind P_x, P_y, P_z die Polarisationsgrade des Targets in der jeweiligen Richtung. P_T

		Target			Rückstoß			Target + Rückstoß			
Strahl-	-	-	-	-	x'	y'	z'	x'	x'	z'	z'
polarisation	-	x	y	z	-	-	-	x	z	x	z
unpolarisiert	σ	-	T	-	-	P	-	$T_{x'}$	$L_{x'}$	$T_{z'}$	$L_{z'}$
linear pol.	$-\Sigma$	H	$-P$	G	$O_{x'}$	$-T$	$O_{z'}$	-	-	-	-
zirkular pol.	-	F	-	$-E$	$C_{x'}$	-	$C_{z'}$	-	-	-	-

Tabelle 1.1.: **Polarisationsobservablen der pseudoskalaren Mesonphotoproduktion.** Aufgetragen sind die voneinander unabhängigen Kombinationen von Polarisationsarten [San+11].

ist der Grad der linearen Polarisation und $P_\odot$ der Grad der zirkularen Polarisation des Photonstrahls. In den Summanden ist eine Kombination von Polarisationsgraden und die entsprechende Polarisationsobservable zu finden. Um aus den Polarisationsobservablen Informationen über die beitragenden Resonanzen zu erhalten, ist es zunächst notwendig die Kinematik und den modellunabhängigen Amplitudenformalismus der Reaktion genauer anzuschauen.

1.2. Kinematik und modellunabhängiger Amplituden-Formalismus für die Photoproduktion

Die Reaktionen, welche in der Photoproduktion von einem pseudoskalaren Meson auftreten, haben allgemein die Form:

$$\gamma N \to N^*,\Delta^* \to BM\,. \tag{1.2}$$

Dabei regt ein einlaufendes Photon (γ) ein Nukleon (N) an. Der Anregungszustand wird mit N^* ($I = 1/2$) oder Δ^* ($I = 3/2$) bezeichnet. Beim Zerfall in den Grundzustand entsteht neben einem Rückstoß-Baryon (B) noch ein Meson (M). Vereinfacht lässt sich die Reaktion darstellen durch $\gamma N \to BM$. Dies ist eine Reaktion mit zwei Teilchen im Anfangs- und Endzustand. Ein Beispiel für eine solche Reaktion ist $\gamma p \to p\pi^0$. Die voll relativistische T-Matrix[1], mit der eine solche Reaktionen beschrieben werden kann, kann durch sogenannte invariante Amplituden unabhängig vom Bezugssystem parametrisiert werden. Nach der Transformation ins Schwerpunktssystem reduziert sich diese T-Matrix auf ein Matrixelement zwischen Spin-1/2 Eigenzuständen. Diese Eigenzustände charakterisieren die Spin-Zustände des Nukleons im Anfangszustand (m_a),

[1] Kurzform für transition matrix - Engl. für Übergangsmatrix.

beziehungsweise des Rückstoß-Baryons im Endzustand (m_e) [Che+57]:

$$\frac{d\sigma}{d\Omega} = \frac{q}{k}| < m_e|\mathcal{F}|m_a > |^2 \,. \tag{1.3}$$

Dabei beschreibt k im Schwerpunktssystem den Betrag des Impulses des einlaufenden Photons. Der Betrag des auslaufenden Mesons im Schwerpunkssystem ist bezeichnet mit q. Das Matrixelement $\mathcal{F}$ beschreibt die Dynamik der Reaktion. Dies ist die komplexe Streuamplitude. Die Streuamplitude lässt sich unter anderem nach den Impuls- und Polarisationsanteilen in vier Komponenten mit vier CGLN-Amplituden F_i aufteilen [Che+57]:

$$F_{CGLN} = i(\vec{\sigma}\cdot\vec{\epsilon})F_1 + \frac{(\vec{\sigma}\cdot\vec{q})\left[\vec{\sigma}\cdot(\vec{k}\times\vec{\epsilon})\right]}{qk}F_2 + i\frac{(\vec{\sigma}\cdot\vec{k})(\vec{q}\cdot\vec{\epsilon})}{qk}F_3 + i\frac{(\vec{\sigma}\cdot\vec{q})(\vec{q}\cdot\vec{\epsilon})}{q^2}F_4 \,. \tag{1.4}$$

Dabei sind $\vec{\sigma}$ die Pauli-Spinmatrizen, $\vec{\epsilon}$ der Polarisationsvektor des einlaufenden Photons, $\vec{k}$ der Impulsvektor des einlaufenden Photons und $\vec{q}$ der Impulsvektor des auslaufenden Nukleons.

Wenn der nach Polarisationsobservablen aufgeteilte Wirkungsquerschnitt aus Formel 1.1 hinzugenommen wird, lassen sich die Polarisationsobservablen durch die CGLN-Amplituden ausdrücken [San+09]. Beispielhaft sind hier die Abhängigkeiten der Observablen für die Strahl- und Targetpolarisation angegeben:

$$E = \frac{q}{k}\mathrm{Re}[|F_1|^2 + |F_2|^2 - 2\cos(\theta)F_1^*F_2 + \sin^2(\theta)(F_2^*F_3 + F_1^*F_4)] \,, \tag{1.5}$$

$$G = \frac{q}{k}\sin^2(\theta)\mathrm{Im}[F_2^*F_3 + F_1^*F_4] \,, \tag{1.6}$$

$$F = \frac{q}{k}\sin(\theta)\mathrm{Re}[F_1^*F_3 - F_2^*F_4 - \cos(\theta)(F_2^*F_3 - F_1^*F_4)] \,, \tag{1.7}$$

$$H = -\frac{q}{k}\sin(\theta)\mathrm{Im}[2F_1^*F_2 + F_1^*F_3 - F_2^*F_4 + \cos(\theta)(F_1^*F_4 - F_2^*F_3)] \,. \tag{1.8}$$

Dabei ist θ der Winkel zwischen $\vec{k}$ und $\vec{q}$ im Schwerpunktssystem. Alle Polarisationsobservablen sind dabei von allen vier Amplituden (F_1 bis F_4) abhängig. Um die vollen Amplituden F_i bis auf eine energie- und winkelabhängige globale Phase eindeutig aus den Polarisationsobservablen gewinnen zu können, ist es theoretisch nötig mindestens 8 wohl gewählte Größen zu messen. Nach [CT97] ist dabei die Messung von Observablen mit Strahl- & Rückstoßpolarisation oder Target- & Rückstoßpolarisation zwingend erforderlich.

1.3. Multipolzerlegung

Um aus den CGLN-Amplituden Informationen über die Resonanzen zu erhalten, müssen die Amplituden zunächst zerlegt werden. Dabei wird ausgenutzt, dass der Drehimpuls bei der Reaktion erhalten ist und damit sowohl die Teilchen im Endzustand, als auch der angeregte Zustand den gleichen Drehimpuls besitzen müssen. Teilt man die Streuamplituden nach den verschiedenen Drehimpulszuständen auf, so treten Resonanzen nur in bestimmten Partialwellen auf. Dadurch können die Streuamplituden nach der Gesamtdrehimpulsquantenzahl ℓ entwickelt werden. Zusätzlich werden noch die winkelabhängigen Teile in Legendre-Polynome $P_\ell(\cos(\theta))$ abgespalten [Che+57]:

$$
\begin{aligned}
F_1(W,\theta) &= \sum_{\ell=0}^{\infty} \Big\{ [\ell M_{\ell+}(W) + E_{\ell+}(W)] P'_{\ell+1}(x) + [(\ell+1) M_{\ell-}(W) + E_{\ell-}(W)] P'_{\ell-1}(x) \Big\} , \\
F_2(W,\theta) &= \sum_{\ell=1}^{\infty} [(\ell+1) M_{\ell+}(W) + \ell M_{\ell-}(W)] P'_{\ell}(x) , \\
F_3(W,\theta) &= \sum_{\ell=1}^{\infty} \Big\{ [E_{\ell+}(W) - M_{\ell+}(W)] P''_{\ell+1}(x) + [E_{\ell-}(W) + M_{\ell-}(W)] P''_{\ell-1}(x) \Big\} , \\
F_4(W,\theta) &= \sum_{\ell=2}^{\infty} [M_{\ell+}(W) - E_{\ell+}(W) - M_{\ell-}(W) - E_{\ell-}(W)] P''_{\ell}(x) .
\end{aligned}
\tag{1.9}
$$

Dabei ist $x = \cos(\theta)$. Die nach Drehimpuls sortierten Koeffizienten $E_{\ell\pm}(W)$ und $M_{\ell\pm}(W)$, welche nur noch von der Schwerpunktsenergie W abhängen, werden Multipole genannt. Die unendlichen Reihen lassen sich dabei bei einem maximalen Drehimpuls ℓ_{max} abbrechen, welcher energieabhängig ist. Nahe der Produktionsschwelle ist es oft möglich nur $\ell = 0, 1$ zu betrachten. Je höher die Energie E_γ, desto mehr Ordnungen von ℓ werden notwendig. Das Abbrechen der Reihe bei einem maximalen Wert ℓ_{max} vereinfacht die Gleichung deutlich. Vor allem wenn ℓ_{max} klein ist, können dadurch in günstigen Situationen auch weniger als 8 Polarisationsobservablen ausreichen, um die Multipole ($E_{\ell\pm}$,$M_{\ell\pm}$) bis auf eine energieabhängige Phase zu bestimmen. Zusätzlich ist keine Strahl- & Rückstoßpolarisation oder Target- & Rückstoßpolarisation zwingend erforderlich [WBT14; Ome81]. Da die Messung der Rückstoßpolarisation relativ aufwendig ist, vereinfacht sich dadurch das Erreichen einer vollständigen Datenbasis, mit der die Multipole eindeutig gelöst werden können, erheblich. Da die Resonanzen mit der starken Wechselwirkung zerfallen gilt für die Reaktionen die Drehimpulserhaltung, die Paritätserhaltung und die Isospinerhaltung. Durch die Drehimpulserhaltung kann man Zusammenhänge zwischen den einzelnen Multipolen, dem Drehimpuls und der Parität der Resonanz bilden. Diese Zusammenhänge sind in Tabelle 1.2 aufgeführt.

γN-System		MB-System				γN-System		MB-System			
L	$\mathcal{M}L$	J	ℓ	$\mathcal{M}_{\ell\pm}$	P	L	$\mathcal{M}L$	J	ℓ	$\mathcal{M}_{\ell\pm}$	P
1	$E1$	1/2	0	E_{0+}	-	2	$E2$	3/2	1	E_{1+}	+
			~~1~~						~~2~~		
		3/2	~~1~~					5/2	~~2~~		
			2	E_{2-}	-				3	E_{3-}	+
	$M1$	1/2	~~0~~				$M2$	3/2	~~1~~		
			1	M_{1-}	+				2	M_{2-}	-
		3/2	1	M_{1+}	+			5/2	2	M_{2+}	-
			~~2~~						~~3~~		

Tabelle 1.2.: Aufgeführt sind die Multipole des Meson-Baryon Endzustandes die aus der Dipol- ($L = 1$) und Quadrupolanregung ($L = 2$) des Nukleons entstehen. Die durchgestrichenen Drehimpulse des Endzustandes sind aufgrund der Paritätserhaltung verboten [Wun12; Leu01].

Zusätzlich sind durch die Paritätserhaltung bestimmte Drehimpulse im Endzustand verboten. Die Parität des Anfangszustandes hängt vom Drehimpuls ab (abhängig von der Kopplungsart des Photons):

$$E : P = (-1)^L \,, \tag{1.10}$$

$$M : P = (-1)^{L+1} \,. \tag{1.11}$$

Für den Endzustand gilt:

$$P = (-1)^{\ell+1} \,. \tag{1.12}$$

Daraus folgt:

$$E : (-1)^L = P = (-1)^{\ell+1} \Rightarrow |L - \ell| = 1 \,, \tag{1.13}$$

$$M : (-1)^{L+1} = P = (-1)^{\ell+1} \Rightarrow L = \ell \,. \tag{1.14}$$

Damit sind die in Tabelle 1.2 gestrichenen Endzustände nicht erlaubt. Durch die Isospinerhaltung können Endzustände von Reaktionen, welche Teilchen mit Isospin $I = 0$ enthalten, nicht aus Resonanzen mit Isospin $I = 3/2$ entstehen. Daher können zum Beispiel Reaktionen mit einem η-Meson im Endzustand nur mittels angeregter Nukleonresonanzen ablaufen. Angeregte Δ-Resonanzen sind hier verboten.

Mithilfe dieser aus den Erhaltungssätzen gewonnenen Informationen können nun Schlüsse gezogen werden, welche Quantenzahlen die Resonanzen haben müssen, deren

Beiträge in den einzelnen Multipolen enthalten sind. In Tabelle 1.3 sind diese Eigenschaften aufgeführt und beispielhaft einige bekannte Resonanzen zu jedem Multipol zugeordnet.

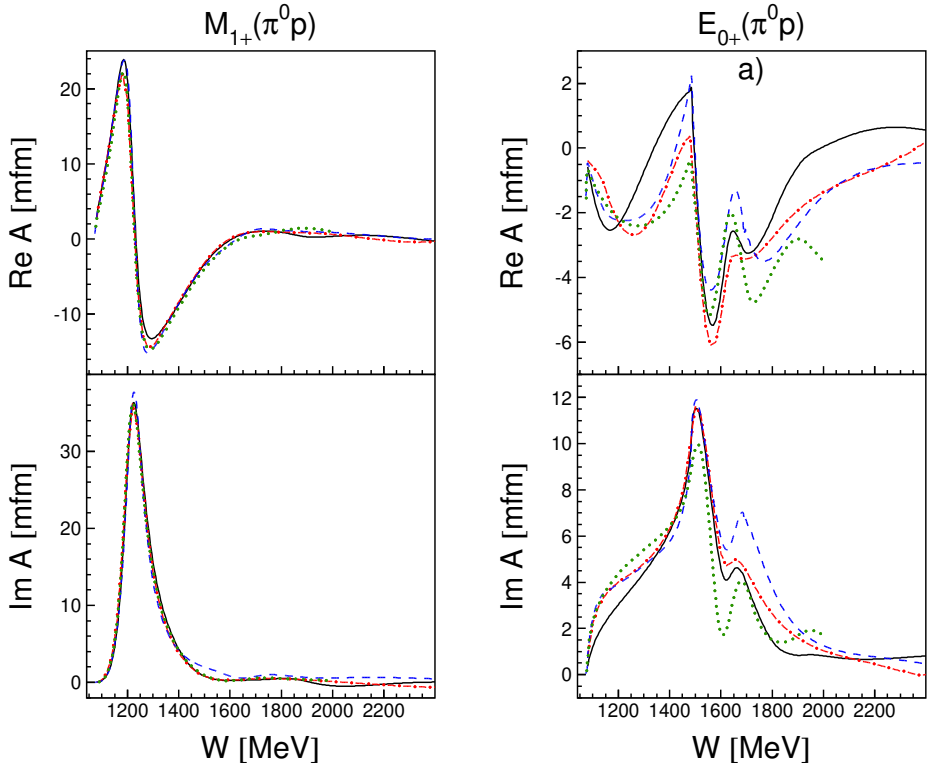
Multipol	L	ℓ	J	P	Resonanzen
E_{0+}	1	0	1/2	-	$N(1535)\frac{1}{2}^-$,$\Delta(1620)\frac{1}{2}^-$,$N(1650)\frac{1}{2}^-$
E_{1+}	2	1	3/2	+	$\Delta(1232)\frac{3}{2}^+$,$\Delta(1600)\frac{3}{2}^+$,$\Delta(1720)\frac{3}{2}^+$,$N(1900)\frac{3}{2}^+$
E_{2-}	1	2	3/2	-	$N(1520)\frac{3}{2}^-$,$\Delta(1700)\frac{3}{2}^-$
E_{3-}	2	3	5/2	+	$N(1680)\frac{5}{2}^+$,$\Delta(1905)\frac{5}{2}^+$
M_{1+}	1	1	3/2	+	$\Delta(1232)\frac{3}{2}^+$,$N(1720)\frac{3}{2}^+$
M_{1-}	1	1	1/2	+	$N(939)\frac{1}{2}^+$,$N(1440)\frac{1}{2}^+$,$N(1710)\frac{1}{2}^+$,$\Delta(1910)\frac{1}{2}^+$
M_{2+}	2	2	5/2	-	$N(1675)\frac{5}{2}^-$,$\Delta(1930)\frac{5}{2}^-$
M_{2-}	2	2	3/2	-	$N(1520)\frac{3}{2}^-$,$\Delta(1700)\frac{3}{2}^-$

Tabelle 1.3.: Für die Multipole aus Dipol- und Quadropolanregung (L = 1,2) sind die Eigenschaften der Reaktion aufgeführt und es wurden beitragende Resonanzen aus [Pat+16] aufgeführt welche als sicher nachgewiesen gelten ($\star\star\star\star$). Zusätzlich wurde vereinzelte Resonanzen aufgeführt, welche als wahrscheinlich bis sicher nachgewiesen ($\star\star\star$) gelten.

Anhand dieser Beispiele wird deutlich, dass in einem Multipol mehrere Resonanzen enthalten sein können. Resonanzen erzeugen im Idealfall dabei einen Nulldurchgang im Realteil des Multipols und ein Maximum im Imaginärteil. In den Abbildungen 1.6(a) und 1.6(b) sind für zwei Multipole beispielhaft der Real- und Imaginärteil aufgeführt. Im Multipol M_{1+} ist deutlich eine Resonanz zu erkennen. Zum Multipol E_{0+} hingegen tragen mehrere Resonanzen bei, sodass die einzelnen Resonanzen nicht mehr klar erkennbar sind. Um nun die beitragenden Resonanzen aus den Multipolen zu extrahieren, ist es notwendig, einen vollständigen Datensatz, mit dem maximal möglichen Informationsgehalt, zu betrachten. Auf diesen Datensatz müssen die im nächsten Abschnitt beschriebenen Methoden den Partialwellenanalyse angewandt werden.

1.4. Partialwellenanalyse

Aus den mit dem Experiment gemessenen Daten werden durch eine Partialwellenanalyse die Resonanzen ermittelt. Dabei wird die komplette Datenbasis in einer Reaktion gleichzeitig betrachtet. Einige Partialwellenanalysen betrachten sogar mehrere Reaktionen gleichzeitig. Diese werden dann Multi-Channel Partialwellenanalysen genannt. Bei der Partialwellenanalyse wird zunächst der in den einzelnen Kanälen auftretende Untergrund modelliert. Dann werden die Resonanzen mit variabler Energie und Breite angepasst.



(a) M_{1+} (b) E_{0+}

Abbildung 1.6.: Real- und Imaginärteil der Multipole M_{1+} (links) und E_{0+} (rechts) im Kanal $\gamma p \to p\pi^0$. Zu sehen sind die Vorhersagen von JüBo (blau), BnGa (schwarz), SAID (rot) und MAID (grün) [Ani+16]. Man erkennt auf der linken Seite deutlich die Resonanz $\Delta(1232)$. Auf der rechten Seite ist gezeigt, wie sich die Resonanzen N(1535) und $\Delta(1620)$ überlagern.

Dafür kann zum Beispiel eine Breit-Wigner Verteilung benutzt werden. Danach können bekannte Resonanzen durch die festgelegten Eigenschaften zugeordnet werden. Zusätzlich können neue Resonanzen durch Anpassen einer weiteren Verteilung gefunden werden. Das Ergebnis wird genauer je mehr verschiedene Observablen und Zerfallskanäle für die Partialwellenanalyse benutzt werden können. Zusätzlich kann die Genauigkeit auch durch größere Raumwinkelabdeckung der Observablen und eine höhere Statistik der Messungen verbessert werden. Wenn eine Partialwellenanalyse durchgeführt wurde können zusätzlich auch noch nicht gemessene Größen vorhergesagt werden und dann vor der Anpassung mit neu gemessenen Daten verglichen werden.

Einige Beispiele für Partialwellenanalysen sind die Bonn-Gatchina-Partialwellenanalyse [A+], das Jülich-Bonn Modell [Rön+], das MAID Modell [Tia+] und die Partialwellenanalyse der George-Washington Universität (SAID) [Bri+]. Genauere Informationen und

ein Vergleich der Partialwellenanalysen findet sich in [Ani+16].

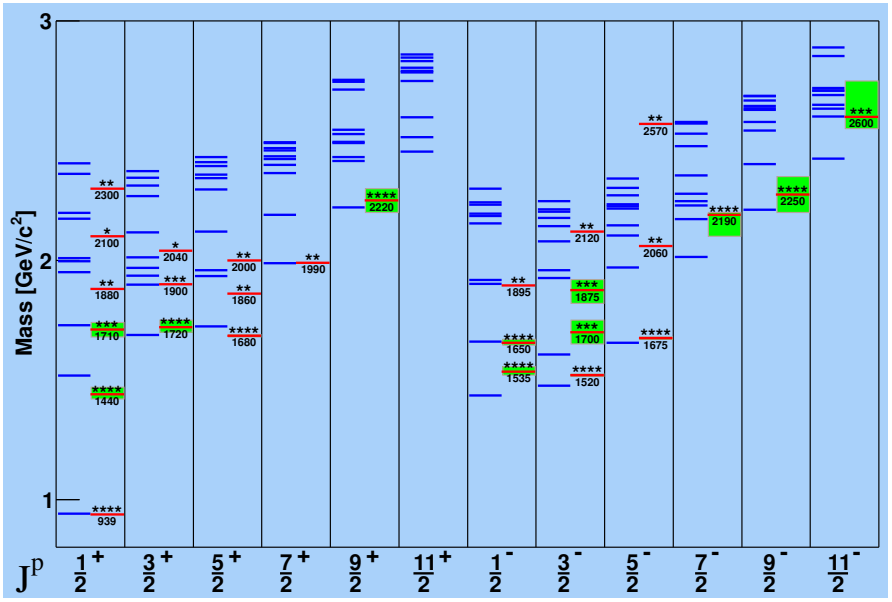
Eine spezielle Form der Partialwellenanalyse ist die abgeschnittene Partialwellenanalyse. Dabei wird ausgenutzt dass die unendlichen Reihen der CGLN-Amplituden aus den Gleichungen 1.10 wie oben beschrieben bei einem maximalen Drehimpuls ℓ_{max} abgebrochen werden können. Die abgeschnittene Partialwellenanalyse wird dabei an die Multipole angepasst und bietet damit eine modellunabhängige Alternative.

1.5. Modelle

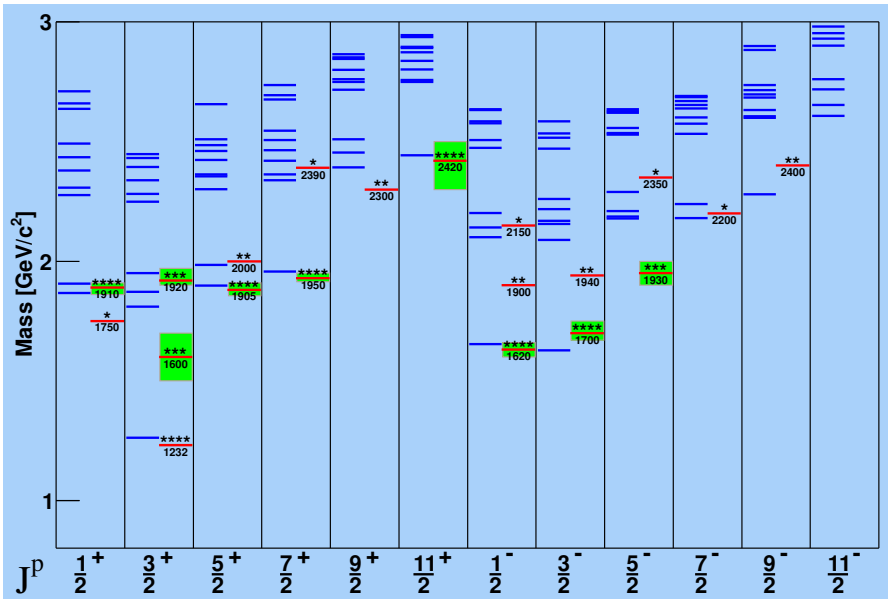
Die mit den in den letzten Kapiteln beschriebenen Methoden erhaltenen Resonanzen kann man dazu benutzen, das Verständnis der Kräfte und des Aufbaus der Nukleonen zu verbessern. Die grundlegende Theorie, welche die Kräfte zwischen den Quarks beschreibt ist die Quantenchromodynamik (QCD). Dabei wird allgemein die starke Kraft zwischen Teilchen mit Farbladung beschrieben. Die QCD kann leider im für die Nukleonresonanzen relevanten Energiebereich nicht wie die Quantenelektrodynamik (QED) störungstheoretischen nach der Kopplungskonstante entwickelt werden. In der QED ist die Kopplungskonstante bei der Berechnung von Prozessen im Niederenergiebereich deutlich kleiner als 1 und damit können Feynman-Diagramme mit einer hohen Zahl von Vertizes vernachlässigt werden. In der QCD ist die Kopplungskonstante im niederenergetische Bereich näher an 1 [Deu05]. Zwei Methoden, die es ermöglichen die Nukleonen durch Annäherungen trotzdem theoretisch zu beschreiben, sind auf der einen Seite Modelle welche die QCD nur in einigen wesentlichen Punkten beschreiben und auf der anderen Seite die Gittereichtheorie.

Ein Beispiel für ein nichtrelativistisches Quarkmodell ist das Modell von Isgur und Karl [IK78]. Dort sind die Nukleonen aus Konstituentenquarks aufgebaut, welche mit dem Potential eines harmonischen Oszillators gebunden sind. Dieses Potential ist zusätzlich noch um eine unharmonische Störung, sowie einem Hyperfeinwechselwirkungsterm erweitert. Ein Beispiel für ein relativistisches Quarkmodell ist das Modell von Löring, Kretzschmar, Metsch und Petry [Lör+01]. Darin wird mit instantoninduzierten Zwei- und Dreikörperkräften gerechnet, welche in die Bethe-Salpeter Gleichung eingesetzt werden.

In Abbildung 1.7 gezeigt ist der Vergleich der Nukleonresonanzen des Quarkmodells von Löring, Kretzschmar, Metsch und Petry mit der experimentellen Datenbasis aus der Quelle [Oli+14]. Man erkennt deutliche Unterschiede. So sind im Modell erkennbar mehr Resonanzen zu finden als gemessen worden sind, besonders im hohen Energiebereich. Auch im niedrigeren Energiebereich sind jedoch Unterschiede zu erkennen. So sind zum Beispiel die energetische Abfolge der Resonanzen mit J = $\frac{1}{2}^-$, $\frac{3}{2}^-$ und $\frac{5}{2}^-$ bei Modell



(a) N^*



(b) Δ^*

Abbildung 1.7.: Dargestellt in Blau sind die Vorhersagen der Massen der Nukleonresonanzen aus dem Quarkmodell von Löring, Kretzschmar, Metsch und Petry [Lör+01]. Im Vergleich dazu sind in Rot die experimentell erhaltenen Resonanzen aus [Oli+14] dargestellt mit ihren Messfehlern in Grün. Die Werte sind dabei nach Drehimpuls und Parität sortiert.

und experimentellen Daten nicht identisch. Wie relevant dies jedoch ist, lässt sich auf Grund der fehlenden Unsicherheiten bei der Modellvorhersage nicht sagen.

Eine unabhängige Methode, um die Vorhersagen der Quarkmodelle zu überprüfen ist die Gittereichtheorie[Wil74]. In der Gittereichtheorie wird das Pfadintegral der Quantenfeldtheorie auf einer diskretisierten und euklidischen Raumzeit berechnet. Dadurch ist es möglich, auch im nicht störungstheoretischen Bereich die QCD numerisch zu lösen.

Der Rechenaufwand für eine relevante Aussage ist jedoch immer noch sehr hoch. Da der Rechenaufwand unter anderem von der angenommen Masse der Konstituentenquarks abhängt, wird diese auf nichtphysikalische Werte erhöht und das Verhalten mit korrekten Massen daraus approximiert. Somit hat die Gittereichtheorie gegenüber den Quarkmodellen den Vorteil, dass keine Annahmen über die Kräfte gemacht werden, aber dafür können die Ergebnisse durch die unphysikalischen Quarkmassen verfälscht sein. Durch den Vergleich der voneinander unabhängigen Ansätze der phänomenologischen Modelle und der Gitter-QCD mit den experimentell erhaltenen Daten können aber dennoch Erkenntnisse gewonnen werden.

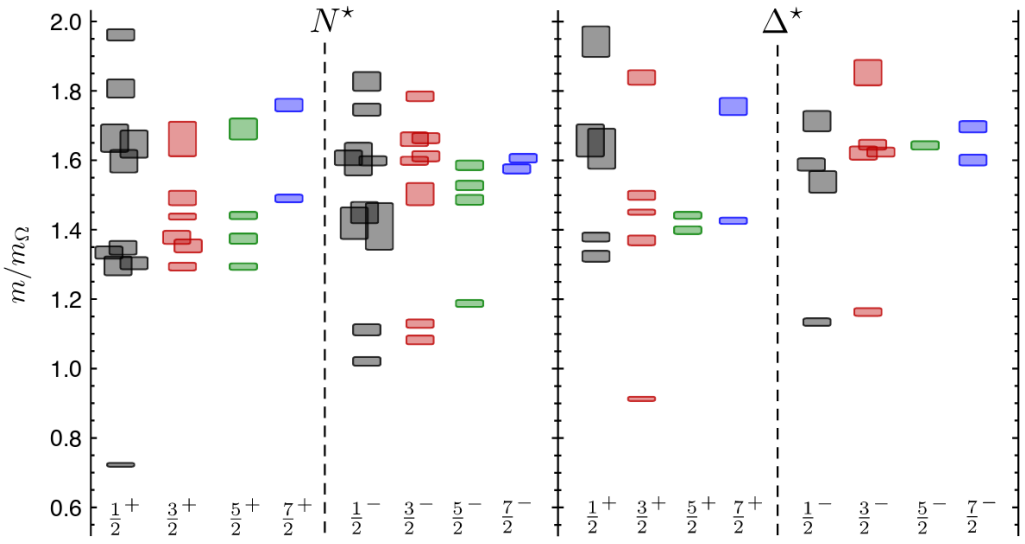


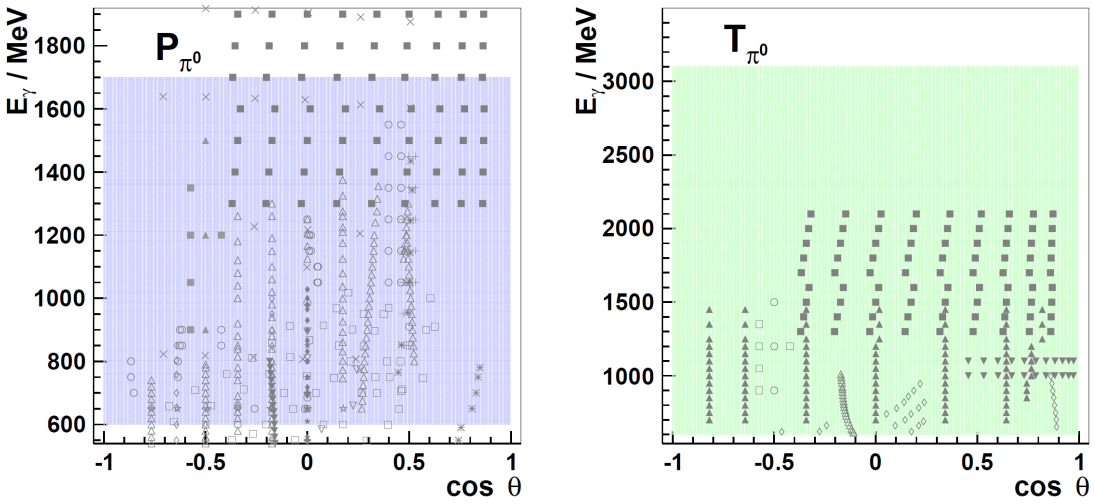
Abbildung 1.8.: Dargestellt sind die Anregungszustände des Nukleons aufgeteilt nach Drehimpuls und Parität aus einer Berechnung von Edwards, Dudek, Richards und Wallace [Edw+11]. Es wurde dabei ein unphysikalische Pion-Masse von 396 MeV angenommen.

In Abbildung 1.8 sind die Vorhersagen der Gittereichrechnungen von Edwards, Dudek, Richards und Wallace gezeigt [Edw+11]. Sie sind analog zu den Quarkmodellen nach

Drehimpuls und Parität getrennt aufgetragen. Hier sind ebenfalls mehr Resonanzen zu erkennen, als experimentell gefunden wurden. Die Rechnungen haben dabei den Nachteil, dass ein unphysikalische Pion-Masse von 396 MeV angenommen wurde, und dass die gezeigten Resonanzen nicht wie in der Natur zerfallen können. Es ist jedoch auffällig, dass sich beide unabhängigen Ansätze (Quarkmodelle und Gittereichrechnungen) einig sind, und mehr Resonanzen vorhersagen als bisher experimentell gefunden wurden. Es scheint hier also eine Diskrepanz zwischen den experimentellen Ergebnissen und den theoretischen Vorhersagen zu geben. Dies kann daran liegen, dass experimentell noch nicht alle Resonanzen gefunden wurden oder dass das theoretische Verständnis der Kräfte in den Nukleonen noch nicht vollständig ist. Um auf experimenteller Seite die Suche nach neuen, noch nicht gefunden, Resonanzen voranzubringen, ist es zunächst nötig statt der bisher hauptsächlich betriebenen Pion-Nukleon Streuung mehr Experimente in der Photonproduktion durchzuführen. Dies ermöglicht durch die Polarisation des Photons den Zugriff auf eine größere Anzahl an Polarisationsobservablen. Zusätzlich wird der Reaktionskanal nicht durch ein Pion im Anfangszustand eingeschränkt. Zusätzlich ist es notwendig mehr Polarisationsobservablen zu messen und für die schon gemessenen Observablen den Energie- und Winkelbereich sowie die Genauigkeit zu erhöhen.

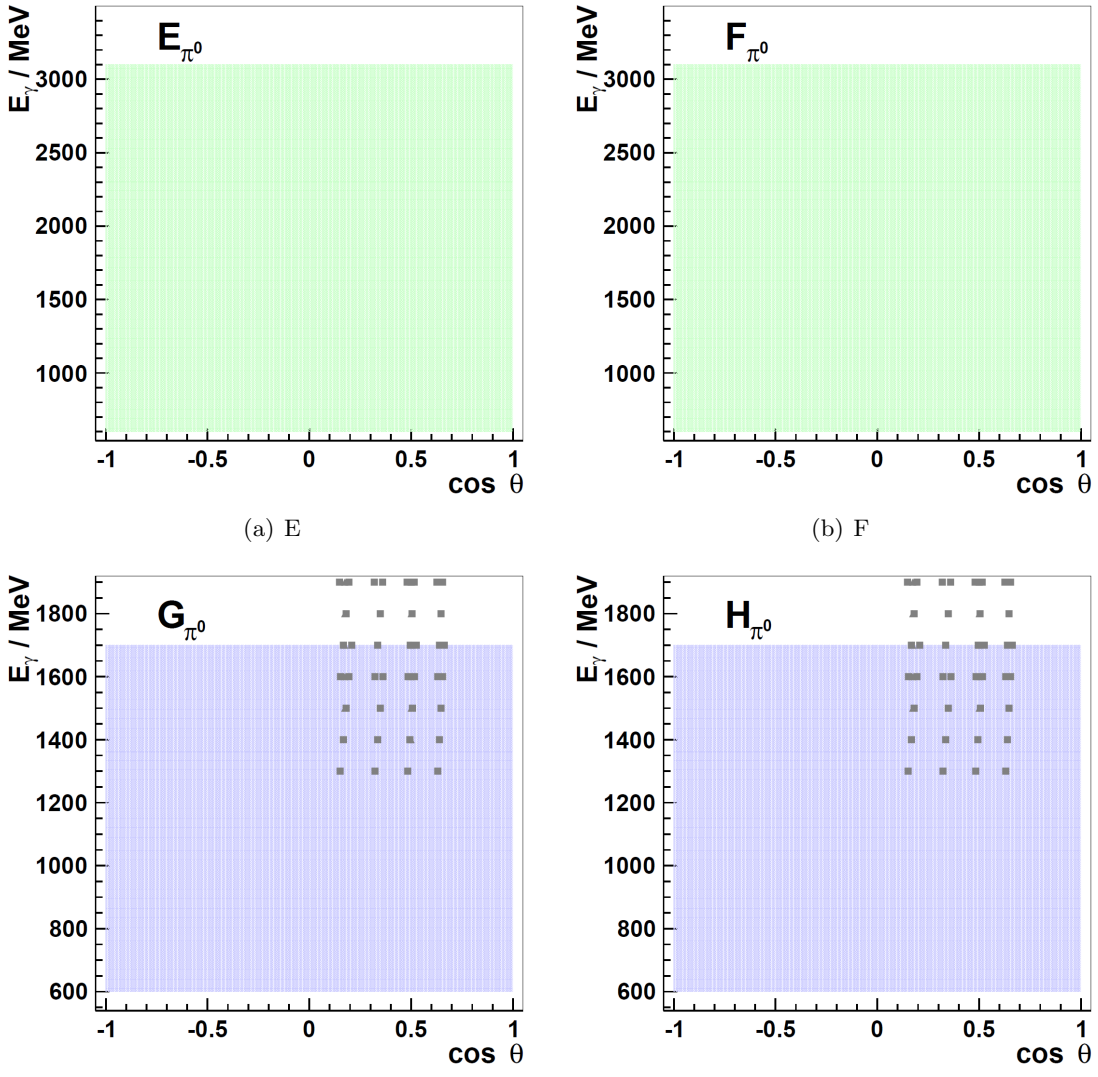
1.6. Experimentelle Datenbasis vor dem Crystal-Barrel/TAPS-Experiment

In den Abbildung 1.9 bis 1.11 ist der Stand der Messdaten der Kanäle $\gamma p \to p\pi^0$ und $\gamma p \to p\eta$ vor Beginn dieser Arbeit gezeigt. Dort ist zu erkennen, dass für viele Observablen nur wenige oder gar keine Daten veröffentlicht waren. Ziel dieser Arbeit war es ein System zu entwickeln, mit Hilfe dessen die Daten von Photoproduktionsexperimenten mit Doppelpolarisation am Crystal-Barrel/TAPS-Experiment gespeichert werden können.



(a) P (b) T

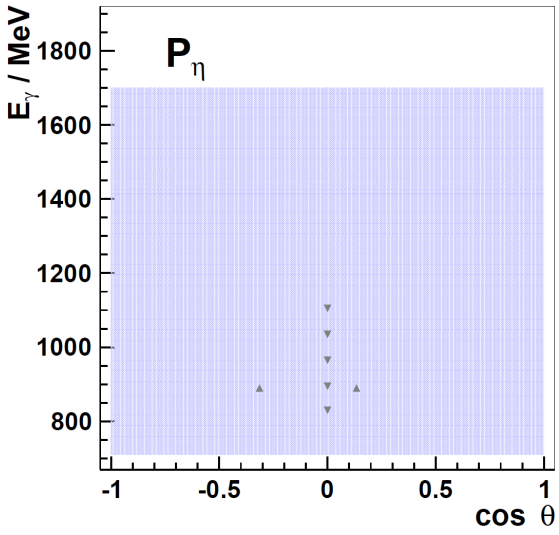
Abbildung 1.9.: **Observablen der Rückstoßpolarisation P und Targetasymmetrie T im Kanal $\gamma p \to p\pi^0$.** Gezeigt ist die Datenbasis vor den neuen Messungen mit dem Crystal-Barrel/TAPS-Experiment. Bilder aus [Afz18a].



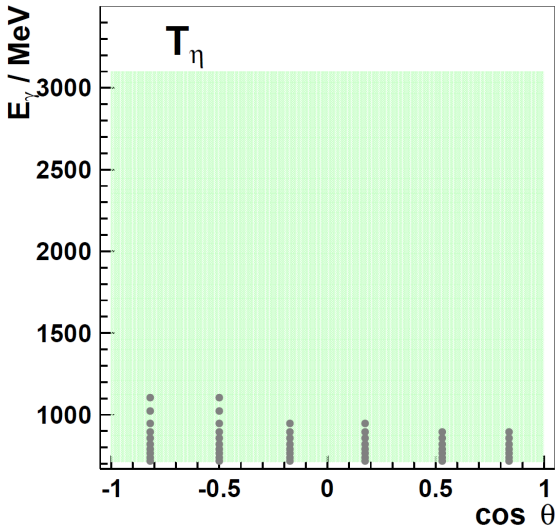
(a) E (b) F

(c) G (d) H

Abbildung 1.10.: **Doppelpolarisationsobservablen bei polarisiertem Strahl und polarisiertem Target E, F, G und H im Kanal $\gamma p \to p\pi^0$.** Gezeigt ist die Datenbasis vor den neuen Messungen des Crystal-Barrel/TAPS-Experiments. Bild aus [Afz18a].



(a) P



(b) T

Abbildung 1.11.: **Observablen der Rückstoßpolarisation P und Targetasymmetrie T im Kanal $\gamma p \to p\eta$.** Gezeigt ist die Datenbasis vor den neuen Messungen mit dem Crystal-Barrel/TAPS-Experiment. Bild aus [Afz18a].

Die Doppelpolarisationsobservablen bei polarisiertem Strahl und polarisiertem Target (siehe Tabelle 1.1) E, F, G und H im Kanal $\gamma p \to p\eta$ hatten vor dem Crystal-Barrel/TAPS-Experiment noch keine Werte und sind daher nicht gezeigt.

1.7. Aufbau der Arbeit

Im letzten Abschnitt wurde gezeigt, dass es notwendig ist mit dem Crystal-Barrel/TAPS-Experiment weitere Messungen durchzuführen. Dabei ist es besonders wichtig Daten mit Doppelpolarisation, also mit polarisiertem Strahl und Target, zu speichern. Dafür wurde das Crystal-Barrel/TAPS-Experiment zunächst umgebaut. Der Stand nach dem Umbau ist in Kapitel 2 beschrieben. Für das neue Experiment war es dann notwendig einen Trigger (Kapitel 3) und eine neue Datenerfassung zu entwickeln. Die im Rahmen dieser Arbeit entwickelte Datenerfassung besteht dabei zum Einen aus einem in Hardware entwickelten Synchronisationssystem (Kapitel 4). Zum Anderen wurde eine Software für die eigentliche Datennahme entwickelt (Kapitel 5). In Kapitel 6 wird die Leistungsfähigkeit des Systems untersucht. Kapitel 7 gibt einen kleinen Überblick über einige Messungen, die mit der Datenakquisition durchgeführt wurden und welche Ergebnisse dadurch erzielt wurden.

2. Das Experiment

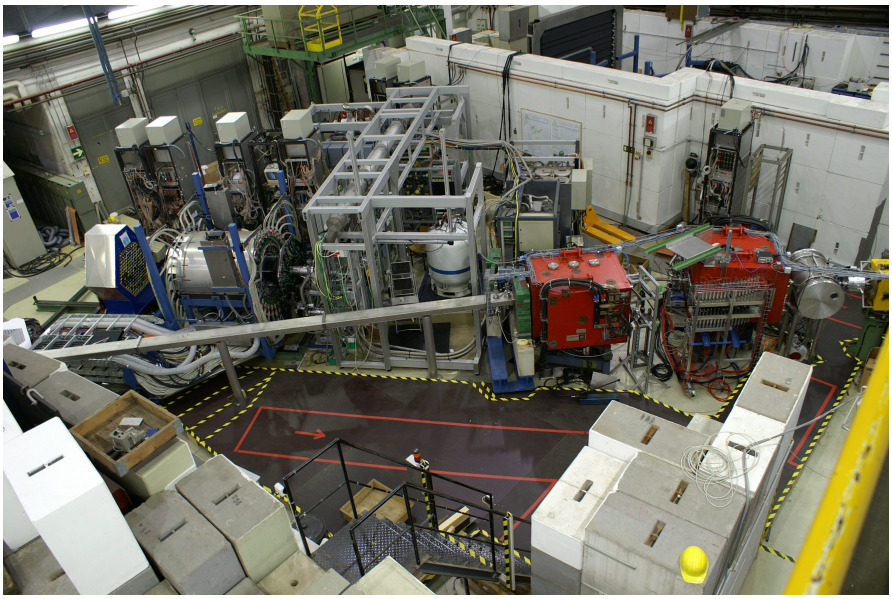


Abbildung 2.1.: **Überblick über das Crystal-Barrel/TAPS-Experiment.**

Das Crystal-Barrel/TAPS-Experiment befindet sich an der Elektron-Stretcher-Anlage in Bonn (siehe Kapitel 2.1) an der Elektronen mit einer Energie von bis zu 3,5 GeV erzeugt werden können. In den folgenden Kapiteln wird der Teilchenbeschleuniger ELSA sowie die einzelnen Detektorkomponenten des Experiments beschrieben. In Kapitel 3 wird der Trigger[1] beschrieben, mit dem die Auslese der Experimentdaten gestartet wird.

2.1. Elektron-Stretcher-Anlage (ELSA)

Die, für die Photoproduktion benötigten Photonen, werden durch Streuung eines Elektronenstrahls erzeugt. Dieser Elektronenstrahl mit einer Energie von bis zu 3,5 GeV wird von der Elektron-Stretcheranlage (ELSA) zur Verfügung gestellt [Hil06]. Diese kann dabei sowohl unpolarisierte als auch linear polarisierte Elektronen mit einer Energie von bis zu 3,5 GeV bereitstellen. Die unpolarisierten Elektronen werden dabei in einer

[1] engl. für Auslöser. System, welches beurteilt, ob gerade ein gewünschtes Ereignis stattgefunden hat und im positiven Fall die Auslese startet.

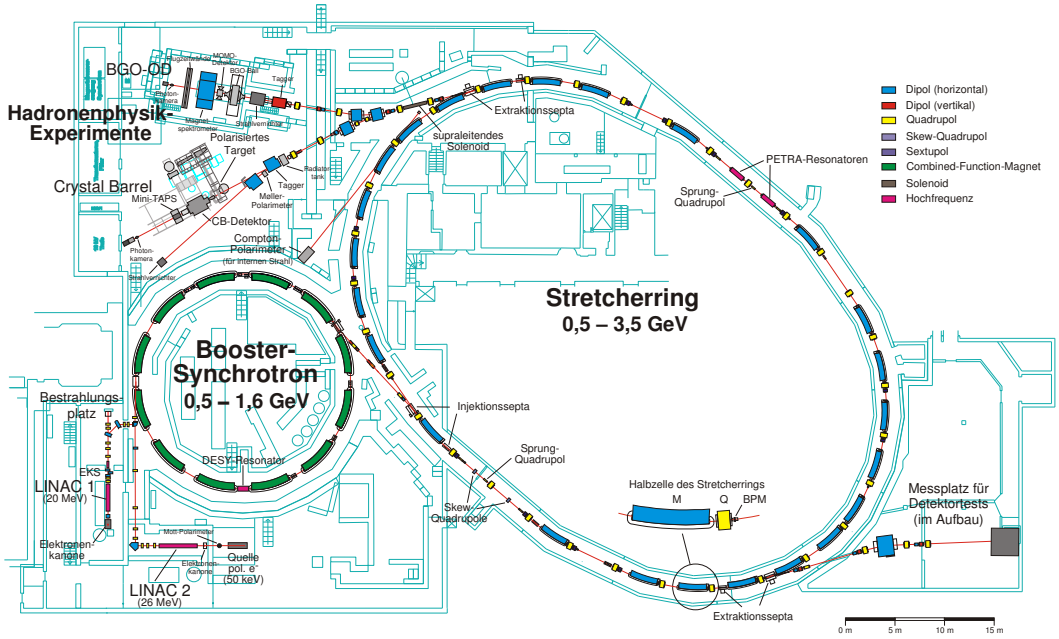


Abbildung 2.2.: **Aufbau der Elektron-Stretcher-Anlage (ELSA).** In einer der beiden Quellen werden Elektronen produziert und durch einen Linearbeschleuniger (LINAC) zum Vorbeschleuniger (Boostersynchrotron) geleitet. Die finale Energie wird dann im eigentlichen ELSA-Ring erreicht und die beschleunigten Elektronen an eines der Experimente extrahiert. Bild aus [Fro16].

sogenannten thermischen Quelle erzeugt, indem Elektronen thermisch ausgelöst und dann von einem elektrischen Feld auf eine Energie von bis zu 48 keV beschleunigt werden. Die polarisierten Elektronen werden erzeugt indem ein Kristall mit einem zirkular polarisierten Ti:Sa Laser bestrahlt wird. Die von der Quelle erzeugten Elektronen werden dann mit einem Linearbeschleuniger auf eine Energie von bis zu 26 MeV gebracht und zur weiteren Beschleunigung in das Boostersynchrotron geleitet, wo die Elektronen auf eine Energie von bis zu 1,6 GeV beschleunigt werden. Anschließend werden sie in den eigentlichen Stretcherring injiziert. Dieser Prozess wird mehrfach wiederholt bis der Stretcherring homogen gefüllt ist. Anschließend werden die Elektronen weiter auf die vom Experiment gewünschte Energie gebracht und an einen der drei Experimentplätze extrahiert. Zur Extraktion wird dabei, durch Anregung einer Resonanz, ein Teil der Elektronen auf eine instabile Bahn gebracht. Diese Elektronen werden dann an das Experiment geleitet. Durch diesen Vorgang kann für eine gewisse Zeit kontinuierlich ein Teil der im Beschleuniger vorhandenen Elektronen entnommen werden. Die Zeit während der dieser kontinuierliche Strahl extrahiert wird, wird Extraktionszeit oder auch

“Spill“ genannt. Die Pause während der, auf Grund des Füllens und des Beschleunigungsvorgangs im ELSA-Ring, keine Elektronen extrahiert werden können wird “Spillpause“ genannt. Eine Übersicht über die ELSA ist in Abbildung 2.2 gezeigt.

2.2. Goniometer

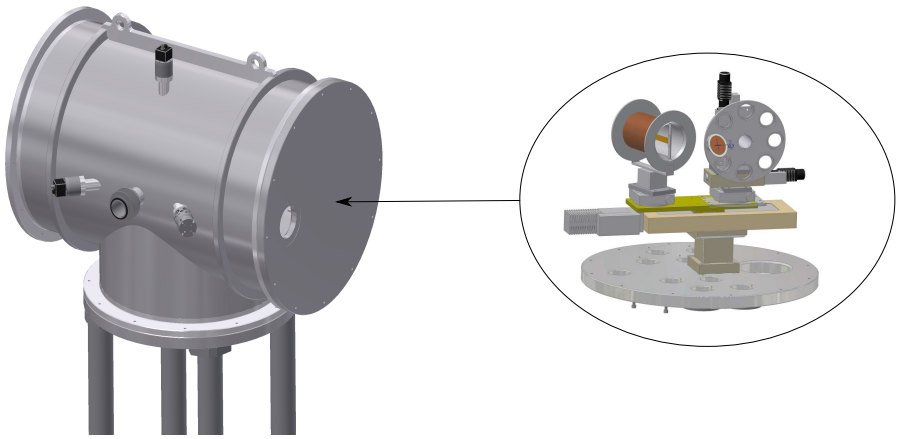


Abbildung 2.3.: **Aufbau der Goniometerkonstruktion.** Auf der linken Seite ist der Vakuumtank der Goniometerkonstruktion gezeigt. Auf der rechten Seite zu sehen ist die Goniometerkonstruktion mit der Möllerfolie im Helmholtzspulenpaar (links) und die Goniometerscheibe mit den verschiedenen Radiatoren (rechts) [Wal18].

Um die vom Experiment benötigten Photonen zu erzeugen, wird am Crystal-Barrel/TAPS-Experiment der Prozess der Bremsstrahlung ausgenutzt. Dazu stehen in einem evakuierten Tank eine Reihe von Bremsstrahlradiatoren zur Verfügung die mit einer Goniometerkonstruktion in den Strahl gefahren werden können. Zur Erzeugung von unpolarisierten und zirkular polarisierten Photonen stehen Kupferradiatoren mit Dicken von 12 µm, 50 µm, 150 µm und 300 µm zur Verfügung. Für die Erzeugung von linear polarisierten Photonen wird der unpolarisierte Elektronenstrahl auf einen Diamanten geleitet, der durch die Goniometerkonstruktion um alle drei Raumachsen gedreht werden kann.

Dadurch kann das Gitter des Diamanten so positioniert werden, dass sich auf Grund des Rückstoßvektors in einem ausgewählten Energiebereich eine Anhäufung polarisierter Photonen ergibt. Zirkular polarisierte Photonen können an beliebigen amorphen Radiatoren aus longitudinal polarisierten Elektronen erzeugt werden. Um eine Messung des Polarisationsgrades der Elektronen und damit auch der Photonen zu ermöglichen, wird jedoch das Möllertarget benutzt [Kam10]. Das Möllertarget wird genauer in Abschnitt 2.4

beschrieben. Es kann mit einem der Fahrtische der Goniometerkonstruktion alternativ zur Goniometerscheibe in den Elektronenstrahl gefahren werden. Zusätzlich kann noch der Winkel der Möllerfolie relativ zum Strahl eingestellt werden.

2.3. Tagger

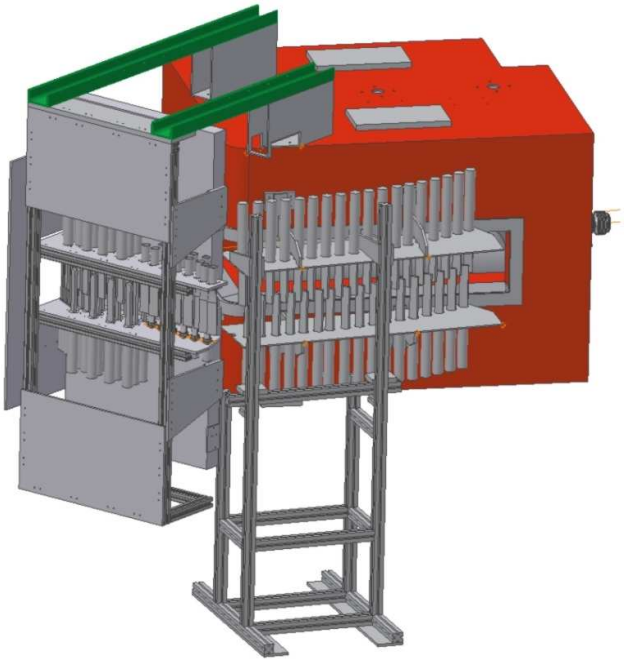


Abbildung 2.4.: **Aufbau des Taggerdetektors.** Am Ablenkmagneten (rot) sind die szintillierenden Latten und Fasern des Taggerdetektors befestigt [Wal18].

Um die Energie der im Bremsstrahlradiator erzeugten Photonen zu bestimmen wird das Tagging-Spektrometer [For09] benutzt. Der Tagger bestimmt die Photonenenergie (E_γ) indirekt, indem die Energie des im Radiator gestreuten Elektrons (E_{Tagger}) bestimmt wird (nachdem es das Bremsstrahlphoton erzeugt hat). Da die Energie des Elektrons vor dem Radiator (E_{ELSA}) bekannt ist, kann, unter Vernachlässigung des Rückstoßes, mit der Formel

$$E_\gamma = E_{ELSA} - E_{Tagger}$$

die Energie des Photons bestimmt werden. Die Elektronenenergie wird durch den Tagger bestimmt indem zunächst die Elektronen in einem Magnetfeld ablenkt werden. Wenn die Magnetfeldstärke bekannt ist, kann dann anhand des Ablenkwinkels die Energie bestimmt werden. Um den Ablenkwinkel zu bestimmen, wird eine Kombination von szintillierenden Latten und Fasern benutzt (Siehe Abbildung 2.4) aus deren Position

sich der Ablenkwinkel ergibt. Es sind dabei 96 Latten und 480 Fasern verbaut. Die Fasern sind im strahlnächsten Bereich des Taggers hinter den Latten platziert um die Winkelauflösung und damit die Energieauflösung im niederenergetischen Bereich der Photonen zu verbessern. Die Latten sind in zwei Reihen versetzt hintereinander aufgebaut, so dass durch die Koinzidenz von jeweils zwei Latten die Energieauflösung verbessert werden kann. Es wird dadurch mit den Latten eine Energieauflösung im Bereich von $\Delta E = 0{,}1\%E_\gamma - 0{,}6\%E_\gamma$ erreicht und mit den Fasern eine Energieauflösung im Bereich von $\Delta E = 0{,}1\%E_\gamma - 0{,}29\%E_\gamma$. Das in den szintillierenden Detektoren erzeugte Licht wird mit Photomultipliern in elektrische Signale umgewandelt und dann auf TDCs[2] zur Zeitbestimmung gegeben. Zusätzlich werden von den Signalen der szintillierenden Latten drei Triggersignale erzeugt. Zunächst wird ein Oder-Signal der Treffer in allen 96 Latten ausgegeben. Dann wird jeweils eine Koinzidenz von zwei benachbarten Latten gebildet und das Oder-Signal der daraus entstehenden Signale ausgegeben. Dadurch kann das Rauschen einzelner Latten unterdrückt werden. Das Signal der Koinzidenzen wird zusätzlich auf Zähler gegeben, welche für die Bestimmung des Photonenflusses benutzt werden. Das dritte Triggersignal ist ein Oder-Signal der Koinzidenzen der 32 Latten mit der niedrigsten Elektronenergie. In diesem Bereich sind die Raten der Signale in den Latten am niedrigsten. Deshalb treten in diesem Oder-Signal weniger Sättigungseffekte bei hohen Elektronstrahlströmen auf.

2.4. Möller-Polarimeter

Zur Bestimmung des Polarisationsgrades eines zirkular polarisierten Photonenstrahls gibt es das Möllerpolarimeter [Kam10]. Der Möllerdetektor besteht aus einer Metallfolie, an der Möllerpaare erzeugt werden und einem Bleiglasdetektor in dem sie nachgeweisen werden. Die Möllerfolie ist im Goniometertank (Kapitel 2.2) in einem Magnetfeld positioniert. Das Magnetfeld, dass die Folie polarisiert, wird von einem Helmholtzspulenpaar erzeugt um eine möglichst gleichmäßige Polarisierung zu erreichen. Des weiteren besteht der Detektor aus 4 Bleikristallen, welche hinter den szintillierenden Latten des Taggerdetektors positioniert sind. Dabei sind jeweils 2 Bleikristalle unterhalb und oberhalb der Strahlachse angebracht. Das Licht der Kristalle wird mit Photomultipliern in elektrische Signale umgewandelt und dann mit Hilfe von TDCs und ADCs[3] digitalisiert sowie auf zwei Zähler gegeben. Von der ELSA werden zwei Signale zur Verfügung gestellt, welche die Polarisationsrichtung anzeigen. Mit diesen Signalen wird jeweils einer der

[2] Abkürzung für Time to Digital Konverter. Ein Elektronikmodul mit dem Zeiten gemessen und in digital zu verarbeitende Zahlen gewandelt werden.

[3] Abkürzung für Analog to Digital Converter. Ein Elektronikmodul, welche die Amplitude eines Signal oder die Fläche unter einem Signal misst und in eine digital zu verarbeitende Zahl umwandelt

beiden Zähler aktiviert. Dadurch zählt jeweils nur ein Zähler während jeder der beiden Polarisationsrichtungen der Elektronen.

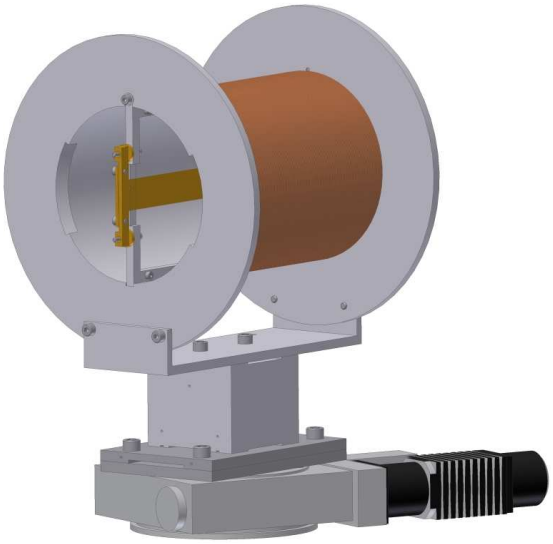


Abbildung 2.5.: **Anordnung der Streufolie des Möllerpolarimeters.** Die Möllerfolie besteht aus $20\mu m$ dickem Vacoflux[4] und ist von einem Helmholtzspulenpaar umgeben [Wal18].

Bei der Möllerstreuung hängt der Wirkungsquerschnitt davon ab wie die Spins der beiden Elektronen zueinander ausgerichtet sind. Wenn die Spins parallel zueinander ausgerichtet sind, ist aufgrund des Pauli-Prinzips die Streurate geringer. Wenn der Polarisationsgrad in der Metallfolie bekannt ist, kann aus der Ratendifferenz zwischen den beiden möglichen Polarisationsrichtungen der Strahlelektronen der Polarisationsgrad der Strahlelektronen bestimmt werden.

2.5. Beamdump

Die meisten der aus ELSA extrahierten Elektronen sind an keiner Reaktion im Bremsstrahltarget beteiligt. Damit diese Elektronen zu keinem erhöhten Strahlungsniveau in den umliegenden Gebäuden, der Umgebung und den Detektoren führen, müssen diese abgebremst und abgeschirmt werden. Die Abschirmung ist dabei so zu wählen, das sämtliche entstehende Strahlung auf ein minimales Niveau gesenkt wird. Um dies zu erreichen wird zunächst ein 470 kg schwerer Eisenblock benutzt, der vakuumisoliert aufgestellt ist und als Faraday-Cup[5] wirkt. Er ermöglicht die Messung der Stromstärke

[4] Legierung aus 49 % Eisen, 49 % Cobalt und 2 % Vanadium. Vertrieben von der Firma Vacuumschmelze.
[5] Ein Faraday-Cup ist ein metallisches Gefäß mit dem Elektronen im Vakuum aufgefangen werden können. Die Ladung der aufgefangenen Elektronen kann dann gemessen werden.

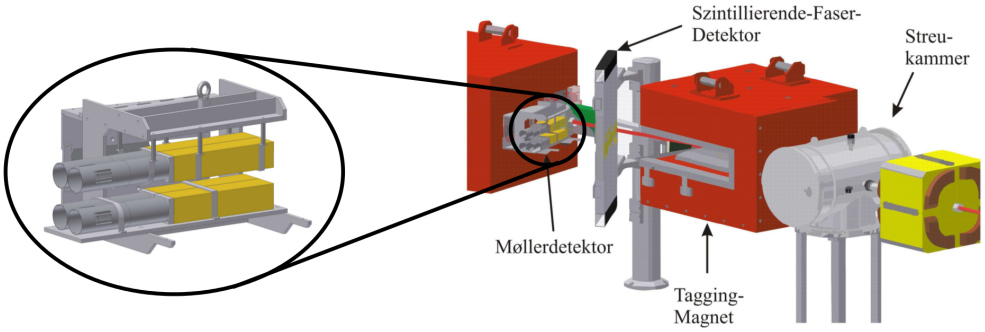


Abbildung 2.6.: **Aufbau der Bleiglasdetektoren des Möllerdetektors.** Die Möllerdetektoren bestehen aus Bleiglas und sind beidseitig der Strahlebene hinter den Taggerfasern angebracht [Ebe06].

des extrahierten Strahls im Bereich von 1 pA bis 500 nA. Der Faraday-Cup ist in einer weiteren Schicht von insgesamt 70 t Eisenblöcken umgeben.

2.6. Bonn-Frozen-Spin-Target

Abbildung 2.7.: **Darstellung des Bonn-Frozen-Spin-Targets.** Gezeigt ist der Kryostat des Targets mit der Targetzelle am linken Ende [Wal18].

Die Nukleonen für die Reaktionen der Mesonphotoproduktion werden vom Target[6] bereitgestellt. Im Bonn Frozen Spin Target [B+99] können die Nukleonen zusätzlich polarisiert werden. Für Protonen werden diese in einem Kryostaten zunächst auf unter 300 mK gekühlt. Dazu wird eine Kombination aus flüssigem He-3 und He-4 benutzt. Die Nukleonen befinden sind dann weitestgehend in Ruhe und haben somit sehr wenig

[6] engl. für Ziel

kinetische Energie. Mit Hilfe eines Magneten mit einer Magnetfeldstärke von 2,5 T werden dann die Spins der freien Elektronen polarisiert. Durch Mikrowellen wird anschließend ein gemeinsamer Spinflip von Proton und Elektron induziert. [CM97]. Da die Relaxationszeit der Elektronen im Vergleich zu den Protonen deutlich niedriger ist, können die freien Elektronen dann fast direkt wieder für weitere Spinflips genutzt werden. Mit diesem Vorgang kann über einige Stunden der Polarisationsgrad der Protonen erhöht werden. Zur Polarisation von Neutronen kann Deuterium verwendet werden. Da dort die Spins der Protonen und Neutronen verbunden sind, wird das Neutron durch den gleichen Vorgang mitpolarisiert. Wenn der Polarisationsvorgang beendet ist, wird die Temperatur des Kryostaten auf unter 100 mK reduziert um die Polarisation der Nukleonen ”einzufrieren“. Das Magnetfeld wird dann auf 0,5 T reduziert und von einer kleinen Haltespule, welche sich im Kryostaten befindet, übernommen. Dadurch kann der große Magnet entfernt werden. Der Polarisationsgrad nimmt exponentiell mit der Zeit ab. Wie lange dabei die Halbwertzeit ist, hängt unter anderem linear von der Temperatur ab, weswegen es wichtig ist eine möglichst niedrige Temperatur des Targetmaterials zu erreichen und diese Temperatur stabil zu halten. In Abbildung 2.7 ist der Kryostat schematisch dargestellt. Das Targetmaterial im Kryostaten muss einigen Anforderungen genügen. Es muss polarisierbare Protonen beziehungsweise Neutronen in möglichst hoher Dichte enthalten. Außerdem muss der maximal erreichbare Polarisationsgrad und die Halbwertzeit mit der der Polarisationsgrad abnimmt möglichst hoch sein. Beide Anforderungen werden durch Butanol erfüllt. Für Protonen wird dabei direkt Butanol benutzt. Für Neutronen wird D-Butanol benutzt. Das Targetmaterial ist dabei zusätzlich mit paramagnetischen Radikalen dotiert um die freien Elektronen für die Polarisation zu erhalten.

2.7. Wasserstofftarget

Zusätzlich zum polarisierten Target steht auch noch ein unpolarisiertes Target zur Verfügung. Mit diesem können Messungen mit reinem Protonenmaterial und ohne Magnetfeld durchgeführt werden. Das Target besteht vereinfacht gesehen aus einer Targetzelle mit 3 cm Durchmesser und 5,1 cm Länge, sowie einer Kühlvorrichtung die das Targetmaterial soweit abkühlt das es in einen flüssigen Zustand übergeht. Als Targetmaterialien können Wasserstoff und Deuterium verwendet werden. Dadurch können Studien an reinen Protonen durchgeführt werden, sowie die Auswirkungen von zusätzlichen Neutronen studiert werden. Zusätzlich können die Daten der Wasserstoffmessungen zur Bestimmung des Anteils an polarisierbaren Protonen im Butanolgemisch benutzt werden. Abbildung 2.8 zeigt die Targetzelle mit ihrer Halterung, die auch den Zufluss und Abfluss des Targetmaterials ermöglicht.

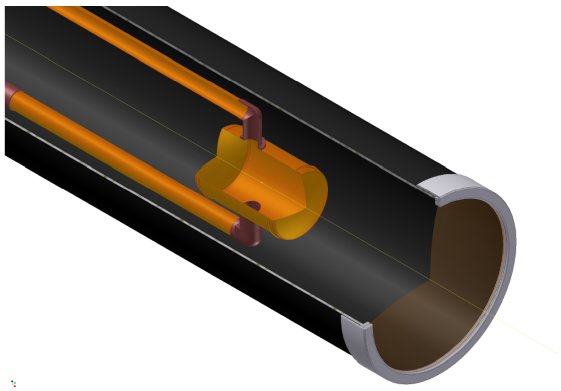


Abbildung 2.8.: **Das unpolarisierte Wasserstoff Target.** Gezeigt ist innerhalb des Vakuumstrahlrohrs (schwarz) die Targetzelle, welche den flüssigen Wasserstoff des Targets aufnimmt (orange). Zusätzlich sind in Orange die Vor- und Rücklaufleitungen des LH_2-Kreislaufes gezeigt [Wal18].

2.8. Innendetektor

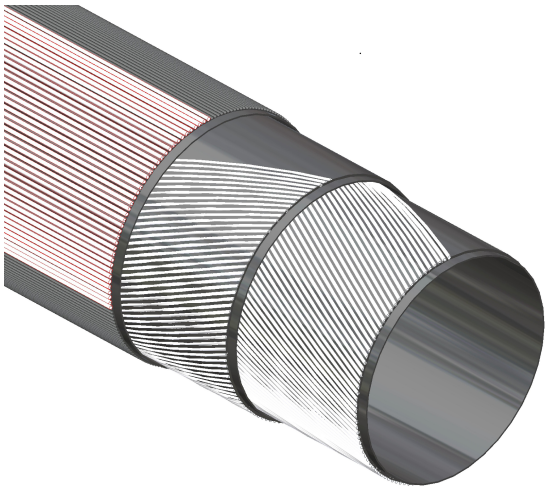


Abbildung 2.9.: **Schematische Darstellung des Innendetektors.** Gezeigt sind die 3 Lagen des Innendetektors. Die beiden inneren Lagen sind um +25,7° beziehungsweise -24,5° gegenüber der äußeren Lage gedreht [Grü18].

Um eine Differenzierung zwischen geladenen und ungeladenen Teilchen im Crystal-Barrel-Detektor zu ermöglichen ist um das Target herum der Innendetektor positioniert [Grü06]. Dieser besteht aus drei Lagen szintillierender Fasern, wobei die äußere Lage parallel zur Strahlrichtung angebracht ist, während die beiden inneren Lagen um +25,7° beziehungsweise -24,5° dazu gedreht sind. Diese Anordnung (gezeigt in Abbildung 2.9)

wurde so gewählt damit zwei Fasern nicht mehr als einen Kreuzungspunkt miteinander haben. Dadurch kann aus einem Treffer in zwei Fasern eine eindeutige Position auf dem Detektor und damit der Durchstoßpunkt bestimmt werden. Insgesamt besteht der Innendetektor aus 513 Fasern mit einem Durchmesser von 2 mm und überdeckt einen Polarwinkelbereich von 24° bis 166°. Das Licht der Fasern wird mit 16-fach Photomultipliern in elektrische Signale umgewandelt. Dieses elektrische Signal wird dann mit TDCs digitalisiert, welche den Zeitpunkt der Treffer im Detektor bestimmen. Zusätzlich werden aus den Photomultipliersignalen noch zwei Signale generiert, welche in der ersten Triggerstufe benutzt werden können. Das erste Signal ist ein simples Oder-Signal aller Fasertreffer. Das zweite wird nur ausgegeben, wenn gleichzeitig in zwei Lagen ein Treffer registriert wurde. Dadurch kann das Rauschen einzelner Fasern unterdrückt werden.

2.9. Crystal-Barrel

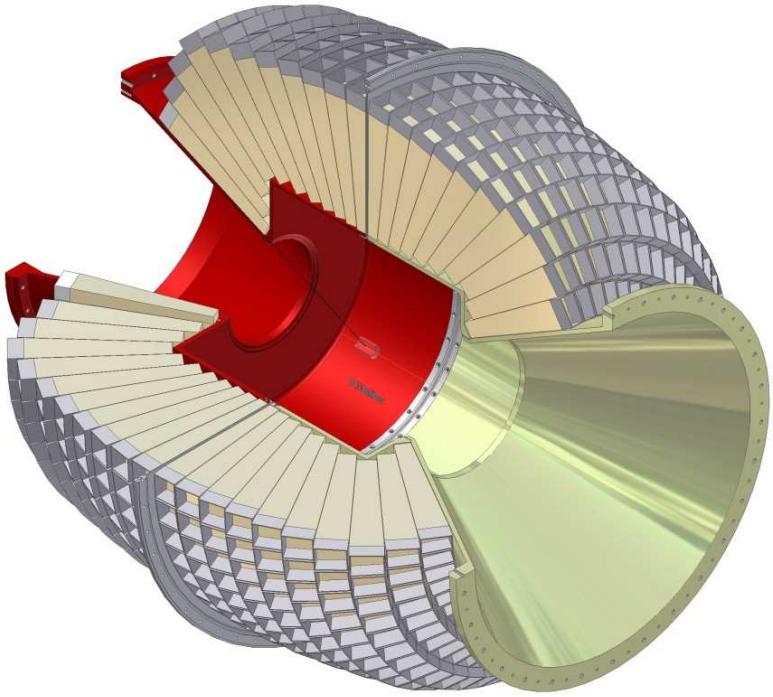


Abbildung 2.10.: **Schematische Darstellung des Crystal-Barrel-Detektors.** Dargestellt sind die 1230 Kristalle des Crystal-Barrel-Kalorimeters angeordnet in Fassform mit ihrer Haltestruktur [Wal18].

Das Crystal-Barrel-Kalorimeter (Abbildung 2.10) ist der zentrale Bestandteil des

Experiments. Es wird zusammen mit dem Forwardplug (2.10) und dem MiniTAPS Detektor (2.12) dafür verwendet die Energie der Zerfallsteilchen zu bestimmen. Das Kalorimeter besteht aus 1230 Cäsium-Jodid-Kristallen, welche mit Thallium dotiert[7] wurden. Diese sind in 21 Ringen angeordnet, wobei 20 Ringe aus 60 Kristallen bestehen und ein Ring aus 30 Kristallen. Vom Target aus gesehen wurde die Form der Kristalle so gewählt, dass bei den Ringen, welche aus 60 Kristallen bestehen, ein Winkel von 6° mal 6° abgedeckt wird. Für den einzelnen Ring, welcher aus 30 Kristallen besteht, decken die Kristalle einen Winkel von 12° mal 6° ab. Insgesamt deckt das Crystal-Barrel-Kalorimeter damit bei voller azimutaler Abdeckung den Polarwinkelbereich von 30° bis 156° ab. Zusammen mit dem Forwardplug und dem MiniTAPS Detektor welche die Teilchen detektieren, welche im Winkel von unter 30° emittiert werden, wird eine Raumabdeckung von fast 4π erreicht. Jeder der Kristalle hat eine Länge von 30 cm was in etwa 16 Strahlungslängen entspricht [Oli+14]. Dadurch werden bei einem 2 GeV Photon über 99% der Energie im Detektor deponiert [Blu+86]. Die Photonen erzeugen dabei einen elektromagnetischen Schauer. Dieser dehnt sich in transversaler Richtung über mehrere Kristalle aus. Dadurch kann mit einer Energiewichtung in der Rekonstruktion eine Winkelauflösung von besser als den 6° eines einzelnen Kristalls erreicht werden. Durch die Form der Kristalle sind diese in einer Fassform angeordnet, was dem Crystal-Barrel-Detektor seinen Namen gibt.

Das in den Kristallen entstehende Licht wird mit einem Wellenlängenschieber angepasst und dann von PIN-Photodioden in ein Stromsignal umgewandelt. Dieses wird durch einen Vorverstärker direkt am Kristall aufbereitet und dann über ein Verzögerungskabel an ADCs geleitet. Für jeden Kristall stehen dabei zwei Kanäle zur Verfügung. Der zweite Kanal ist dabei mit einem festgelegten Faktor abgeschwächt. Dadurch können auf dem einen Kanal niedrige Energien genau vermessen werden. Durch den anderen Kanal kann dann der Energiebereich ausgedehnt werden. Somit können sowohl kleine als auch große Energien gemessen werden. Ein Zeitsignal ist nicht sinnvoll zu gewinnen, da das Ausgangssignal nach dem Vorverstärker eine zu lange Anstiegszeit hat. Aus diesem Grund können die Signale des Crystal-Barrels auch nicht in der ersten Triggerstufe benutzt werden, es gibt jedoch eine zweite Stufe, welche das Signal getrennt auswertet und noch in die Triggerentscheidung einfließen lässt (siehe Kapitel 3.2.2).

2.10. Forwardplug

Der Forwardplug ist eine Erweiterung des Crystal-Barrel-Kalorimeters. Er besteht aus den gleichen mit Thallium dotierten Cäsium-Jodid-Kristallen, welche in drei Ringen mit

[7] In der Kristallstruktur mit Fremdatomen angereichert um freie Ladungsträger oder Löcher zu erzeugen.

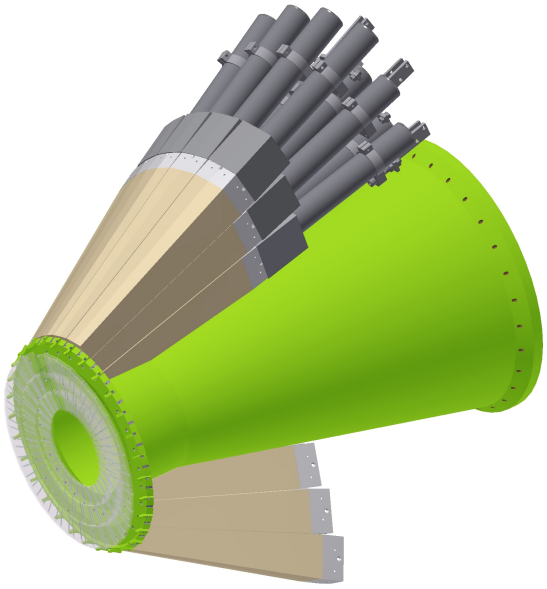


Abbildung 2.11.: **Schematische Darstellung des Forwardplug Detektors.** Gezeigt sind die 90 CsI(Tl) Kristalle des Forwardplugs aufgebaut in drei Ringen mit ihren Photomultipliern (schwarz) sowie der Haltestruktur (grün). Vor den Kristallen befinden sich zusätzlich die szintillierenden Plättchen des Forward-Veto Detektors (durchsichtig) [Wal18].

jeweils 30 Kristallen angeordnet sind. Die Kristalle decken dabei vom Target aus gesehen jeweils einen Winkel von 6° mal 12° ab. Dadurch können mit Hilfe des Forwardplugs die Energie der Zerfallsteilchen im Polarwinkelbereich von 12° bis 30° bestimmt werden. In einem Experiment mit feststehenden Target treten in diesem Winkelbereich durch die Impulserhaltung deutlich höhere Raten an Zerfallsteilchen auf. Deswegen wird das Licht der Kristalle nicht wie im Crystal-Barrel-Kalorimeter mit PIN-Photodioden ausgelesen sondern mit Photomultipliern. Photomultiplier haben eine höhere Verstärkung bei zugleich kleinerem Rauschen und können daher besser pulsgeformt werden. Dadurch sind die zu digitalisierenden Pulse schmaler und es können höhere Raten gemessen werden, ohne dass sich aufeinander folgende Signale überlagern. Zusätzlich kann das Signal dadurch mit TDCs ausgewertet werden. Außerdem kann mit einer Triggerlogik (siehe Kapitel 3.5) die Anzahl der Treffer im Forwardplug bestimmt werden. Die Signale dieser Logik sind schnell genug um in der ersten Triggerstufe benutzt zu werden.

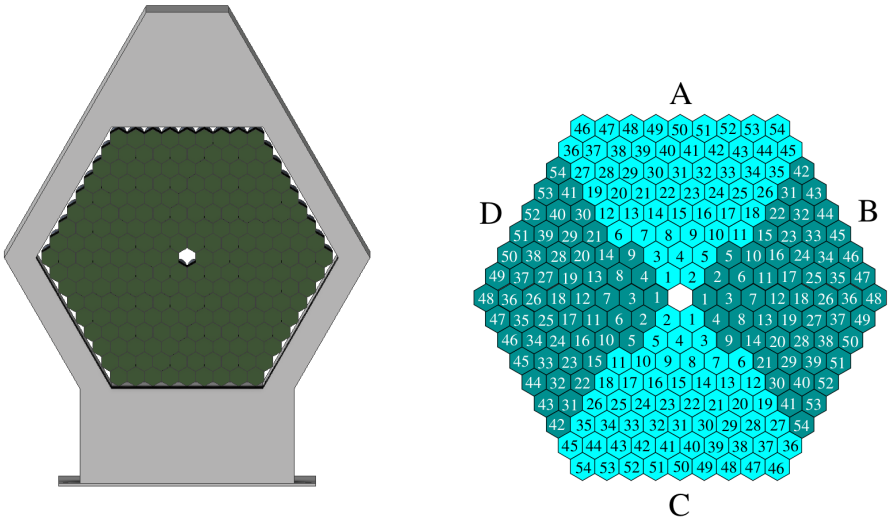
2.11. Forward-Veto

Direkt vor dem Forwardplug befindet sich der Forward-Veto-Detektor. Mit ihm wird bestimmt ob die Teilchen, welche im Forwardplug Energie deponieren, geladen sind. Der Forward-Veto-Detektor besteht aus 180 Plättchen aus szintillierendem Plastik. Die Plättchen sind dabei in drei Ringen vor den drei Kristallringen des Forwardplugs montiert. Dabei befindet sich jeweils ein Plättchen direkt vor jedem Kristall. Zusätzlich befindet sich eine zweite Ebene vor diesen Plättchen, welche um eine halbe Plättchenbreite verschoben ist. Geladene Teilchen welche in die Kristalle des Forwardplugs gelangen, werden dadurch immer von zwei Plättchen registriert. Durch Koinzidenz von zwei Plättchen kann dadurch zum Einen das Rauschen der Plättchen unterdrückt werden. Zum Anderen kann die Ortsauflösung verbessert werden. Der Aufbau des Forward-Veto-Detektors am Forwardplug ist schematisch in Abbildung 2.11 gezeigt.

Das Licht, dass in den Kristallen entsteht, wird mittels lichtleitender Fasern auf 16-fach Photomultiplier geführt. Die Zeitinformation der elektrisches Ausgangssignal wird dann von TDCs digitalisiert. Zusätzlich wird ein Oder aller Koinzidenzen hintereinander liegender Plättchen an die erste Triggerstufe geleitet.

2.12. MiniTAPS

Der MiniTAPS-Detektor ist für die am meisten nach vorne gerichteten Zerfallsteilchen ausgelegt. Er deckt den Polarwinkelbereich von 1° bis 12° ab und besteht aus 216 Barium-Fluorid-Kristallen. Die Kristalle haben eine sechseckige Form und sind in Form einer Wand angeordnet. Abbildung 2.12(a) zeigt diese Wand aus Sicht des Targets. Das Licht der Kristalle wird analog zum Forwardplug mit Photomultipliern in elektrische Signale gewandelt. Vor den Kristallen sind ähnlich wie beim Forwardplug zusätzlich szintillierende Plastikplatten angebracht, um geladene Zerfallsteilchen identifizieren zu können. Die Energie- und Zeitinformation der Signale der Kristalle und Plättchen wird mit speziell dafür entwickelten Modulen (CAEN V872B) digitalisiert. Die Kristalle sind dabei in 4 Sektoren A-D aufgeteilt (siehe Abbildung 2.12(b)). Für jeden dieser Sektoren wird nun ein Oder der Signale der einzelnen Kristalle gebildet. Aus diesen Oder-Signalen werden dann zwei Signale für die erste Triggerstufe generiert. Das erste ist ein simples Oder der 4 Sektorsignale. Auf der zweiten Triggerleitung wird nur etwas ausgegeben, wenn gleichzeitig zwei Sektoren getroffen werden.



(a) Aufbau des MiniTAPS Detektors mit seiner Haltestruktur. In Grün gezeigt sind die Kristalle. Die davor hängenden Veto-Plättchen sind nicht gezeigt[Wal18].

(b) Aufteilung des MiniTAPS Detektors in vier Sektoren mit 54 Kristallen [Mak11].

Abbildung 2.12.: **Schematische Darstellung des MiniTAPS Detektors.** Dargestellt sind die 216 Kristalle des MiniTAPS Detektors mit Blick vom Target ausgehend.

2.13. Cherenkov

Eine der häufigsten auftretenden Untergrundreaktionen im Target ist die Paarproduktion von Elektronen und Positronen. Aufgrund der in Strahlrichtung gerichteten kinetischen Energie des Elektronenstrahls werden die Elektronen und Positronen bevorzugt nach vorne gerichtet abgestrahlt und treffen den MiniTAPS Detektor. Diese Ereignisse weisen mit zwei Teilchen im MiniTAPS Detektor die gleiche Signatur wie zum Beispiel die Reaktion $\gamma p \rightarrow p\pi^0$ auf. Durch die Identifikation von Elektronen und Positronen im Cherenkov Detektor, können die Reaktionsteilchen der Paarproduktion unterschieden werden. Der Cherenkov-Detektor befindet sich vom Target aus gesehen vor dem MiniTAPS Detektor und deckt dessen gesamten Winkelbereich ab. Dadurch können Elektronen und Positronen welche den MiniTAPS-Detektor treffen mit dem Cherenkov-Detektor identifiziert werden. Der Cherenkov Detektor besteht dazu aus einer großen Kammer mit CO_2 Gas. In diesem Gas erzeugen Elektronen und Positronen Kegel von Cherenkov-Licht. Dieses wird dann von einer Spiegelkonstruktion auf einen Photomultiplier gerichtet, der diese in ein elektrisches Signal umwandelt. Die Zeitinformation dieses Signals wird dann von einem TDC digitalisiert. Zusätzlich wird ein Signal für die erste Triggerstufe gebildet,

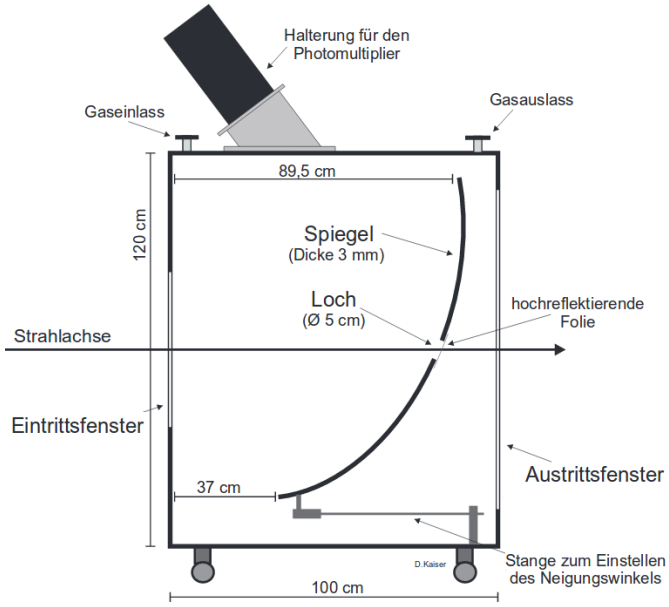


Abbildung 2.13.: **Schematische Darstellung des Cherenkov Detektors.** Gezeigt ist der mit CO_2 gefüllte Kasten des Cherenkov-Detektor in dem sich der Spiegel befindet. Der Spiegel leitet das Licht der entstehenden Cherenkov-Strahlung auf den Photomultiplier am oberen Ende [Kai07].

mit dem der Trigger blockiert werden kann, wenn ein Teilchen im Cherenkov detektiert wird. Der Cherenkov Detektor ist schematisch in Abbildung 2.13 dargestellt.

2.14. GIM

Am Ende der Photonstrahlführung steht der Gamma-Intensität-Monitor (GIM). Er überwacht die Intensität des Photonenstrahls der auf das Target trifft. Durch Vergleich mit den Raten im Tagger lässt sich feststellen wie viele Photonen zwischen dem Tagger und dem Target verloren gehen. Dies ist eine Voraussetzung für die Bestimmung von absoluten Wirkungsquerschnitten. Der GIM Detektor besteht aus einer 4x4 Matrix aus Bleiflouridkristallen, welche über Photomultiplier ausgelesen werden. Der GIM Detektor ist schematisch in Abbildung 2.14 dargestellt.

2.15. Flumo

Da der GIM Detektor (Kapitel 2.14) durch den Primärphotonstrahl sehr hohen Raten ausgesetzt ist, kommt es bei hohen Photonströmen zu Sättigungseffekten, welche die

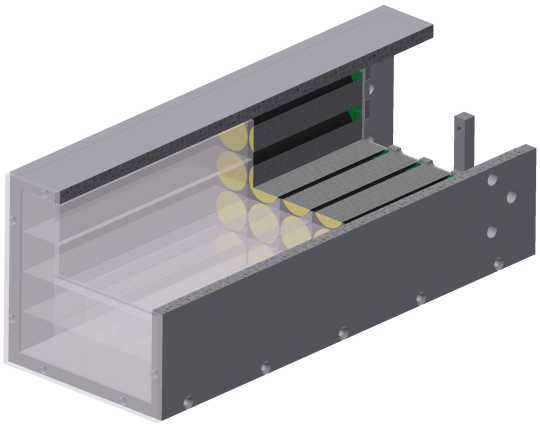


Abbildung 2.14.: **Schematische Darstellung des GIM Detektors.** Gezeigt sind die 16 Bleiflouridkristalle des GIM Detektors in einer 4x4 Matrix auf der linken Seite. Auf der rechten Seite in Schwarz dargestellt sind die 16 Photomultiplier [Wal18].

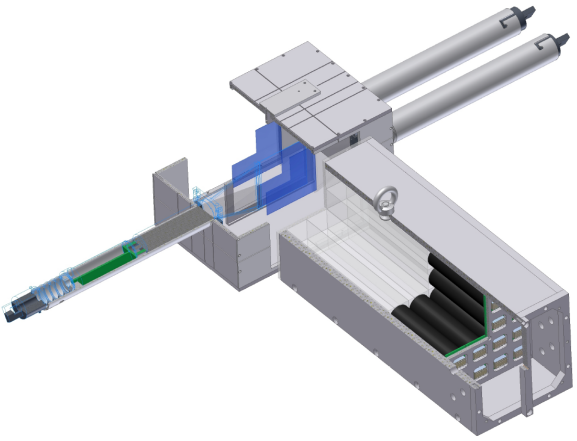


Abbildung 2.15.: **Schematische Darstellung des Flumo Detektors.** Links oberhalb des GIM Detektors sind die szintillierenden Platten (blau) des Flumo Detektors mit ihrer Halterung (grau) gezeigt [Wal18].

Bestimmung des Photonenflusses verfälscht. Um dem entgegenzuwirken wurde der Flussmonitor (Flumo) entwickelt [Die08]. Der Flumo-Detektor basiert auf dem konstanten Wirkungsquerschnitt der Paarbildung. Er besteht aus drei szintillierenden Platten, welche im Photonenstrahl stehen. Zwischen der ersten und der zweiten Platte ist dabei eine Kupferplatte. Der Photonenstrahl selber erzeugt in den szintillierenden Platten kein

Signal. Er erzeugt jedoch in der Kupferplatte durch den Effekt der Paarbildung mit einer konstanten Untersetzung Elektron-Positron-Paare, welche dann in den dahinter liegenden Platten ein Signal erzeugen. Die vor der Kupferplatte liegende szintillierende Platte dient dabei als Veto um mit dem Strahl kommende geladene Teilchen herauszufiltern. Gezählt wird dann jedes mal, wenn die Platte 1 kein Signal aussendet, aber die Platten 2 und 3. Durch den Wirkungsquerschnitt der Paarbildung liegt die Detektionsrate des Flumo Detektors bei etwa 5% [Die08]. Dadurch sind die Raten des Flumo Detektors niedriger als im GIM Detektor und es treten bei den im Experiment verwendeten Photonenströmen keine Sättigungseffekte auf. Kalibriert wird der Detektor bei niedrigen Photonenströmen mit dem GIM Detektor. Schematisch ist der Flumo Detektor zusammen mit dem GIM in Abbildung 2.15 gezeigt.

2.16. Lichtpulser

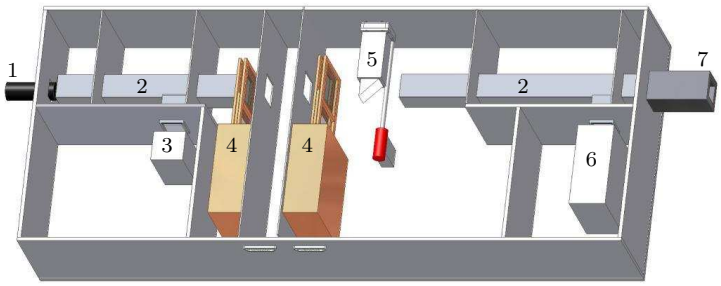


Abbildung 2.16.: **Schematische Darstellung des Lichtpulseraufbaus.** Auf der linken Seite wird durch eine Xenon-Lampe (1) ein Lichtblitz erzeugt. Dieser wird durch einen Lichtmischer (2) und sechs verfahrbare Filter (4) geleitet. Dann kann das Licht entweder an den Forwardplug ausgekoppelt werden (5) oder über einen weiteren Lichtmischer (2) an den Crystal-Barrel-Detektor ausgekoppelt werden (7). Zusätzlich stehen zur Überprüfung der Stabilität noch ein Photodetektor (3) und ein Eichdetektor (6) zur Verfügung. Bild modifiziert aus [Bös06].

Zur Eichung des Crystal-Barrel-Detektors (Kapitel 2.9) gibt es ein System, welches Lichtblitze in die Kristalle einkoppelt (siehe Abbildung 2.16). Dadurch kann zum Einen die Stabilität der Lichtsammlung und damit die Stabilität des Energiesignals überprüft werden. Zum Anderen können damit die beiden Energiebereiche der ADCs aufeinander abgestimmt werden. Dazu wird zunächst ein Lichtblitz mit einer Xenon-Lampe erzeugt. Dieser wird dann über einen Lichtmischer und lichtleitende Fasern an die Kristalle geleitet. Zusätzlich können mechanisch Kombinationen lichtabschwächender Filter in

den Lichtstrahl gefahren werden. Dadurch kann eine Vielzahl unterschiedlicher Lichtintensitäten zu den Kristallen geleitet werden. Die verschiedenen Intensitäten simulieren dabei unterschiedliche Energiedepositionen in den Kristallen. Die Lampen und die Filter befinden sich in zwei Aufbauten, jeweils für eine Hälfte des Crystal-Barrel-Detektors. In dem Aufbau der strahlabwärts gelegenen Hälfte befindet sich zusätzlich noch ein Aufbau zur Beleuchtung der Kristalle des Forwardplugs. Ob der Forwardplug oder die Crystal-Barrel-Hälfte beleuchtet wird, selektiert eine drehbare Lichtauskopplung. Die Lichtstärke der Xenon-Lampe wird mit einem ADC überprüft. Die Filterstellungen werden mit einem spezielle CAMAC-Modul selektiert. Diese Modul steuert auch die Xenon-Lampe und selektiert das Blitzen des Forwardplugs. Zusätzlich wird ein Signal für die Auslese generiert, dass an den Trigger geleitet wird. Der Aufbau des Lichtpulsers ist genauer in [Bös06] beschrieben.

3. Trigger

Beim Abspeichern der Detektorinformationen von im Experiment gesehenen Ereignissen entsteht immer eine Totzeit. Während dieser Zeit ist das Detektorsystem nicht in der Lage neue Ereignisse aufzunehmen. Treten während dieser Totzeit Ereignisse auf, gehen sie also verloren. Es ist daher ungünstig das Speichern der Informationen zu starten, wenn kein sinnvolles oder gewünschtes Ereignis stattgefunden hat. Für das Messprogramm nicht sinnvolle Ereignisse sind zunächst einmal Untergrundereignisse, welche nicht aus dem Zerfall angeregter Nukleonen entstehen. Die im Radiatortarget entstehenden Gammaquanten können zum Beispiel durch Paarproduktion e^+e^--Paare erzeugen, welche dann Signale in den Detektoren produzieren. Zusätzlich kann es sinnvoll sein sich auf einen Zerfallskanal wie zum Beispiel $\gamma p \to p\eta$ zu beschränken. Aufgrund des Isospins des η-Mesons können Δ-Resonanzen nicht in diesen Kanal zerfallen. Dadurch entsteht ein saubereres Spektrum von weniger Resonanzen. Wie in Abbildung 3.1 zu sehen ist, ist jedoch der Wirkungsquerschnitt des Zerfallskanals $\gamma p \to p\pi^0$ im, vom Crystal-Barrel/TAPS-Experiment untersuchten Energiebereich, deutlich größer als der vom Kanal $\gamma p \to p\eta$. Dadurch würde in einer ungefilterten Datennahme hauptsächlich der Kanal $\gamma p \to p\pi^0$ gemessen und ein großer Teil der entstehenden η-Mesonen würden durch die Totzeit beim Abspeichern der Ereignisse verloren gehen. Es wäre daher nützlich, wenn π^0-Ereignisse nicht abgespeichert würden wodurch mehr η-Mesonen gemessen werden könnten. Um diese beiden Reaktionskanäle zu unterscheiden ist es notwendig sich den Zerfall der π^0- und η-Mesonen anzuschauen. Das π^0-Meson zerfällt zu (98,823 ± 0,034) % in 2γ [Oli+14]. Das η-Meson zerfällt zwar auch zu (39,41 ± 0,20) % in 2γ, zusätzlich kann es jedoch auch mit einer Wahrscheinlichkeit von (32,68 ± 0,23) % in $3\pi^0$ zerfallen, welche dann insgesamt in 6γ zerfallen. Wenn man also nur die Detektorinformationen speichert, wenn ein Proton und 3 oder mehr Photonen gesehen wurde, werden keine Ereignisse aus dem Kanal $\gamma p \to p\pi^0$ erfasst, aber zu mindestens der in $3\pi^0$ zerfallende Teil der Ereignisse aus dem Kanal $\gamma p \to p\eta$. Um dies zu erreichen, braucht man eine Komponente, welche anhand bestimmter Kriterien beurteilt ob das auftretende Ereignis gespeichert werden soll. Diese Komponente ist die Triggerelektronik. Das grundsätzliche Prinzip eines Triggers ist unabhängig von der jeweiligen Implementation. Das Triggersystem sammelt Informationen aus den Detektoren und vergleicht diese mit vorher festgelegten Kriterien. Erfüllt ein Ereignis diese Kriterien, so wird vom Triggersystem das Abspeichern des

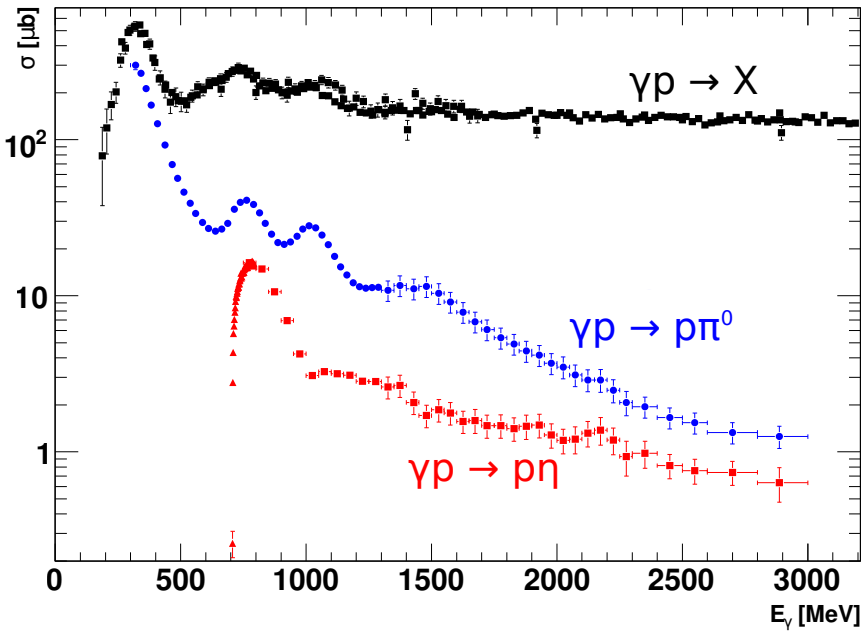


Abbildung 3.1.: **Wirkungsquerschnitte in der Photoproduktion am Proton.** Aufgetragen ist der Absorptionsquerschnitt für die Photoproduktion am Proton [Nak+10] als Funktion der Photonenergie in schwarz. Zusätzlich sind die totalen Wirkungsquerschnitte der Photoproduktion von π^0-Mesonen [Pee+07] in blau und η-Mesonen [Bar04; Kru+95] in rot aufgetragen.

Ereignisses angewiesen. Sind die Kriterien nicht erfüllt, wird auf das nächste Ereignis gewartet.

Bei der Implementation eines Triggersystems ist es wichtig zu betrachten, wie lange nach dem Ereignis die Ausleseelektronik die Entscheidung des Triggersystems erhält. Dies ist notwendig, da die Ausleseelektronik die Daten des Ereignisses meist nur eine begrenzte Zeit vorhalten kann, bevor sie in den Modulen zwischengespeichert werden müssen. Wenn die Informationen bis zu diesem Zeitpunkt noch nicht zwischengespeichert sind, gehen sie verloren. Beim Crystal-Barrel/TAPS-Experiment ist diese Zeit bei der Elektronik des MiniTAPS-Detektors am kürzesten. Die dort verwendete Elektronik benötigt die Entscheidung des Triggersystems spätestens 750 ns nach dem Ereignis [CAE04]. Daraus lässt sich die Zeit bestimmen, nach der alle Signale am Triggersystem ankommen müssen, um für die Triggerentscheidung berücksichtigt werden zu können. Dazu muss zum Einen noch die Signallaufzeit vom Triggersystem zu der Elektronik des MiniTAPS-Detektors von 150 ns beachtet werden. Des Weiteren ist die für die Triggerentscheidung benötigte Zeit von 100 ns zu berücksichtigen. Als letztes werden in der späteren Analyse noch Daten bis zu 50 ns nach dem Ereignis benötigt. Aus diesen Werten ergibt sich eine Zeit

von 450 ns nach der spätestens alle Signale am Triggersystem ankommen müssen, um in der Triggerentscheidung berücksichtigt werden zu können.

Im derzeitigen Experimentaufbau erreichen alle Signale bis auf das des Crystal-Barrel-Detektors das Triggersystem innerhalb von 300 ns. Für die Bestimmung der Anzahl der Treffer im Crystal-Barrel-Detektor ist der FACE[1] zuständig (siehe Abschnitt 3.4). Der FACE benötigt für seine Entscheidung im Durchschnitt 6 µs, was deutlich über der maximalen Zeit von 450 ns liegt. Da die Anzahl der Treffer im Crystal-Barrel-Detektor jedoch essentiell für eine Entscheidung über gewünschte Ereignisse ist, muss die Triggerentscheidung in zwei Stufen erfolgen. Die erste Stufe untersucht alle innerhalb der geforderten 450 ns ankommenden Signale. Falls diese erste Stufe entscheidet, dass es sich um ein, den Kriterien entsprechendes Ereignis handelt, wird ein *Timeref*-Signal an die Ausleseelektronik gesendet, alle vorhandenen Informationen zwischenzuspeichern. Zusätzlich wird der FACE angewiesen die Anzahl der Treffer im Crystal-Barrel-Detektor zu bestimmen. Das Ergebnis sendet der FACE dann an die zweite Trigger-Stufe, welche dies mit den in der ersten Stufe erhaltenen Informationen zusammen auswertet und eine endgültige Entscheidung trifft ob das Ereignis gespeichert werden soll. Wenn die zweite Stufe eine positive Entscheidung trifft wird ein *Event*-Signal an die Elektronik der Subdetektoren gesendet, mit dem das Auslesen und Speichern der Informationen angewiesen wird. Bei einer negativen Entscheidung wird auf das nächste den Kriterien entsprechende Ereignis in der ersten Stufe gewartet und die Ausleseelektronik der Detektoren werden mit einem *Sysreset*-Signal angewiesen die zwischengespeicherten Informationen zu löschen und sich auf das nächste Ereignis vorzubereiten. Bei einer negativen Triggerentscheidung fällt also nur die Totzeit für die FACE-Entscheidung von im Durchschnitt 6 µs an, anstelle der für das komplette Abspeichern des Ereignisses benötigten Zeit von in der Größenordnung 500 µs. Wenn negative Entscheidungen also in vergleichbarer oder sogar höherer Zahl als positive auftreten, wird die entstehende Totzeit, während der gewünschte Ereignisse nicht detektiert werden können, deutlich reduziert. Das hier beschriebene System ist schematisch in Abbildung 3.2 gezeigt. Der zeitliche Ablauf in der ersten und zweiten Stufe ist in Anhang B.1 und Anhang B.2 gezeigt.

In den folgenden Abschnitten werden zunächst einige vom Crystal-Barrel/TAPS-Experiment hauptsächlich benutzen Triggerbedingungen erklärt. Anschließend werden die verschiedenen vom Crystal-Barrel/TAPS-Experiment benutzen Implementationen der beiden Triggerkomponenten beschrieben. Abschließend wird noch die spezielle Elektronik beschrieben, mit der die Triggersignale des Crystal-Barrel-Detektors (Kapitel 3.4) sowie des Vorwärtsdetektors (Kapitel 3.5) erzeugt werden.

[1] Fast Cluster Encoder - Logik zur Bestimmung der Anzahl zusammenhängender Gebiete.

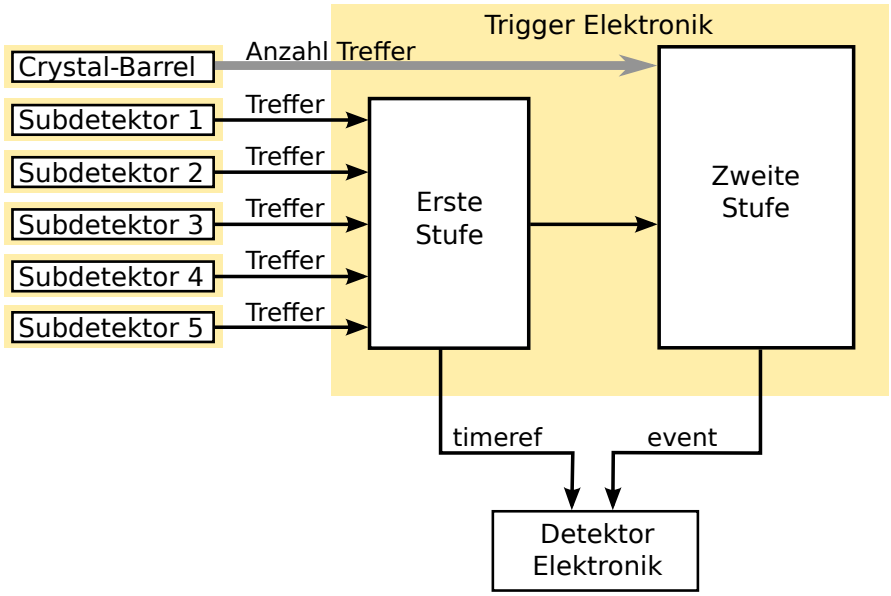


Abbildung 3.2.: **Überblick über den Trigger des Crystal-Barrel/TAPS-Experiments.** Alle Subdetektoren bis auf den Crystal-Barrel-Detektor liefern im Falle eines Treffers ein Signal an die erste Triggerstufe. Falls diese eine positive Entscheidung trifft, wird ein Signal an die zweite Stufe und ein Signal (*Timeref*) an die Subdetektoren gesendet. Die zweite Stufe wertet dann zusätzlich das Signal des Crystal-Barrel-Detektors aus und sendet im Falle einer positiven Entscheidung ebenfalls ein Signal an die Subdetektoren (*Event*).

3.1. Triggerbedingungen im Crystal-Barrel/TAPS-Experiment

Um entscheiden zu können ob es sich bei dem vorliegenden Ereignis um ein gewünschtes Ereignis handelt, stehen der Triggerelektronik des Crystal-Barrel/TAPS-Experiments 16 Signale zu Verfügung (siehe Anhang B.1). Die Eingangssignale lassen sich dabei in verschiedene Gruppen aufteilen. Zunächst einmal gibt es normale Triggersignale. Anhand dieser Signale kann eine Multiplizität der Zerfallsprodukte gefordert werden. Im Crystal-Barrel/TAPS-Experiment sind das die Signale der Detektoren für neutrale Teilchen (Crystal-Barrel, Vorwärtsdetektor, MiniTAPS). Die zweite Gruppe von Triggersignalen sind Veto-Signale. Wenn in diesen Detektoren ein Signal gesehen wird, ist das ein Indiz dafür, dass ein unerwünschtes Ereignis stattgefunden hat. Im Crystal-Barrel/TAPS-Experiment ist dies nur der Cherenkov-Detektor. Die nächste Gruppe von Signalen werden von den Detektoren für geladene Teilchen gesendet. Sie können zum Beispiel dafür benutzt werden zu fordern, dass eines der gesehenen Teilchen geladen ist, um zum Beispiel sicherzustellen, dass ein Proton gesehen wurde. Im Crystal-Barrel/TAPS-Experiment sind dies der Vorwärtsvetodetektor, der Innendetektor und der MiniTAPS-Veto-Detektor. Das letzte Signal, welches noch für einen Experimenttrigger relevant ist ist das Triggersignal des Taggers. Wenn dieses Signal gefordert wird, ist sichergestellt, dass das Photon welches die Reaktion ausgelöst hat getaggt wurde. Wenn das Photon nicht energiemarkiert (getaggt) wurde ist die Energie des Photons nicht bekannt und die Energie des vom Photon angeregten Zustandes kann nicht direkt bestimmt werden. Da die Bestimmung über indirekte Methoden aus den Zerfallsprodukten ungenauer ist, ist es wünschenswert, dass bei jedem Ereignis auch ein Triggersignal vom Tagger gesendet wurde. Aus diesen Signalen können nun die Triggerbedingungen zusammengesetzt werden. Die Ausgangsprodukte der meisten Reaktionen am Proton sind das Proton und mindestens zwei weitere ungeladene Teilchen. Wenn alle Ausgangsprodukte dieser Reaktionen vollständig gesehen wurden, müssen also mindestens 3 Teilchen gesehen worden sein, wobei eines davon geladen sein muss. Das Proton kann jedoch, wenn es nur eine kleine Energie hat, eventuell nicht detektiert werden. Da sein Viererimpuls jedoch aus den anderen Zerfallsprodukten rekonstruiert werden kann, ist es trotzdem sinnvoll Ereignisse, in denen das Proton nicht detektiert wurde, zu speichern. Daher reicht es aus 2 neutrale Teilchen zu fordern. Diese neutralen Teilchen werden abhängig von ihrem Winkel im MiniTAPS-, Vorwärts- oder Crystal-Barrel-Detektor gemessen. Daher ergeben sich die möglichen Verteilungen:

- 2 Teilchen im MiniTAPS-Detektor
- 1 Teilchen im MiniTAPS-Detektor + 1 Teilchen im Vorwärtsdetektor
- 1 Teilchen im MiniTAPS-Detektor + 1 Teilchen im Crystal-Barrel-Detektor

- 2 Teilchen im Vorwärtsdetektor
- 1 Teilchen im Vorwärtsdetektor + 1 Teilchen im Crystal-Barrel-Detektor
- 2 Teilchen im Crystal-Barrel-Detektor

Die letzte Bedingung (2 Teilchen im Crystal-Barrel-Detektor) hat dabei das Problem, dass für diese Bedingung nur Detektoren ausschlaggebend sind, die nicht in der ersten Stufe des Triggers ausgewertet werden können. Da die zweite Stufe aber immer erst von der ersten Stufe aktiviert werden muss, kann diese Bedingung so nicht realisiert werden. Um trotzdem Ereignisse aufnehmen zu können, aus denen keine Zerfallsprodukte in Richtung des Vorwärtsdetektors oder des MiniTAPS kommen, kann stattdessen der Innendetektor in Verbindung mit zwei Teilchen im Crystal-Barrel-Detektor gefordert werden. Dadurch werden zu mindestens solche Reaktionen erfasst, bei denen das Proton im Innendetektor ein Signal erzeugt hat. Dadurch entstehen die in Tabelle 3.1 aufgeführten Bedingungen (aufgeteilt auf die erste und zweite Stufe).

#	erste Stufe	zweite Stufe
1	2 in MT	
2	1 in MT & 1 in VD	
3	1 in MT	1 in CB
4	2 in VD	
5	1 in VD	1 in CB
6	1 in IN	2 in CB

Tabelle 3.1.: **Triggerbedingungen für 2 Teilchen.** Aufgeführt sind die Triggerbedingungen für einen einfachen Zweiteilchentrigger. MT = MiniTAPS, VD = Vorwärtsdetektor, CB = Crystal-Barrel, IN = Innendetektor.

Nun kann es sinnvoll sein zu fordern, dass auch der Tagger ein Signal gesehen hat um sicherzustellen das die Energie des Photons bekannt ist. Wenn man diese Bedingung hinzunimmt, entsteht die Bedingung aus Tabelle 3.2.

Da in Vorwärtsrichtung relativ viele e^+e^--Paare entstehen, ergibt sich bei diesen Triggerbedingungen jedoch ein Problem. Die e^+e^--Paare gehören nicht zu gewünschten Reaktionen, können jedoch trotzdem Treffer in den Detektoren erzeugen. Durch das feste Target im Experiment, sind diese e^+e^--Paare dabei besonders häufig in Vorwärtsrichtung zu finden. Dadurch ist der MiniTAPS-Detektor hiervon am stärksten betroffen, da dieser den niedrigsten Winkelbereich in Vorwärtsrichtung abdeckt. Somit sind die Triggerbedingungen 1 und 3 in Tabelle 3.2 besonders betroffen. Die Triggerbedingung 1 fordert nur 2 Teilchen im MiniTAPS-Detektor, würde also von einem e^+e^--Paar vollständig erfüllt. Die Triggerbedingung 3 fordert zwar einen Treffer im Crystal-Barrel, würde also von

#	erste Stufe	zweite Stufe
1	TAG & 2 in MT	
2	TAG & 1 in MT & 1 in VD	
3	TAG & 1 in MT	1 in CB
4	TAG & 2 in VD	
5	TAG & 1 in VD	1 in CB
6	TAG & 1 in IN	2 in CB

Tabelle 3.2.: **Triggerbedingungen für 2 Teilchen mit Tagger.** Aufgeführt sind die Triggerbedingungen für einen einfachen Zweiteilchentrigger mit der Forderung einer bestimmen Photonenergie durch den Tagger. MT = MiniTAPS, VD = Vorwärtsdetektor, CB = Crystal-Barrel, IN = Innendetektor, TAG = Tagger.

einem e^+e^--Paar nicht erfüllt. Die Bedingung des Treffers im Crystal-Barrel-Detektor wird jedoch erst in der zweiten Triggerstufe ausgewertet. Dadurch wird die erste Stufe weiterhin oft von e^+e^--Paaren ausgelöst und dadurch würde eine hohe Totzeit durch die häufige negative Entscheidung des FACE entstehen. Es gibt nun zwei Möglichkeiten dieses Problem zu lösen. Zum Einen können diese beiden Triggerbedingungen modifiziert werden. Bei der Triggerbedingung 3 kann analog zur Bedingung 6 noch ein Treffer im Innendetektor gefordert werden. Dadurch wird diese Bedingung nicht mehr von e^+e^--Paaren erfüllt, aber weiterhin von gewünschten Ereignissen, bei denen das Proton im Innendetektor ein Signal erzeugt hat. Die Triggerbedingung 1 müsste weggelassen werden. Daraus ergibt sich die als *Trig41* bezeichnete Triggerbedingung in Tabelle 3.3.

#	erste Stufe	zweite Stufe
1	TAG & 1 in MT & 1 in VD	
2	TAG & 1 in MT & 1 in IN	1 in CB
3	TAG & 2 in VD	
4	TAG & 1 in VD	1 in CB
5	TAG & 1 in IN	2 in CB

Tabelle 3.3.: **Triggerbedingungen Trig41.** Aufgeführt sind die Triggerbedingungen für einen einfachen Zweiteilchentrigger mit der Forderung einer durch den Tagger bestimmten Photonenergie und unter Beachtung von e^+e^--Paaren. MT = MiniTAPS, VD = Vorwärtsdetektor, CB = Crystal-Barrel, IN = Innendetektor, TAG = Tagger.

Alternativ steht noch ein Signal des Cherenkov-Detektors zur Verfügung. Der Cherenkov-Detektor befindet sich direkt vor dem MiniTAPS-Detektor und sendet nur dann ein

Signal wenn Elektronen oder Positronen den Detektor treffen. Wenn man nun zusätzlich zu den Bedingungen aus Tabelle 3.2 fordert, dass der Cherenkov-Detektor kein Signal gesendet hat, wird erreicht, dass e^+e^--Paare die Bedingungen der ersten Stufe nicht mehr erfüllen können. Die daraus resultierenden Bedingungen werden im Crystal-Barrel/TAPS-Experiment *Trig42c* genannt und sind in Tabelle 3.4 aufgeführt.

#	erste Stufe	zweite Stufe
1	TAG & !CHE & 2 in MT	
2	TAG & !CHE & 1 in MT & 1 in VD	
3	TAG & !CHE & 1 in MT	1 in CB
4	TAG & !CHE & 2 in VD	
5	TAG & !CHE & 1 in VD	1 in CB
6	TAG & !CHE & 1 in IN	2 in CB

Tabelle 3.4.: **Triggerbedingungen für 2 Teilchen mit Tagger und Cherenkov Veto (Trig42c).** Aufgeführt sind die Triggerbedingungen für einen einfachen Zweiteilchentrigger mit der Forderung einer durch den Tagger bestimmten Photonenergie und keinem Signal im Chrenkov Detektor. MT = MiniTAPS, VD = Vorwärtsdetektor, CB = Crystal-Barrel, IN = Innendetektor, TAG = Tagger, CHE = Cherenkov.

3.2. CAMAC-Trigger

Der CAMAC[2]-Trigger wurde als Trigger für das Crystal-Barrel/TAPS-Experiment in den Jahren 2007 bis 2012 eingesetzt. Er wurde im Rahmen einer Diplomarbeit von Alexander Winnebeck [Win06] entwickelt. Wie schon in der Einführung des Abschnitts 3 beschrieben ist es im Crystal-Barrel/TAPS-Experiment aufgrund der langsamen Entscheidungszeit des FACE von 6 µs notwendig den Trigger in zwei Stufen zu realisieren. Für den CAMAC-Trigger wurden dazu zwei speziell für diese Aufgabe entwickelte CAMAC-Module eingesetzt. Für die erste und zweite Stufe existiert dabei jeweils ein Modul. Auf diesen Modulen befinden sich FPGA[3]-Chips deren Verschaltung durch VHDL[4]-Programme beschrieben wird. In den nächsten beiden Kapiteln wird die Funktionalität dieser beiden Module beschrieben.

3.2.1. Die erste Triggerstufe

Die erste Triggerstufe verarbeitet die Triggersignale der Detektoren, welche ihr Triggersignal innerhalb von 450 ns senden. Dazu hat das Modul der ersten Stufe 16 Eingänge auf denen Signale im ECL[6]-Standard empfangen werden. Die Belegung dieser 16 Eingänge im Crystal-Barrel/TAPS-Experiment sind in Anhang B.1 aufgeführt. Die Signale der 16 Eingänge werden an 16 festgelegte, voneinander unabhängige, Triggerblöcke verteilt. Jeder Triggerblock besitzt eine feste Triggerbedingung welche im Triggerblock überprüft wird. Diese Triggerbedingungen können sowohl das gleichzeitige Auftreten mehrerer Eingangssignale fordern, als auch fordern, dass eines oder mehrere Eingangsignale zeitgleich kein Signal zeigen (Veto). Wenn die geforderte Bedingung erfüllt ist, sendet der Triggerblock ein Signal aus. Welche Eingangsignale an die Triggerblöcke geleitet werden und mit welchen Bedingungen daraus das Ausgangssignal generiert wird ist dabei in der Programmierung der FPGA festgelegt und kann nur durch eine neue Programmierung der FPGA verändert werden. Es kann jedoch ohne neue Programmierung ausgewählt werden, welche der Logikblöcke derzeit benutzt werden. Da im Normalfall keine 16 parallelen

[2] Computer Automated Measurement And Control - Bussystem zur Kommunikation mit Elektronikmodulen.
[3] Field Programmable Gate Array - Bauteil in das logische Schaltungen programmiert werden können.
[4] VHSIC[5]Hardware Description Language - Beschreibungssprache mit der die interne Verschaltung in programmierbaren logischen Bausteinen beschrieben werden kann. Ursprünglich entwickelt im Rahmen des VHSIC-Programmes kann diese Sprache für die Entwicklung der meisten aktuellen programmierbaren Logikbausteinen, wie zum Beispiel FPGAs, CPLDs und ASICs verwendet werden.
[5] Very High Speed Integrated Circuit - Ein Programm des Verteidigungsministriums der USA in den 1980er Jahren zur Beschleunigung der Entwicklung von Bauteilen mit sehr schnellen integrierten Schaltungen. Im Rahmen dieses Programms entstand unter anderem die Beschreibungssprache VHDL.
[6] Abkürzung für MECL - Motorola-Emitter-coupled logic - Differenzieller Übertragungsstandard mit Pegeln zwischen −5 V und 0 V [Blo72]

Triggerbedingungen gefordert werden, können so parallel mehrere Triggerbedingungen vorbereitet werden. Zudem ist es einfach möglich mehrere geteste Programmierungen der FPGA vorzuhalten und diese während der Datennahme zu wechseln. Falls jedoch neue Triggerbedingungen gewünscht werden ist dies so einfach nicht möglich da dafür eine neue Programmierung erstellt und getestet werden muss. Da dies ein sehr zeitaufwendiger Prozess ist, wurde der CAMAC-Trigger im Jahr 2013 durch einen neu entwickelten Trigger ersetzt (siehe Kapitel 3.3).

Durch die erste Triggerstufe werden mehrere Ausgangssignale gesendet. Nachdem die erste Stufe angewiesen wird mit dem Überprüfen der Triggersignale zu beginnen, wird zunächst das *Sysreset*-Signal ausgesendet. Dieses Signal weist die Elektronik der Subdetektoren an, sich auf das nächste Ereignis vorzubereiten. 5 µs nach dem Aussenden dieses Signals beginnen die Triggerblöcke mit dem Auswerten der Triggersignale. Nachdem einer der Triggerblöcke seine geforderte Triggerbedingung gefunden hat, werden eine Reihe von Signalen ausgesendet. Zunächst einmal wird das *Timeref*-Signal an die zweite Stufe und an die Ausleseelektronik aller Subdetektoren gesendet. Dadurch wird diesen die positive Entscheidung der ersten Stufe mitgeteilt. Zusätzlich ist, in der Programmierung der FPGA, für jeden Triggerblock festgelegt, welches Signal zusätzlich an die zweite Stufe gesendet wird. Dies kann zum Einen das *Bypass*-Signal sein, welches signalisiert,

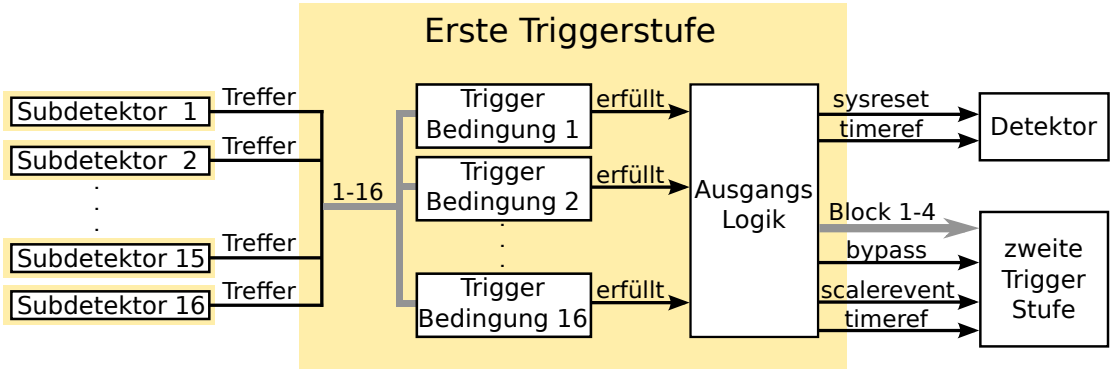


Abbildung 3.3.: **Überblick über die erste Triggerstufe des Crystal-Barrel/TAPS-Experiments.** Alle Subdetektoren bis auf den Crystal-Barrel-Detektor liefern im Falle eines Treffers ein Signal an die erste Triggerstufe. Die Triggerblöcke liefern sobald die jeweilige Triggerbedingung erfüllt ist ein Signal an die Ausgangslogik. Diese sendet dann die geforderte Trefferzahl der FACE (Block1-4 / *Bypass*), ein *Timeref*-Signal und gegebenenfalls ein *scaler_event*-Signal an die zweite Stufe. Das *Timeref*-Signal wird außerdem an die Elektronik der Subdetektoren verteilt. Bevor die erste Stufe mit dem Auswerten der Triggerbedingungen beginnt wird zusätzlich ein *Sysreset*-Signal ausgesendet.

dass keine zusätzliche Entscheidung der zweiten Stufe nötig ist. Wenn eine Entscheidung nötig ist, wird über eine von vier Leitungen signalisiert, welcher der Triggerblöcke der zweiten Stufe gefordert ist. Als letztes kann bei jeder Triggerbedingung noch zusätzlich ein *scaler_event*-Signal gesendet werden. Dieses Signal weist die zweite Stufe an eine andere *Event-ID* auszusenden. Dadurch werden die Subdetektoren angewiesen zusätzliche Informationen zu speichern. Alle Ausgangssignale der ersten Stufe werden als Signale im ECL-Standard ausgesendet.

3.2.2. Die zweite Triggerstufe

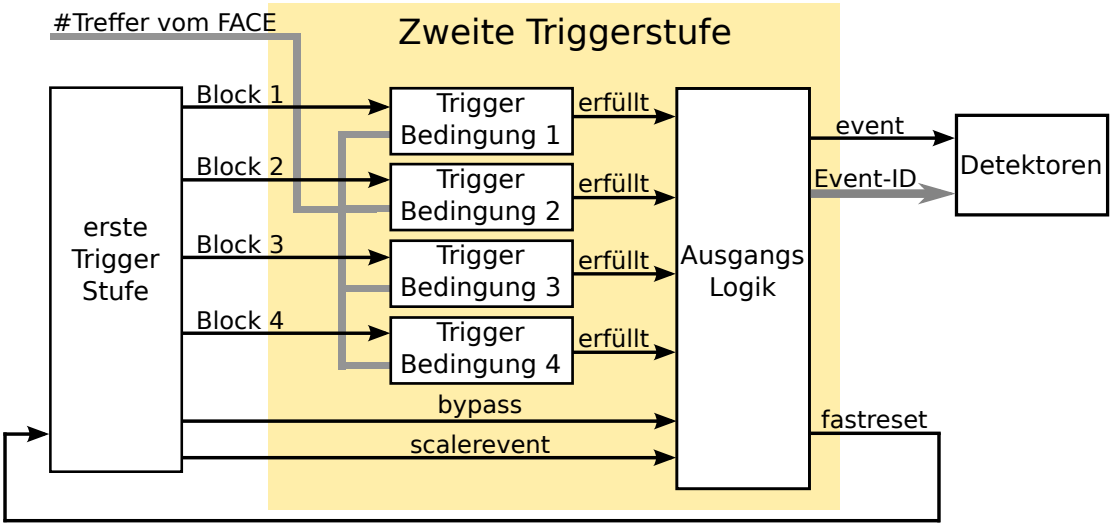


Abbildung 3.4.: **Überblick über die zweite Triggerstufe des Crystal-Barrel/TAPS-Experiments.** Die zweite Stufe erhält von der ersten Stufe die Anweisung welche Triggerbedingung gefordert ist oder ob gar keine Bedingung notwendig ist (*Bypass*). Zusätzlich werden spezielle Ereignisse noch mit einem *scaler_event* markiert. Wenn die ausgewählte Bedingung mit den vom FACE gelieferten Informationen erfüllt wird, wird von der jeweiligen Triggerblock ein Signal gesendet (*erfüllt*). Wenn die Ausgangslogik eines dieser Signale oder ein *Bypass*-Signal erhält, wird ein *Event*-Signal an die Detektoren gesendet. Ansonsten wird nach einer festgelegten Zeit ein *Fastreset*-Signal an die erste Stufe gesendet. Abhängig davon ob das *scaler_event*-Signal gesetzt ist, werden unterschiedliche *Event-ID*-Signale an die Detektoren gesendet.

Das Modul der zweiten Stufe ist bezüglich der Hardware identisch zum Modul der ersten Stufe aufgebaut. Es unterscheidet sich jedoch in der Programmierung der FPGA-Logik. Um die Entscheidung der zweiten Stufe treffen zu können enthält das Modul vier Triggerblöcke. An jeden dieser Triggerblöcke wird vom FACE die Anzahl der Treffer

im Crystal-Barrel-Detektor gesendet. Zusätzlich wird von der ersten Triggerstufe der geforderte Block durch ein Signal ausgewählt. Für jeden dieser vier Blöcke kann eine minimale und eine maximale Anzahl von Treffern im Crystal-Barrel-Detektor vorgegeben werden. Da die zweite Stufe deutlich weniger komplex ist als die erste Stufe, kann die Konfiguration dabei ohne Neuprogrammierung der FPGA verändert werden. Dadurch sind die Triggerbedingungen der zweiten Stufe auch während der Datennahme beliebig anpassbar. Durch die parallele Ansteuerung der vier Blöcke können in der zweiten Stufe gleichzeitig vier verschiedene Bereiche von Trefferzahlen im Crystal-Barrel-Detektor festgelegt werden, welche von der ersten Stufe gefordert werden können. Die Entscheidung über die Anzahl der Treffer im Crystal-Barrel-Detektor kann jedoch auch komplett übergangen werden. Dazu signalisiert die erste Stufe mit einem *Bypass*-Signal, dass nicht auf die Auswertung des FACE gewartet werden soll und direkt ein Ausgangssignal an die Detektoren gegeben werden soll. Dadurch wird bei Entscheidungen, die keine Treffer im Crystal-Barrel-Detektor erfordern, ohne Zeitverzögerung die Datenspeicherung gestartet. Im Falle einer positiven Entscheidung einer der vier Triggerblöcke oder eines *Bypass*-Signals, wird ein *Event*-Signal an die Detektoren gesendet. Dies weist die Detektoren an die aufgenommenen Informationen endgültig zu speichern. Im Falle, dass die geforderten Triggerbedingungen nicht erfüllt werden oder der FACE mehr als 10 µs für seine Entscheidung braucht, wird mit einem *Fastreset*-Signal signalisiert, dass die gespeicherten Daten verworfen werden sollen und die erste Triggerstufe erneut mit der Suche nach passenden Triggerbedingungen beginnen soll. Im Falle einer positiven Entscheidung der zweiten Stufe wird zusätzlich eine 3-bit breite *Event-ID* ausgegeben. Über die *Event-ID* können die Subdetektoren angewiesen werden zusätzliche Informationen auszulesen. Eine genauere Beschreibung der Auswirkung verschiedener *Event-ID*s befindet sich in Kapitel 5.1.1. Beim CAMAC-Trigger wurden bisher zwei verschiedene IDs implementiert. Für eine normale Auslese wurde eine 0 übertragen. Falls ein *scaler_event* Signal an die zweite Triggerstufe übertragen wurde, wurde eine 4 übertragen. Dadurch wird die Auslese zusätzlicher Zähler aktiviert. Alle diese Signale werden wieder im ECL-Standard übertragen. Der Aufbau der zweiten Triggerstufe ist schematisch in Abbildung 3.4 gezeigt.

3.3. VME-Trigger

Die CAMAC-Variante des Triggers hat einige Nachteile. Dadurch, dass sich die erste Triggerstufe nicht konfigurieren lässt, ist es sehr aufwändig die Triggerbedingungen zu ändern (siehe Kapitel 3.2.1). Die Logikblöcke des verwendeten FPGA-Chips sind zudem vom aktuellen Design zu 99% belegt, so dass neue Funktionalität nicht mehr hinzugefügt werden kann. Die Übertragung von Daten über den CAMAC-Bus ist zudem im Vergleich zur Übertragung über den VME-Bus um den Faktor 5 bis 10 langsamer. Da bei jedem Ereignis der Zustand des Triggers, sowie einige Zähler gespeichert werden, wird die Totzeit durch Verwendung des VME-Bus-Systems deutlich reduziert. Als letztes ist noch zu erwähnen, dass die Module nur noch schwer zu reparieren sind, da sie nicht mehr erhältliche Bausteine enthalten. Aus diesen Gründen wurde ein neues Triggermodul entwickelt. Als Basis dafür wurde ein von A. Winnebeck für das Olympus-Experiment [Mil+14] entwickeltes Triggermodul benutzt. Es wurde im Rahmen dieser Arbeit auf das Crystal-Barrel/TAPS-Experiment angepasst. Der Trigger wird dabei mit einer kommerziell erhältliche FPGA-Karten realisiert, welche über den VME-Bus gesteuert wird. Der in dieser Karte verwendete FPGA-Chip (XC3S1500) hat dabei etwa die zehnfache Zahl der Logikblöcke gegenüber dem in der CAMAC-Variante verwendeten FPGA-Chip (XC2S100). Dadurch ist es möglich, die erste Triggerstufe komplett konfigurierbar zu gestalten. Außerdem ist es nicht mehr nötig zwei verschiedene Module für die erste und zweite Stufe zu benutzen. Stattdessen sind jetzt beide Triggerstufen in einem Modul integriert. Dadurch entfällt die Verkabelung zwischen den beiden Stufen was den Aufbau ausfallsicherer macht. Im Folgenden werden die einzelnen Bestandteile des VME-Triggers sowie die im Rahmen dieser Arbeit erfolgten Modifizierungen genauer erklärt (siehe auch Abbildung 3.5).

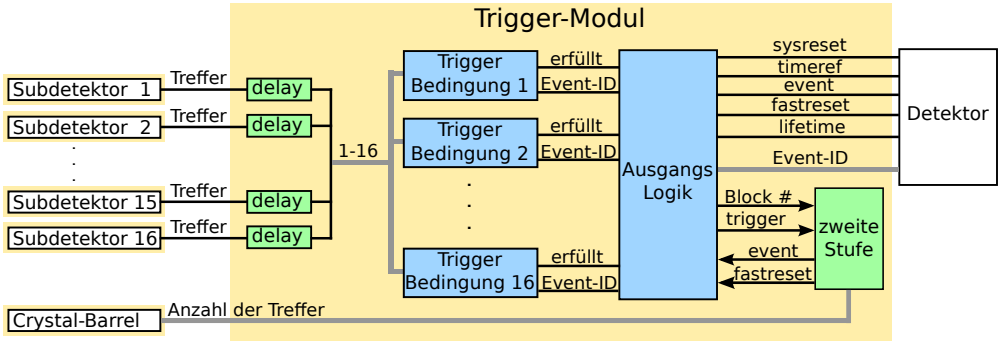


Abbildung 3.5.: **Überblick über den VME-Trigger.** Die grünen Elemente wurden dabei im Rahmen diese Arbeit hinzugefügt. Die blauen Elemente wurden für den Einsatz im Crystal-Barrel/TAPS-Experiment modifiziert.

3.3.1. Zeitanpassung der Eingänge

Die Triggersignale der Detektoren sind nicht Zeitstabil zueinander, da es in der Elektronik der Detektoren zu amplitudenabhängigen Verzögerungen des Triggersignals kommen kann. Und auch innerhalb der FPGA kann es zu unterschiedlichen Signallaufzeiten kommen, je nachdem welche Logikelemente passiert werden. Dies kann zu unerwünschten Effekten in der Logik der FPGA führen, wenn ein Signal B durch die verschiedenen Verzögerungen manchmal vor einem anderen Signal A und manchmal nach diesem an einem Logikelement ankommt. Wenn zum Beispiel durch das Logikelement der Zustand des Signals B beim Eintreffen des Signals A ausgewertet wird, wird der Zustandswechsel in Signal B nicht erkannt, wenn dieses kurz nach dem Signal A eintrifft. Dieses Beispiel eines Fehlerfalls wird in Abbildung 3.6 gezeigt.

Abbildung 3.6.: **Darstellung der Auswirkung von leicht wechselnden Signallaufzeiten bei asynchronen Logiken.** Die hier zugrundeliegende Schaltung überprüft beim Anstieg vom asynchronen Signal A das Signal B. Auf der linken Seite kommt dabei das Signal B kurz nach dem Signal A. Daher wird kein Ausgangssignal ausgegeben. Auf der rechten Seite kommt das Signal B kurz vor dem Signal A. Daher wird ein Ausgangssignal gesendet.

Dieser Effekt durch Schwankungen im Eintreffzeitpunktes kann dazu führen, dass die Logik in den meisten Fällen einwandfrei funktioniert und nur in Ausnahmefällen Fehler zeigt. Dieser Effekt wird Racecondition genannt. Um zu verhindern das dieses Problem auftritt wird Logik in FPGAs meist synchron aufgebaut. Das bedeutet, dass die Logikelemente nur mit einer festen Frequenz ihre Eingänge auswerten. Das Signal, welches diese Frequenz vorgibt, wird Takt genannt. Es schaltet mit einer festen Frequenz zwischen den logischen Zuständen 0 und 1. Die Logikelemente ändern ihren Zustand dabei nur, wenn der Zustand des Takt-Signals sich ändert. Die Logik in den FPGAs wird dann so angelegt, dass sichergestellt ist, dass die Ausgangssignale der einzelnen Elemente bis zum nächsten Schaltvorgang an den nachfolgenden Logikelementen angekommen sind.

Um jetzt die Trigger-Signale der Subdetektoren auf der FPGA in einem synchronen Design verwenden zu können, müssen diese zunächst an das Takt-Signal angepasst werden. Dies wird erreicht indem für jedes Signal ein neues Signal generiert wird,

welches bei der nächsten steigenden Taktflanke nach dem Ansteigen des Ursprungssignals eingeschaltet wird und beim nächsten Ansteigen des Takt-Signals nach dem Abfallen des Ursprungssignals ausgeschaltet wird. Dieser Vorgang, welcher das Ursprungssignal mit dem Takt synchronisiert, ist in Abbildung 3.7 dargestellt.

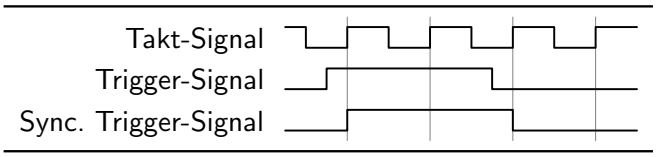


Abbildung 3.7.: **Synchronisation von Signalen auf den Takt in einer FPGA.** Bei jeder ansteigenden Flanke des Takt-Signals wird der Zustand des Trigger-Signals ausgewertet und auf das synchronisierte Trigger-Signal übertragen.

Für den korrekten Betrieb des Triggers ist es nötig, dass nach einem Ereignis alle Trigger-Signale zeitgleich an den Logikeinheiten ankommen. Dies kann entweder extern durch Einfügen von Kabeln der passenden Länge in die Triggerleitung geschehen oder digital in der FPGA. Die digitale Lösung in der FPGA hat dabei den Vorteil, dass dies im laufenden Strahlbetrieb ohne Unterbrechung des Beschleunigers geschehen kann. Für die digitale Verzögerung der Signale in der FPGA werden die synchronisierten Triggersignale durch ein konfigurierbares Schieberegister um ein Vielfaches der Periodendauer des Taktes verschoben. Um wie viele Zyklen die Signale dabei verzögert werden, kann für jeden Eingang über ein VME-Register konfiguriert werden.

3.3.2. Triggerlogikblöcke

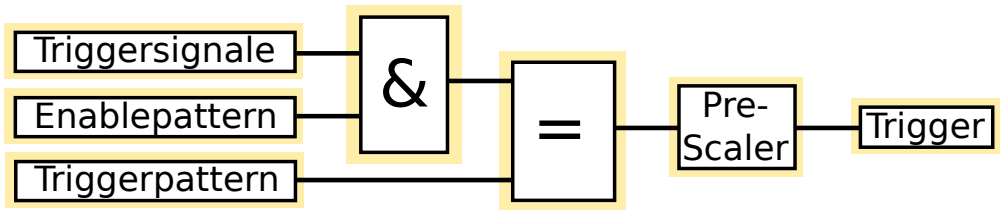


Abbildung 3.8.: **Schematischer Aufbau der Triggerlogikblöcke.** Zunächst wird mit dem *enablepattern* ausgewählt welche Triggereingänge betrachtet werden. Anschließend werden die ausgewählten Triggereingänge mit dem gewünschten *triggerpattern* verglichen. Abschließend wird mit einem *Prescaler* jedes X-te Triggerereigniss ausgewählt und ausgegeben.

Die Logikeinheiten die dafür zuständig sind zu entscheiden ob eine Triggerbedingung erfüllt ist, sind die Triggerlogikblöcke (TLBs). Es existieren im VME-Trigger parallel 16

identische Logikblöcke. Die zeitlich aneinander angepassten Triggersignale werden an jeden dieser Blöcke geleitet. Um dann festzulegen, bei welchen Signalen ein Triggersignal generiert wird, kann jeder dieser Logikblöcke mit drei 32bit-Wörtern (siehe Anhang C.1) konfiguriert werden. Dies erfolgt über das VME-Interface und kann somit jederzeit in wenigen Mikrosekunden durchgeführt werden. Mit diesen Konfigurationswörtern kann gesteuert werden bei welchen Mustern auf den Eingängen Triggersignale ausgegeben werden. Dabei kann sowohl ein Signal gefordert werden, als auch gefordert werden, das auf den Eingängen kein Signal anliegt. Dies ist notwendig um Detektoren zur ermöglichen, welche als Veto für den Trigger dienen sollen. Bei den meisten Experimenten gibt es Ereignisse welche benötigt werden um die Daten zu kontrollieren oder zu kalibrieren. Wenn diese Ereignisse jedoch mit einer hohen Rate auftreten ist es hinderlich diese einfach mit den Daten aufzunehmen. Dadurch würde eine hohe Totzeit entstehen und die gespeicherte Zahl der direkt für das Experimentprogramm benötigten Ereignisse wäre deutlich reduziert. Um diese Kontrollereignisse trotzdem während der Datennahme speichern zu können, gibt es eine Option die Rate in einzelnen Triggerlogikblöcken zu unterdrücken. Dabei wird nicht bei jedem Auftreten des geforderten Triggermusters ein Triggersignal ausgegeben, sondern mit einer reduzierten Rate. Dadurch kann die Rate mit der diese Kontrollereignisse gespeichert werden reduziert werden. Dieser Vorgang wird ‘‘Prescaling’’ genannt. Der Faktor mit der die Ausgabe der Triggersignale unterdrückt wird, kann für jeden Triggerlogikblock einzeln konfiguriert werden. Des Weiteren wird die von der zweiten Triggerstufe geforderte Multiplizität sowie die auszugebende Event-ID konfiguriert. Die Logikblöcke können außerdem so konfiguriert werden, dass sie nicht einen Triggereingang sondern ein mit einer konfigurierbaren festen Frequenz laufendes Signal für die Triggerentscheidung benutzen. Dadurch kann die Datennahme unabhängig von Triggerbedingungen getestet werden oder zu echten Ereignissen unkorrelierte Daten gespeichert werden. Unkorrelierte Daten werden zum Beispiel benutzt um Pedestals[7] in ADC-Karten zu bestimmen.

3.3.3. Ausgangslogik

Die Ausgangslogik ist dafür zuständig die Ausgangssignale des Triggers zu generieren. Dies sind zunächst die Signale, die ausgesendet werden nachdem einer der Triggerlogikblöcke ein Triggermuster gefunden hat. Bevor jedoch die Triggerlogikblöcke überhaupt gestartet werden, muss die Ausleseelektronik der Detektoren noch darauf vorbereitet werden ein Ereignis aufzunehmen. Dies wird durch das Übertragen des *Sysreset*-Signals angewiesen. Durch dieses Signal werden unter anderem die Zwischenspeicher der Elektronikmodule

[7] Energieeinträge wenn kein Ereignis aufgetreten ist. Diese Werte werden benutzt, um Verschiebungen der Nulllinie sowie das elektronische Rauschen zu bestimmen.

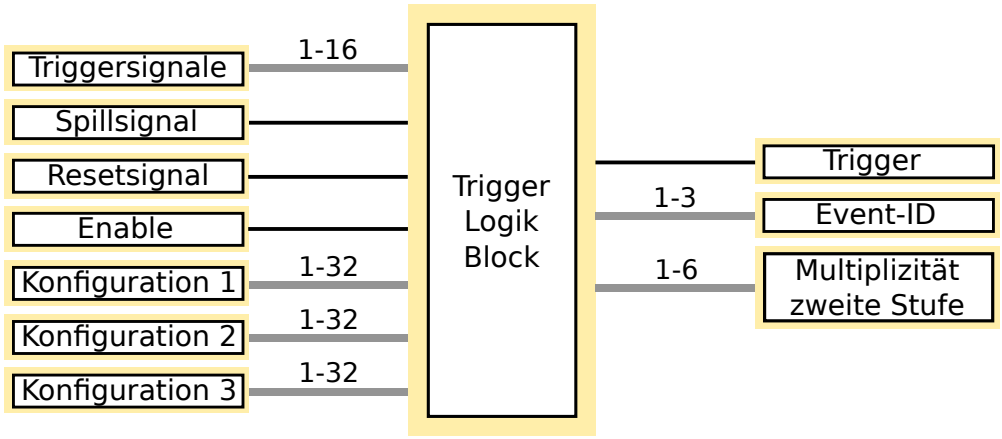


Abbildung 3.9.: **Ein- und Ausgangssignale der Triggerlogikblöcke.**

gelöscht. Das *Sysreset*-Signal wird ausgegeben sobald das Triggermodul über den VMEbus angewiesen wird ein passendes Triggerereignis zu suchen. Dannach folgt eine Zeit von 5 µs um sicherzustellen, dass alle Elektronikmodule erfolgreich zurückgesetzt wurden. Dannach werden dann die Triggerlogikblöcke aktiviert. Solange die Logikblöcke aktiv sind und noch keiner der Logikblöcke ein Triggersignal ausgegeben hat, wird das *Livetime*-Signal ausgegeben. Dieses *Livetime*-Signal kann benutzt werden um Zähler nur dann zählen zu lassen, wenn das Experiment für auftretende Ereignisse sensitiv ist. Das nächste Signal was verarbeitet wird ist das *Event-ID*-Signal. Das *Event-ID* Signal teilt der Datenauslese mit, welcher Typ von Ereignis stattgefunden hat. Dadurch können die Detektoren steuern, welche Module der Ausleseelektronik ausgelesen werden sollen. Das *Event-ID*-Signal ist dabei priorisiert. Dadurch kann festgelegt werden, welche *Event-ID* ausgegeben wird, falls zwei unterschiedliche Triggermuster gleichzeitig erfüllt sind. Eine höhere Priorität wird durch eine niedrigere *Event-ID* angezeigt. Die *Event-ID*s von allen Triggerlogikblöcken werden dafür ausgewertet und die niedrigste von den Logikblöcken gesendete *Event-ID* weitergegeben. Die Auswertung erfolgt dabei 40 ns nach dem das erste Triggersignal von einem der Triggerlogikblöcke gesendet wurde. Dies soll sicherstellen, dass ein etwas später auftretendes Triggermuster höhererer Priorität trotzdem berücksichtigt wird. Zu diesem Zeitpunkt wird auch gespeichert, welche Logikblöcke ein Triggersignal ausgegeben haben. Dies kann später ausgelesen werden, um zu bestimmen welche Triggerbedingungen von dem gerade verarbeiteten Ereignis erfüllt waren.

Ein weiteres Signal, das ausgegeben wird, ist das *Timeref*-Signal. Dieses Signal wird direkt an die Ausleseelektronik geleitet. Daraus werden dann die benötigten Signale für

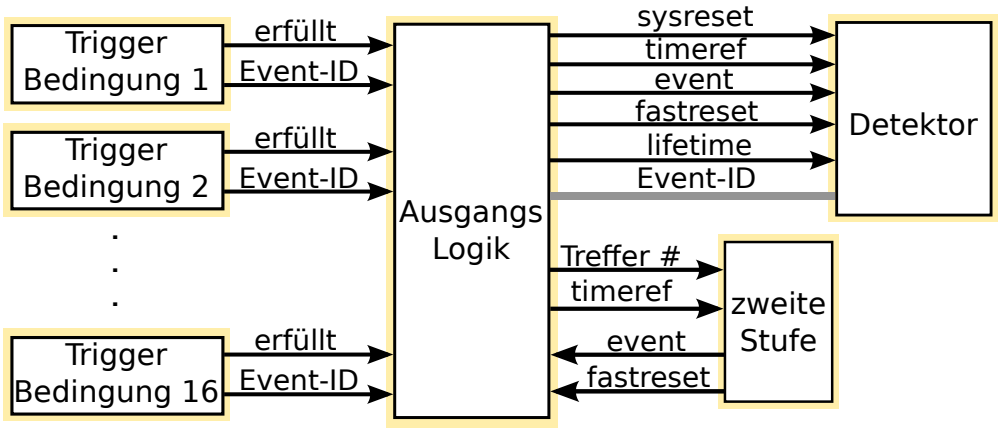


Abbildung 3.10.: **Die Ausgangslogik des VME-Triggers.** Die Ausgangslogik empfängt die Signale der einzelnen Triggerlogikblöcke und der zweiten Triggerstufe und sendet Signale an die zweite Triggerstufe und gibt Signale für die Detektoren aus.

TDC[8]- (Start bzw. Stop) und ADC[9]-Module (Gate[10]) generiert. Das *Timeref*-Signal wird dabei direkt gesendet, nachdem einer der Logikblöcke ein Triggersignal ausgegeben hat.

Das nächste Signal, das ausgegeben wird, ist das *Event*-Signal. Dieses Signal weist die Subdetektoren an, die in der Ausleseelektronik generierten Informationen auszulesen und zu speichern. Im Falle eines einstufigen Triggers würde einfach ein Signal auf den *Event*-Ausgang gelegt, sobald einer der Triggerlogikblöcke ein Trigger-Signal ausgibt. Für einen zweistufigen Trigger muss zusätzlich noch die Entscheidung der zweiten Stufe berücksichtigt werden. Jeder Triggerlogikblock welcher eine positive Entscheidung getroffen hat und ein Triggersignal ausgesandt hat, sendet dafür zunächst die geforderte Trefferzahl im Crystal-Barrel-Detektor an die Ausgangslogik. Diese bestimmt dann 40 ns nachdem das erste Signal der Logikblöcke angekommen ist die niedrigste geforderte Trefferzahl und sendet diese an die zweite Triggerstufe. Zusätzlich wird noch das *Timeref*-Signal an die zweite Stufe gesendet. Die zweite Stufe sendet dann ein *Event*- oder ein *Fastreset*-Signal zurück. Beide Signal werden dann von dem Triggermodul ausgegeben.

3.3.4. Zweite Triggerstufe

Wie schon im Abschnitt 3.2 erwähnt, ist die Entscheidungszeit der Triggerlogik des Crystal-Barrel-Detektors (FACE) mit im Durchschnitt 6 µs zu langsam um in der ersten

[8] Time to Digital Converter. Modul zur Bestimmung des Zeitpunktes eines Ereignisses.
[9] Analog to Digital Converter. Modul zur Energiebestimmung eines Ereignisses.
[10] engl. für Gatter. Signal während dem das Analogsignal vom ADC ausgewertet wird.

Triggerstufe berücksichtigt werden zu können. Daher musste auch für den VME-Trigger eine zweite Triggerstufe implementiert werden. Die zweite Stufe benötigt, um eine Entscheidung fällen zu können, zunächst die Ausgangssignale des FACE. Der FACE sendet zunächst die Anzahl der im Crystal-Barrel-Detektor gefundenen Treffer. Des weiteren sendet er noch ein Signal, nachdem er den kompletten Crystal-Barrel-Detektor ausgewertet hat. Zusätzlich benötigt die zweite Stufe noch einige der Ausgangssignale der ersten Stufe. Dies sind zunächst das Triggersignal, welches die Triggerentscheidung der zweiten Stufe anstößt. Des weiteren noch das für den auslösenden Logikblock konfigurierte *2ndlevel*-Wort. Damit wird ausgewählt wie viele Treffer gefordert werden und ob überhaupt auf eine Entscheidung der FACE gewartet werden muss. In der zweiten Stufe gibt es dann drei mögliche Fälle. Wenn keine Entscheidung des FACE gefordert ist gibt die zweite Stufe direkt ein Signal auf die *Event*-Leitung und löst damit die Datennahme aus. Wenn innerhalb von 10 µs, nachdem die zweite Stufe von der erste Stufe aktiviert wurde, kein Signal des FACE angekommen ist, wird ein Signal auf die *Fastreset*-Leitung gegeben, wodurch die erste Triggerstufe für eine neue Entscheidung geöffnet wird. Dadurch wird dann auch die Ausleseelektronik der Detektoren zurückgesetzt und auf das nächste Ereignis vorbereitet. Falls das Signal des FACE vor diesen 10 µs an der zweiten Stufe ankommt wird die von dem FACE gesendete Anzahl der Treffer mit der von der ersten Stufe geforderten Anzahl verglichen. Falls diese Anzahl in dem geforderten Bereich liegt, wird ein Signal auf die *Event*-Leitung gegeben. Falls die geforderte Anzahl nicht in dem geforderten Bereich liegt wird ein Signal auf die *Fastreset*-Leitung gegeben.

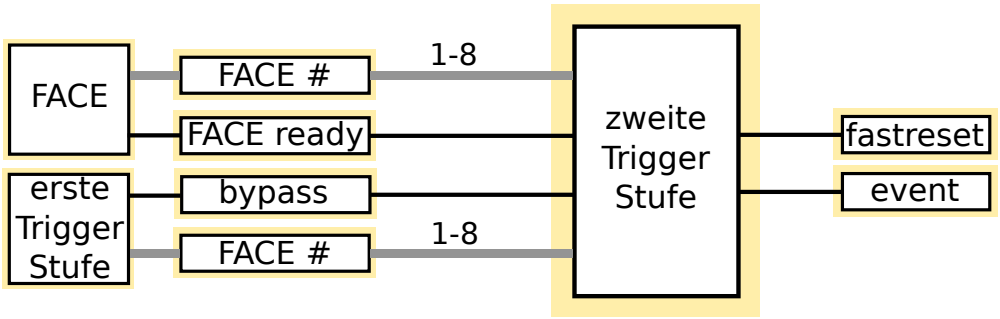


Abbildung 3.11.: **Ein- und Ausgangssignale der zweiten Triggerstufe.**

3.4. Fast Cluster Encoder

Der Fast Cluster Encoder wurde im Rahmen einer Doktorarbeit von H. Flemming an der Universität Bochum entwickelt [Fle01] und wird seit dem Jahr 2000 im Crystal-

Barrel/TAPS-Experiment eingesetzt. Die Aufgabe des FACE ist es die Signale des Crystal-Barrel-Detektors auszuwerten und zu bestimmen wie viele Teilchen diesen getroffen haben. Dabei kann nicht einfach nur die Zahl der gleichzeitig auftretenden Signale gezählt werden, da im Normalfall die Schauerausbreitung in den Kristallen des Crystal-Barrels so groß ist, das sich die Schauer auf mehr als einen Kristall ausbreiten. Dadurch entstehen durch ein einzelnes Zerfallsteilchen Signale in mehreren Kristallen. Um diese Treffer nicht mehrfach zu zählen bestimmt der FACE zunächst zusammenhängende Gebiete (Cluster) von Signalen des Crystal-Barrels und zählt dann die Zahl dieser Cluster. Um dies zu erreichen, werden zunächst die Signale der, je nach Experimentphase, bis zu 1380 Kristalle abgegriffen nachdem diese pulsgeformt wurden. Diese Signale werden dann von Diskriminatoren digitalisiert und an Latch-Speichermodule geleitet. Diese Module speichern den aktuellen Zustand der Signale, wenn sie dazu aufgefordert werden. Der FACE ist daher nicht freilaufend sondern muss von einer externen Quelle gestartet werden. Es ist also immer eine erste Triggerstufe nötig, die eine Anfrage über die Zahl der Treffer stellt. Nachdem die Signale in den Latchmodulen gespeichert wurden werden sie von der eigentlichen FACE-Logik ausgewertet. Diese ist in ASIC-Chips realisiert. Dabei handelt es sich um Bausteine, die Logikmodule enthalten, die für die geforderte Aufgabe miteinander verbunden werden. Sie sind damit ähnlich zu den FPGA-Bausteinen, die im Experiment Trigger verwendeten werden. Im Gegensatz zu diesen kann diese Verbindung der Logikmodule jedoch nicht wieder geändert werden. Die Funktionalität wird daher einmal bei der Erstellung der Chips festgelegt und kann danach nicht mehr verändert werden. Die ASIC-Bausteine haben jedoch den Vorteil, dass sie besser optimiert werden können und daher im Normalfall deutlich schneller arbeiten. Um an die Zahl der Cluster

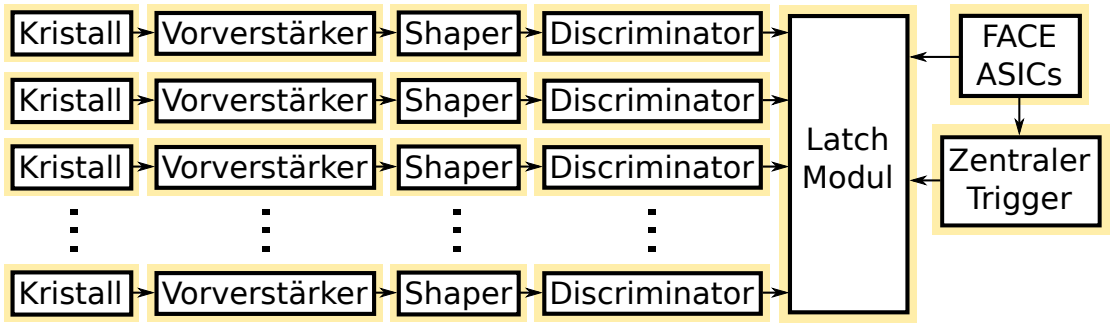


Abbildung 3.12.: **Überblick über den Signalfluss der Triggerlogik des Crystal-Barrel-Detektors.** Die Signale der Kristalle werden nach einem Vorverstärker in einem Shaper signalgeformt. Dannach mit einem Discriminator digitalisiert und nach Aufforderung vom Trigger in Latchmodulen gespeichert. Die FACE-Logik bestimmt dann aus dem gespeicherten Muster die Anzahl der Cluster und sendet diese an den Trigger.

zu kommen bestimmt die FACE-Logik zunächst nach einer vorgegebenen Priorität einen Kristall mit einem Treffer. Dann wird dieser Treffer gelöscht und alle mit diesem Kristall in einem Cluster verbundenen Treffer werden gesucht und gelöscht. Danach sucht der Algorithmus den nach der Priorität nächsten Treffer und löscht wieder alle verbunden Treffer. Jedes mal wenn ein neuer initialer Treffer gefunden wird, wird dabei die Zahl der Cluster um eins erhöht. Diese Suche und das anschließende Löschen der Treffer geschieht so lange bis keine Treffer mehr übrig sind. Danach wird dann ein Signal an den zentralen Trigger gegeben, dass die FACE fertig ist. Die Zahl der Treffer wird dann ebenfalls an den Trigger gesendet. Zur Funktionskontrolle kann das aktuelle Treffermuster auch aus den Latchmodulen ausgelesen werden. Dieses wird mit den Experimentdaten gespeichert. Dadurch ist es möglich im Nachhinein diese Treffermuster mit den Energiewerten der Crystal-Barrel-Kristalle zu vergleichen.

3.5. Clusterfinder des Vorwärtskonus

Analog zum Crystal-Barrel-Detektor wird auch im Vorwärtskonus die Anzahl der Treffer durch eine dedizierte Logik bestimmt. Da es auch hier Aufgrund der Schauerausbreitung zwischen den Kristallen zu Signalen von mehreren Kristallen bei einem Treffer in nur einem Kristall kommen kann, wird auch hier nicht direkt die Anzahl der Signale bestimmt. Es wird stattdessen ebenfalls die Zahl der zusammenhängenden Gebiete vom Clusterfinder gezählt. Der Clusterfinder des Vorwärtskonus wurde im Rahmen einer Doktorarbeit von Ch. Funke entwickelt [Fun08]. Im Folgenden wird der Aufbau und die Funktion dieses Clusterfinders beschrieben. Die Kristalle des Vorwärtskonus sind in 3 Reihen mit je 30 Kristallen angeordnet. Der Clusterfinder teilt dies zunächst in 5 Blöcke im Format 6x3 Kristalle auf. Wenn jetzt für jeden dieser Blöcke die Anzahl der Cluster bestimmt und die Gesamtzahl summiert würde, wäre die Anzahl der Cluster zu hoch. Dies liegt daran, das Cluster sich auf zwei Blöcke verteilen können und somit doppelt gezählt würden. Um dies zu verhindern werden zusätzlich zu den eigentlichen Blöcken Überlappblöcke gebildet, welche bestimmen ob sich Cluster über mehr als einen Block ausbreiten. Diese Überlappblöcke bestehen aus 6 Kristallen im Format 2x3 und werden jeweils gebildet aus den Kristallen, welche an den benachbarten Block angrenzen. Diese Aufteilung ist in Abbildung 3.13 gezeigt.

Um nun die Anzahl der Cluster in den Blöcken zu bestimmen wird ein Speicherchip benutzt. In diesen ist für jede mögliche Anordnung von Treffern in dem Block die Zahl der Cluster gespeichert. Die Signale der Kristalle bestimmen dabei die ausgewählte Adresse im Speicherchip. Der gespeicherte Wert entspricht der Zahl der Cluster. Diese Zahl wird von den Speicherchips an das Hauptmodul des Clusterfinders weitergegeben,

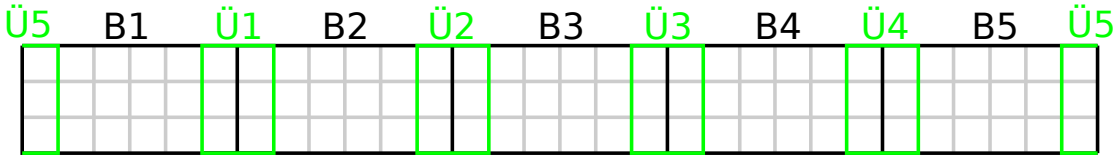


Abbildung 3.13.: **Aufteilung der 90 Kristalle des Vorwärtskonus für den Clusterfinder.** In Schwarz gezeigt sind die Blöcke in die die Kristalle zur Bestimmung der Zahl der Cluster aufgeteilt sind (B1 - B5). In Grün gezeigt sind die Überlappblöcke, die Cluster bestimmen sollen, die sich auf mehr als einen Block verteilen (Ü1 - Ü5).

welches dann die Summe der Blöcke bestimmt und davon die Summe der Überlappblöcke abzieht. Dies erfolgt ebenfalls mit einem vorprogrammierten Speicherchip der gleichen Form. Um die Speicherchips zu laden sind diese auf M-Modulen [ANSI97] auf 4-fach M-Modul-Carrier A201S der Firma MEN platziert, welche an den VME-Bus angebunden sind. Die Aufteilung der Blöcke und Überlappblöcke auf die M-Module ist in Abbildung 3.14 gezeigt.

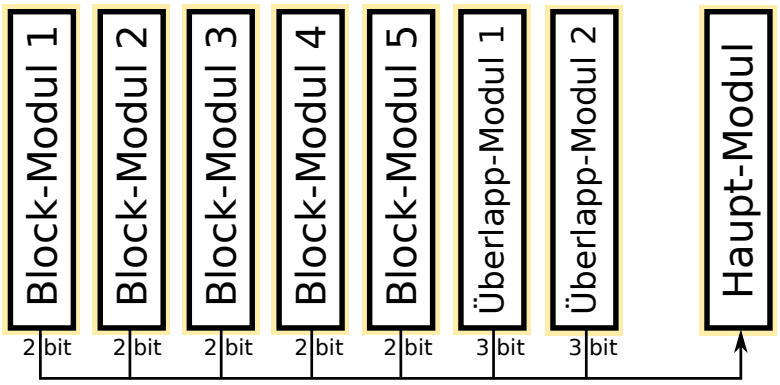


Abbildung 3.14.: **Aufbau des Clusterfinders des Vorwärtskonus.** Jeder 6x3 Block wird auf ein M-Modul geführt. In den Modulen für die Übergangssektoren sind mehrere Übergangsblöcke in einem Modul zusammengefasst. Von den Block-Modulen geht jeweils eine 2-bit breite Leitung zum Hauptmodul, von den Übergangs-Modulen jeweils eine 3-bit breite Leitung.

4. Synchronisation

4.1. Das Synchronisationssystem

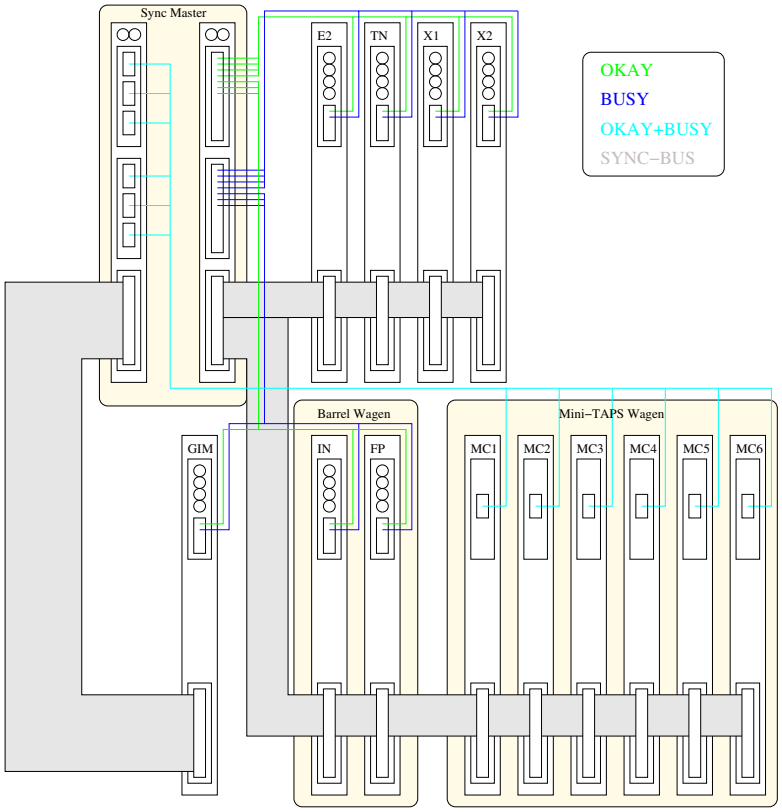


Abbildung 4.1.: **Schematische Darstellung des Synchronisationssystems.** Vom Sync-Master wird der Sync-Bus (grau) in drei Strängen zu den einzelnen Sync-Clients geführt. Für die MiniTAPS Sync-Clients wird das *Busy* und *Okay* Signal zusammen auf einem 8 adrigen CAT7 Kabel (türkis) zum Sync-Master geführt. Für die restlichen Detektoren werden diese beiden Signale einzelnd über jeweils eine Doppelader (grün+blau) geführt.

Zur Durchführung eines Teilchenphysikexperiments ist es nötig aus den Signalen, die von den Detektoren ausgegeben werden, den Zeitpunkt und die Energie der Signale zu

extrahieren. Dafür werden elektronische Module verwendet. Dies sind ADCs[1] für die Energieinformation und TDCs[2] für die Zeitinformation. Beide Module werden im Folgenden zusammenfassend Ausleseelektronik genannt. Das Sammeln und Speichern dieser Informationen wird allgemein Datennahme genannt. Das Extrahieren der Information und die Übertragung der digalisierten Daten an die Software, welche zum Sammeln der Informationen zuständig ist, kostet Zeit. Da während dieser Zeit die Aufnahme weiterer Ereignisse nicht möglich ist, ist das Detektorsystem während dieser Zeit tot. Diese Zeit wird deswegen auch Totzeit genannt. Für ein effizientes Experiment ist es nötig diese Zeit zu minimieren. Dies wird durch ein möglichst hohe Parallelisierung der Übertragung der Daten aus der Ausleseelektronik erreicht. Eine Parallelisierung hat jedoch den Nachteil, das die Informationen nicht zusammen an einer Stelle vorliegen, sondern verteilt auf mehreren Rechnern. Damit die zusammengehörigen Teilkomponenten korrekt zum Gesamtereignis zusammengesetzt werden können ist es nötig die Komponenten untereinander zu synchronisieren. Die dafür zuständige Komponente ist das Synchronisationssystem. Die erste Hauptaufgaben des Synchronisationssystems ist es dabei sicherzustellen, dass keine neuen Ereignisse aufgenommen werden, solange noch nicht alle parallelen Komponenten mit dem vorhergehenden Ereignis fertig sind. Die zweite Hauptaufgabe ist es eine Ereignisnummer an alle parallelen Komponenten zu verteilen, welche dann an die jeweiligen Teilkomponenten des Ereignisses angehängt werden. Dadurch wird sichergestellt das nur die zusammengehörenden Komponenten zusammengesetzt werden.

Bisher wurde dieses Synchronisationssystem mit modifizierten Latch-Counter Modulen des SAPHIR[3]-Experimentes realisiert [Sch04]. Die Logik dieser Module wird über GAL[4]-Bausteine erreicht. Der Prozess diese Module zu modifizieren ist sehr aufwendig, da dafür die GAL-Bausteine aus den Modulen entfernt werden müssen und diese anschließend in einer externen Programmierstation überschrieben werden. Für Änderungen die nicht in den GAL-Bausteinen direkt durchgeführt werden können muss zusätzlich die Verdrahtung auf den Modulen geändert werden. Dieser Prozess, der bei jeder Funktionsänderung an allen Modulen im Experiment durchgeführt werden muss, ist sehr zeitaufwendig, so dass Änderungen nicht einfach getestet und implementiert werden können. Um diesen Vorgang zu vereinfachen und das Synchronisationssystem flexibler zu gestalten, wurden im Rahmen dieser Arbeit die vorhandenen Module durch neue Module auf Basis eines

[1] engl. für Analog Digital Converter - Modul um analoge Signale in Digitale umzuwandeln
[2] engl. für Time to Digital Converter - Modul um Zeitinformationen zu digitalisieren
[3] Spectrometer Arrangement for Photon induced Reactions [Sch+94].
[4] Generic Array Logic, ein Bauteil in das logische Verknüpfungen einprogrammiert werden können

FPGA[5]-Bausteins ersetzt. Der Vorteil dieses Systems ist, dass es direkt über die VME[6]-Schnittstelle programmiert werden kann und somit, zur Änderung der Funktionalität, das Modul im Aufbau verbleiben kann.
Außerdem kann durch eine FPGA deutlich komplexere Logik realisiert werden, so dass der Funktionsumfang des Synchronisationssystems deutlich erweitert werden kann. In den folgenden Kapiteln wird die verwendete Hardware, die entwickelte Software sowie die Funktionalität der einzelnen Module beschrieben. Das Synchronisationssystem besteht dabei aus Synchronisations-Master Modulen (Sync-Master), Synchronisations-Client Modulen (Sync-Client), einem unidirektionalen Bus sowie jeweils zwei Rückkanälen pro Sync-Client Modul. Der Bus ist dabei, damit er zu allen Subdetektoren gelangt, in 3 Stränge geteilt (siehe Abbildung 4.1). Der zeitliche Verlauf und die Kommunikation zwischen den einzelnen Bestandteilen ist in Abbildung 4.2 gezeigt.

4.1.1. Das Sync-Client Modul

Die Hauptaufgabe des Sync-Client Moduls ist es den Status der Auslese des Subdektors zum Sync-Master zu übertragen. Dadurch wird sichergestellt dass der Trigger nicht für weiter Ereignisse geöffnet wird, solange der entsprechende Subdetektor noch nicht fertig mit der Übertragung der digitalisierten Informationen aus den Auslesekarten des Subdetektors ist. Dafür stehen ihm zwei Signale zur Verfügung (Das *Busy*-Signal und das *Okay*-Signal). Das *Busy*-Signal wird dabei gesetzt solange die Auslese des Subdetektors läuft und der Subdetektor nicht bereit für ein weiteres Ereignis ist. Das *Okay*-Signal wird gesetzt wenn der Subdetektor nach der Auslese bereit für das nächste Ereignis ist. Dabei wird das *Busy* direkt vom Sync-Client Modul gesetzt, damit sichergestellt ist, dass der Trigger nicht wieder geöffnet wird, solange die Auslese noch im Gang ist. Dies ist nötig, da es sich bei dem verwendeten Betriebssystem nicht um ein Realtime-System handelt. Bei Nicht-Realtime Systemen kann die Abarbeitung des Programms jederzeit vom System unterbrochen werden um andere Aufgaben durchzuführen. Die Dauer dieser Unterbrechung ist dabei nicht limitiert und kann durchaus auch mehr als eine Sekunde betragen. Beim Setzen des *Busy*-Signals über die Software wäre es daher möglich, dass der Prozess vor dem Setzen des *Busy*-Signals unterbrochen wird und in der Zeit die vergeht bis das *Busy*-Signal gesetzt wird, der Trigger schon das nächste Ereignis freigegeben hat. Durch das Setzten des *Busy*-Signals mit der FPGA ist garantiert, dass nach einer konstanten Zeit von einigen hundert Nanosekunden nach dem *Event*-Signal das *Busy*-Signal am Sync-Master eintrifft. Gelöscht wird das *Busy*-Signal

[5] Field Programmable Gate Array - Bauteil in das logische Schaltungen programmiert werden können. Die realisierbare Logic ist deutlich komplexer als bei GAL-Bausteinen.
[6] Busstandard zur Datenübertragung (ANSI/VITA 1-1994)

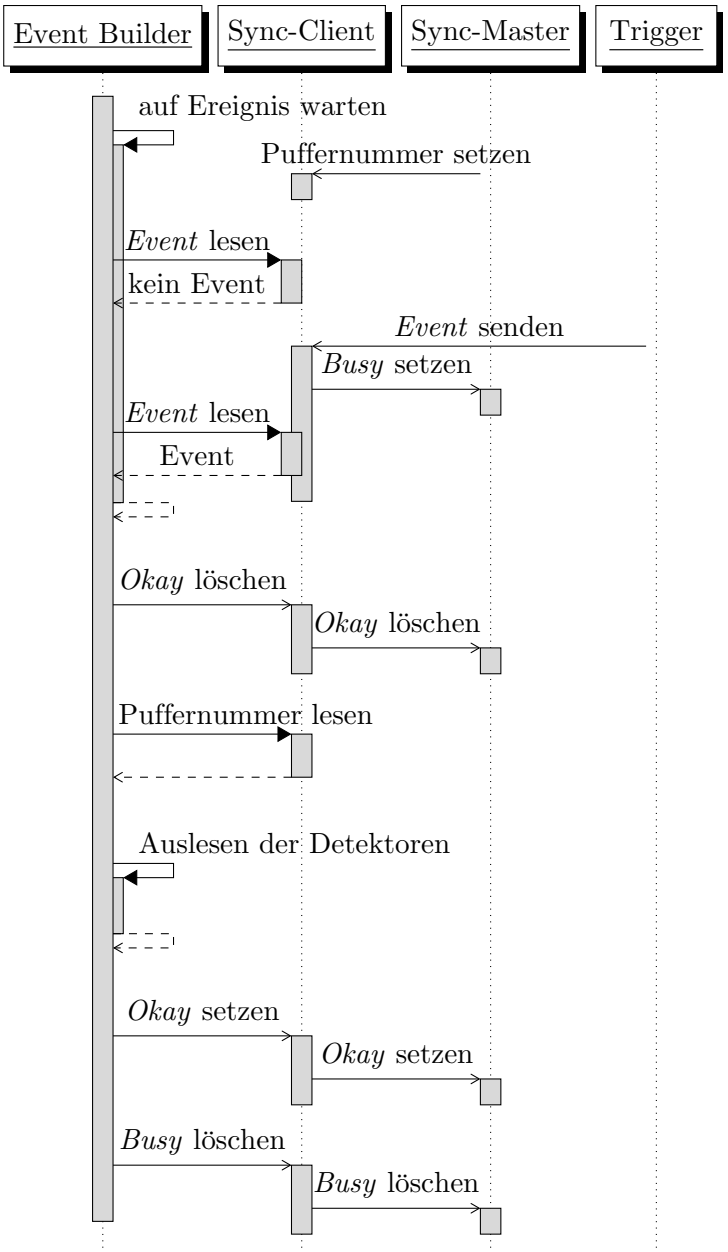


Abbildung 4.2.: **Schematische Darstellung des zeitlichen Ablaufs des Synchronisationssystems** Während der Eventbuilder auf ein Ereignis wartet wird zunächst vom Sync-Master über den Sync-Bus die Puffernummer gesetzt. Nach einer positiven Triggerentscheidung wird ein *Event*-Signal an den Sync-Client gesendet, wobei sofort von diesem ein *Busy* an den Sync-Master signalisiert wird. Nachdem der Eventbuilder das positive *Event*-Signal gelesen hat, löscht er das *Okay*-Signal und liest die Puffernummer. Nach erfolgter Auslese setzt er das *Okay*-Signal und löscht das *Busy*-Signal.

entweder über den VME-Bus oder durch ein *Sysreset*-Signal vor dem erneuten Öffnen des Triggers. Zum Testen kann das *Busy*-Signal auch über den VME-Bus gesetzt werden. Das *Okay*-Signal wird nur über den VME-Bus gesetzt und gelöscht. Die Logik die das *Busy*- und *Okay*-Signal erzeugt ist in Abbildung 4.3 gezeigt.

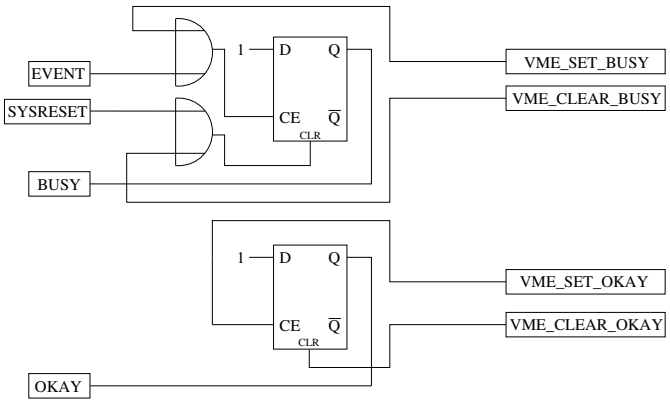


Abbildung 4.3.: **Logik-Block zur Erzeugung des *Busy*- und des *Okay*-Signals**. Beide Signale können über den VME-Bus gesetzt und gelöscht werden. Das Busy Signal wird zusätzlich vom *Event*-Signal gesetzt und vom *Sysreset*-Signal gelöscht.

Eine weitere Aufgabe des Sync-Client Moduls ist es die Ereignisnummer und den Ereignis-Typ vom Sync-Master zu empfangen und zu speichern. Beide Nummern werden beim Eintreffen des *Event*-Signals gespeichert und können dann vom Ausleseprozess gelesen und ausgewertet werden. Der Ereignis-Typ steuert dann die Art der Auslese, die Ereignisnummer wird an das jeweilige Ereignis angehängt.

Die letzte wichtige Aufgabe der Sync-Client Module ist es dem Ausleseprozess des Subdetektors zu signalisieren, wann dieser mit der Auslese beginnen soll. Dazu werden vom Trigger zwei Signale an alle Sync-Client Module verteilt. Mit dem *Event*-Signal signalisiert der Trigger, dass die Subdetektoren mit der Auslese beginnen sollen. Im Falle eines zweistufigen Triggers wird dieses Signal von der zweiten Stufe geliefert und ist nicht mit dem von der ersten Stufe gelieferten zum Triggerzeitpunkt zeitstabilen *timeref*-Signal zu verwechseln. Das zweite Signal welches an die Sync-Client Module verteilt wird ist das *Sysreset*-Signal. Dieses Signal wird vom Trigger, einige μs bevor der Trigger anfängt auf neue Ereignisse zu warten, gesetzt. Das *Sysreset*-Signal signalisiert den Subdetektoren, dass sie sich auf das nächste Ereignis vorbereiten sollen. Im Sync-Client Modul wird nun bei Eintreffen des *Event*-Signal ein Flipflop[7] eingeschaltet. Der Flipflop wird dann vom

[7]Ein Flipflop ist eine elektronische Schaltung die zwei stabile Zustände einnehmen kann. Es wird meist

Ausleseprozess über den VME-Bus zurückgesetzt, bevor die eigentlichen Auslese der Ausleseelektronik beginnt. Zusätzlich wird das Signal auch noch vom *Sysreset*-Signal zurückgesetzt.

Um steuern zu können, welche Subdetektoren ausgelesen werden können die einzelnen Sync-Client Module über den Sync-Bus aktiviert und deaktiviert werden. Nur wenn ein Sync-Client aktiviert ist wird das *Event*-Signal an das *Event*-Flipflop weitergeleitet. Wenn ein Sync-Client nicht aktiviert ist, bekommt der für die Auslese des Subdetektors zuständige Prozess daher auch kein Signal zur Auslese. Dies wird unter anderem dafür genutzt alle Ausleseprozesse in einem festen Zeitrahmen von einigen hundert Nanosekunden zu beenden. Dadurch ist sichergestellt, dass alle Subdetektoren die gleiche Anzahl an Ereignissen aufnehmen und nicht ein Subdetektor am Ende noch ein zusätzliches Ereignis sendet.

Zusätzlich zu den grundlegenden Funktionen befinden sich auf dem Sync-Client Modul noch eine Reihe von 32-bit Zählern, welche in Anhang D.2 aufgeführt sind. Diese Zähler dienen der Diagnose und Überwachung des Moduls. Es handelt sich dabei um 32-bit zwischenspeichernde[8] Zähler, welche über den VME-Bus ausgelesen werden. Ein Zähler zählt die Signale auf den *Event*- und *Sysreset*-Eingängen. Durch Vergleich dieser Zahlen mit der Anzahl vom Subdetektor ausgelesener Ereignisse und der vom Sync-Master Modul übertragenen Puffernummer werden Mehrfachpulse oder fehlende Pulse erkannt, sollten diese auf einem der beiden Eingänge auftreten. Desweiteren gibt es einen freilaufenden Zähler der mit einer festen Frequenz von 100 MHz zählt. Durch Abfragen dieses Zählers ist es möglich die Dauer einzelner Prozesse während der Datennahme, wie zum Beispiel der Auslese der Module über den VME-Bus, zu bestimmen. Außerdem gibt es noch zwei Zähler die nur während bzw. außerhalb der Totzeit, also während das *Busy*-Signal gesetzt ist, zählen. Mit Hilfe dieser Zähler kann der Anteil der mit Datennahme verbrachten Zeit bestimmt werden.

Ein Event-Builder ist zusätzlich noch für die Steuerung des Triggers zuständig. Damit ändert sich der in Abbildung 4.2 gezeigte Ablauf in einigen Punkten (siehe Abbildung D.1 im Anhang). Im Falle eines zweistufigen Triggers muß dieser Event-Builder außerdem noch darauf reagieren, dass der Trigger keine positive Entscheidung getroffen hat. Dies wird vom Trigger durch das *Fastreset*-Signal angezeigt. Zum Empfang dieses Signals hat der Sync-Client des für den Trigger zuständigen Event Builders einen weiteren Eingang. Der für die Kontrolle des Triggers zuständige Prozess kann dann, wenn auf diesem Eingang ein Signal ankommt, das Triggerfenster der ersten Stufe erneut öffnen. Dieses

durch ein Signal ein Zustandswechsel ausgelöst und durch ein zweites Signal der Zustand wieder zurück gesetzt.

[8] Zähler, welche auf ein Signal hin ihren Stand in ein Zwischenregister übertragen. Dadurch ist ein gleichzeitiger Stand der einzelnen Zählerstände garantiert.

Fastreset-Signal wird analog zu den anderen Signalen ebenfalls mit einem Zähler gezählt.

Die Befehle und Register mit denen die Sync-Client Module über den VME-Bus gesteuert werden können sind im Anhang D.1 aufgeführt. Der zeitliche Verlauf im Synchronisationssystem im Sonderfall des für den Trigger zuständigen Event Builders ist in den Abbildungen D.1 und D.2 im Anhang gezeigt.

4.1.2. Das Sync-Master Modul

Das Sync-Master Modul ist das zentrale Modul des Synchronisationssystems. Es ist zusammen mit der Triggerelektronik (siehe Kapitel 3) an den selben lokalen Eventbuilder (siehe Kapitel 5.1) angeschlossen. Die Hauptaufgabe des Sync-Master Moduls ist es sicherzustellen, dass der Trigger nicht wieder geöffnet wird solange die Ausleseprozesse der Subdetektoren nicht bereit für ein weiteres Ereignis sind. Dafür sammelt das Sync-Master Modul den Status über die *Busy*- und *Okay*-Leitungen von allen Sync-Client Modulen ein. Der für den Trigger zuständige Prozess überprüft dann ob die *Okay*-Signale von allen aktivierten Subdetektoren gesetzt sind. Analog müssen alle *Busy*-Signale gelöscht sein. Wenn diese beiden Bedingungen erfüllt sind, kann der Trigger für das nächste Ereignis freigegeben werden.

Eine weitere Aufgabe ist es sicherzustellen, dass die von den verschiedenen Subdetektoren gesendeten Teile der Ereignisse korrekt zusammengebaut werden. Dies wird über eine Puffernummer erreicht, welche vom Sync-Master über den Sync-Bus an alle Sync-Client Module verteilt wird. Dabei wird die Puffernummer übertragen, nachdem alle Sync-Client Module über die *Busy*- und *Okay*-Leitungen gemeldet haben, dass sie bereit sind. Der Trigger wird erst für das nächste Ereignis geöffnet, nachdem die Übertragung der Puffernummer stattgefunden hat. Dadurch ist sichergestellt, dass während der Auslese der Subdetektoren bei allen Sync-Client Modulen die gleiche Puffernummer gespeichert ist.

Die letzte Aufgabe des Sync-Master Moduls ist es die Sync-Client Module der für die Auslese ausgewählten Subdetektoren zu aktivieren. Dies geschieht wie in Kapitel 4.1.3 beschrieben ebenfalls über den Sync-Bus. Zusätzlich können am Ende eines Datenblocks, durch ein Kommando auf dem Sync-Bus, alle Sync-Client Module innerhalb eines Zeitrahmens von wenigen hundert Nanosekunden deaktiviert werden. Dieser Zeitrahmen ist nur durch die Laufzeit des Signals auf dem Sync-Bus bestimmt, so dass die Sync-Client Module immer mit dem gleichen Zeitversatz deaktiviert werden. Dadurch ist sichergestellt, dass alle Subdetektoren die gleiche Anzahl an Ereignissen senden.

Die verfügbaren VME-Adressen mit denen das Sync-Master Modul angesprochen werden kann ist im Anhang D.2 aufgeführt.

4.1.3. Der Sync-Bus

Der Sync-Bus besteht aus einem 16-bit breiten Bus über den Signale vom Sync-Master an die Sync-Clients gesendet werden. Der Bus hat zwei Aufgaben. Zunächst wird für jedes Ereignis eine Puffernummer übertragen. Diese 5-bit Zahl wird von den Subdetektoren mit in die Datenpakete geschrieben. Dadurch ist es beim Zusammenstellen der Ereignisse möglich die zusammengehörenden Datenpakete herrauszusuchen.

Des Weiteren können die einzelnen Sync-Client Module über diesen Bus aktiviert und deaktiviert werden. Nur wenn ein Sync-Client aktiviert ist, akzeptiert er eintreffende *Event*-Signale. Dadurch ist es möglich die Auslese zeitnah im Rahmen von einigen hundert Nanosekunden in allen Detektorkomponenten zu stoppen. Zusätzlich wird über den Bus eine 3-bit Event ID übertragen. Die Event ID ist eine vom Trigger festgelegte Nummer, die abhängig von der Bedingung, welche das Ereignis ausgelöst hat, gesetzt werden kann. Dadurch können bei bestimmten Triggerbedingungen zusätzliche Komponenten zum Ausleseprozess hinzugefügt oder weggelassen werden. Es können dadurch zum Beispiel auch zusätzliche Zählermodule in gewissen Zeitabständen ausgelesen werden. Realisiert wird der Sync-Bus im Experiment durch ein 16-fach twisted-pair[9] Flachbandkabel, in dem jedoch nur 10 Aderpaare belegt sind. Der verwendete Signalstandard ist ECL, womit eine Übertragung der Signale über eine Strecke von bis zu 70 m fehlerfrei möglich ist. Der Bus kann zur Übertragung von Daten in zwei verschiedene Modi versetzt werden. Welcher Modus verwendet werden soll, wird durch den *Adress Select*-Kanal (Kanal 9) angezeigt. Im ersten Modus (angezeigt durch eine 0 auf dem *Address Select*-Kanal) wird auf den Kanälen 3 bis 7 die Puffernummer übertragen. Die Puffernummer ist eine laufende Nummer, welche nach jedem Ereignis um eins erhöht wird. Der zweite Modus dient der Aktivierung der einzelnen Module (*Address Select*-Kanal ist 1). Dabei wird auf den Kanälen 3 bis 7 die Adresse des zu aktivierenden Detektors angelegt und der Detektor dann mit einer steigenden Flanke von *Command Strobe* auf Kanal 8 aktiviert. Die Adresse 0 hat dabei eine besondere Bedeutung, da bei ihr die Aktivierung aller Module gelöscht wird. Der Aufbau des Sync-Busses ist in Tabelle 4.1 gezeigt.

4.1.4. Das Sync-Tap Modul

Da die Ausleseelektronik der Detektoren im Experimentbereich räumlich weit voneinander getrennt sind würde das Vorbeiführen des Sync-Busses an abgelegenenen Komponenten zu einer erheblichen Verlängerung des Busses führen. Dies ist aber nur in berenztem Maße möglich, da die Signale sonst durch den Transport zu sehr abgeschwächt und

[9] Englisch für verdrehtes Paar - Dabei werden Signalkabel paarweise miteinander verdrillt um eine störungsfreiere Übertragung zu ermöglichen.

Bit	15 - 10	9	8	7	6	5	4	3	2	1	0
Modus 1	Frei	Addr Sel 0		Buf-Nr Bit 4	Buf-Nr Bit 3	Buf-Nr Bit 2	Buf-Nr Bit 1	Buf-Nr Bit 0	Event ID	Event ID	Event ID
Modus 2	Frei	Addr Sel 1	Cmd-Strobe	Addr Bit 4	Addr Bit 3	Addr Bit 2	Addr Bit 1	Addr Bit 0	Event ID	Event ID	Event ID

Tabelle 4.1.: **Belegung der Leitungen des Sync-Busses.**
Oben dargestellt ist die Belegung im Falle der Übertragung der Puffernummer (Modus 1). Unten ist die Belegung bei der Aktivierung der Sync-Client Module zu sehen (Modus 2). Bit Nr. 9 (*Address Select*) bestimmt welcher Modus gerade verwendet wird.

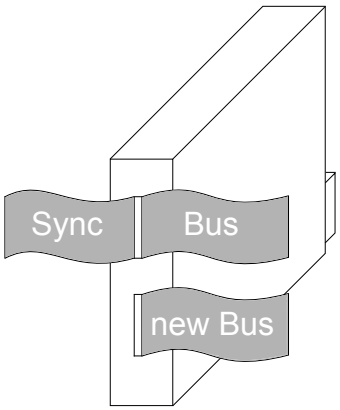
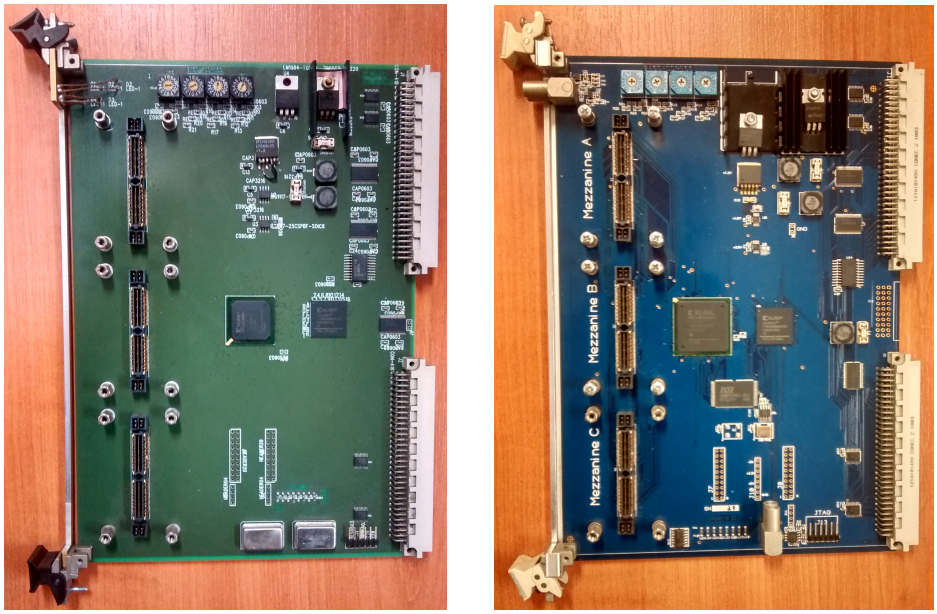


Abbildung 4.4.: **Schematische Darstellung des Sync-Tap Moduls**

verformt würden. Um diese Verlängerung zu verhindern wurde das Sync-Tap Modul entwickelt. Es handelt sich dabei um ein NIM Modul mit einem nicht terminierten ECL-Eingang. Dadurch können die Informationen des Busses an dieser Stelle abgegriffen werden ohne ihn dort zu unterbrechen. Die Signale des Busses werden dann auf dem Modul zunächst nach LVTTL[10] und dann wieder zurück nach ECL konvertiert. Dadurch sind die beiden ECL-Signale komplett voneinander getrennt und das neue Signal hat wieder die Orginalamplitude und -form. Dieses neue ECL-Signal kann dann vom Ausgang des Moduls weitergeleitet werden. Dadurch ist es möglich an jeder beliebigen Stelle eines Busses einen neuen Teilstrang zu erzeugen und diesen zu abgelegenen Detektoren zu führen. Es ist mit diesem Module ebenfalls möglich die Signale des Busses aufzufrischen um eine gute Signalqualität über große Strecken sicherzustellen.

[10] Low Voltage Transitor-Transitor-Logic - Einadriger Übertragungsstandard mit Pegeln zwischen 0 V und 3,3 V

4.2. Die Hardware des Synchronisationssystems



(a) VFA (b) VFB2

Abbildung 4.5.: **Die VME-FPGA Grundplatinen** Auf der linken Seite zu sehen ist das Basismodell (VFA). Auf der rechten Seite ist das Modell VFB2 gezeigt, welches zusätzliche Ein- und Ausgänge des NIM-Standards mit LEMO-Buchsen zu Verfügung stellt. Dadurch können auch wenn die Zusatzkarten mit anderen Übertragungsstandards ausgestattet sind immer Signale des NIM-Standards empfangen und gesendet werden.

Um den Sync-Master und den Sync-Client zu realisieren, wurden VME-Module unterschiedlichen Types benutzt. Alle Module enthalten FPGAs vom Typ XC3S1500 oder XC3S4000 der Firma Xilinx. Das Basismodell VFA besitzt zudem 3 Steckplätze auf welche eine Reihe Zusatzkarten gesteckt werden können, um Eingangs- oder Ausgangssignale unterschiedlichen Typs mit der FPGA zu verbinden. Das erweiterte Modell VFB2 bietet zusätzlich zwei Eingänge für NIM-Signale mit LEMO[11]-Buchsen auf der Vorderseite, sowie 2 Ausgänge des gleichen Typs im hinteren Bereich der Platine. Dadurch ist immer gewährleistet, dass Signale vom NIM-Standard empfangen und gesendet werden können, auch wenn die Zusatzkarten mit anderen Übertragungsstandards ausgestattet sind. Die unterschiedlichen Module sind in Abbildung 4.5 gezeigt. Im Folgenden werden die verwendeten Aufsteckkarten des Synchronisationssystems erläutert.
Die Synchronisations-Master sind in zwei verschiedenen Varianten realisiert. Die erste

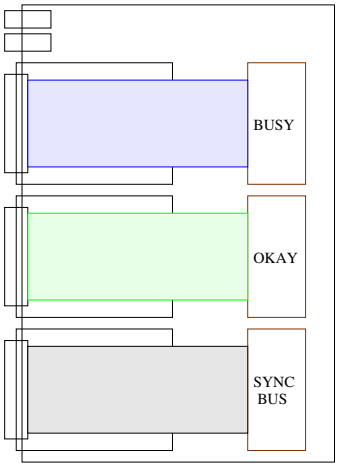
[11] Steckertyp LEMO 00 der Firma LEMO

Variante benutzt zwei Karten mit 16 ECL-Eingängen für die *Busy*- und *Okay*-Signale, sowie eine Karte mit 16 ECL-Ausgängen für den Sync-Bus, wobei jeweils die Hälfte der Ausgänge für jeweils einen eigenständigen Bus verwendet werden. Dies ist möglich, da die ersten drei Kanäle der Busse (EventID) sowie die zwei weiteren von den Sync-Clients benötigten Signale (*Event* und *Sysreset*) nicht vom Sync-Master Modul gesendet werden, sondern direkt vom Triggermodul an die Sync-Client Module geleitet werden. Dadurch müssen pro Bus nur 7 Kanäle vom Sync-Master übertragen werden.
Die zweite Variante besteht aus zwei Karten mit jeweils 3 RJ45[12] Steckern. Über jeden der RJ45 Stecker werden dabei zwei Signale zu einem Synchronisations-Client herausgeführt (*Event* und *Sysreset*) sowie zwei Signale empfangen (*Busy* und *Okay*). Die Signale werden dabei nach dem LVPECL[13]-Standard übertragen. Die Signale *Event* und *Sysreset* werden durch die beiden LEMO-Buchsen auf der Vorderseite des Moduls auf die FPGA geleitet. Auf dem dritten Steckplatz befindet sich auch hier eine Karte mit 16 ECL-Ausgängen für den Sync-Bus. Die zweite Variante bietet dabei Vorteile bei der Verkabelung, da nur ein Kabel für die vier zusätzlich zum Sync-Bus benötigten Signale verlegt werden muss. Außerdem werden bei dieser Variante alle Signale differentiell und massefrei übertragen. Ein Nachteil ist jedoch, dass sich mit einem Sync-Master Modul nur 6 Client Karten, statt wie bei der ersten Variante 16 Karten, ansteuern lassen. Dadurch sind zur Ansteuerung der gleichen Anzahl von Sync-Clients mehr Sync-Master Module nötig. Beide Varianten sind schematisch in Abbildung 4.6 dargestellt.

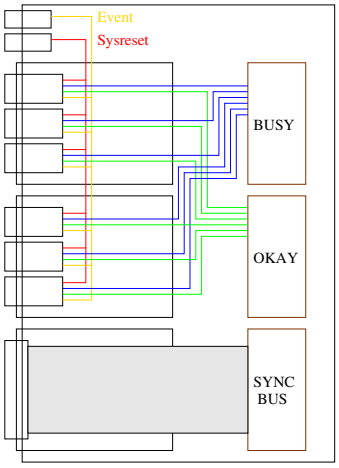
Die Synchronisation-Client Module sind ebenfalls in zwei verschiedenen Ausführungen realisiert. Die erste Variante benutzt dabei eine spezielle Karte mit 4 ECL-Ausgängen, sowie 4 NIM-Eingängen. Über diese Karte werden die *Busy*- und *Okay*-Signale auf den ECL-Ausgängen herausgeführt sowie die *Event*- und *Sysreset*-Signale auf den NIM-Eingängen empfangen. Die zweite Variante benutzt dafür eine Steckkarte mit einem RJ45 Stecker über den alle 4 Signale laufen. Als Signalstandard wird dabei, analog zu den RJ45 Steckern des Sync-Master-Moduls, LVPECL benutzt. Beide Varianten benutzen als zweite Aufsteckkarte eine Karte mit 16 ECL-Eingängen. Dabei sind diese Eingänge bei allen Karten bis auf die letzte Karte auf jedem Bus nicht terminiert, wodurch das Signalkabel an allen Modulen eines Busses vorbeigeführt werden kann.

[12] Steckkontakt mit 8 Kontakten. Häufig benutzt in Telekommunikations- und Netzwerkanwendungen. Spezifiziert in ANSI/TIA-1096-A.

[13] Abkürzung für low voltage positive emitter-coupled logic. Differenzieller Signalstandard konzipiert für Systeme mit +3,3V Spannungsversorgung. Die Amplitude der Signale beträgt dabei entweder 1,6V oder 2,4V.

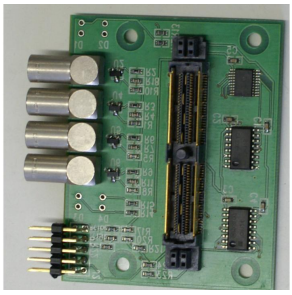


(a) Schematische Darstellung der ersten Variante des Sync-Masters

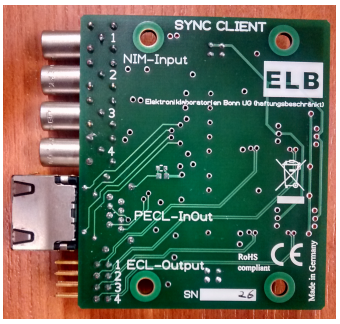


(b) Schematische Darstellung der zweiten Variante des Sync-Masters

Abbildung 4.6.: **Schematische Darstellung der beiden Varianten des Sync-Master Moduls.** Dargestellt sind in Blau der Verlauf der *Busy*-Signale, in grün der Verlauf der *Okay*-Signale. Die Signale für *Event* (gelb) und *Sysreset* (rot) werden in der Variante A direkt vom Trigger-Modul verteilt, in Variante B ist die Verteilung im Modul dargestellt.



(a) Foto der NIM-ECL Austeckkarte



(b) Foto der PECL Aufsteckkarte

Abbildung 4.7.: Dargestellt sind die beiden Varianten der Steckkarten für das Sync-Client Modul. Links ist die erste Variante mit 4 NIM Eingängen und 4 ECL Ausgängen gezeigt. Auf der rechten Seite ist die zweite Variante mit dem RJ45 Stecker für die LVPECL Signale gezeigt. Diese Platine besitzt aus Kompatibilitätsgründen zusätzlich noch die gleichen Anschlüsse wie die erste Variante.

4.3. Das COMPASS TCS-System

Im Crystal-Barrel/TAPS-Experiment werden in mehreren Komponenten TDC-Karten sowie Zähler-Karten, die für das COMPASS-Experiment [COM+07] entwickelt wurden, eingesetzt. Diese CATCH[14]-Module [Hei+01] können nur mit einem speziellen vom COMPASS-Experiment eingesetzte optische Trigger-Control-System [Gru06](TCS) gesteuert werden. Deshalb war es nötig dieses in das Crystal-Barrel/TAPS-Experiment zu integrieren. In den folgenden Abschnitten ist die Funktionsweise des Systems erklärt. Außerdem werden die im Rahmen dieser Arbeit erfolgten Anpassungen für das Crystal-Barrel/TAPS-Experiment erläutert.

4.3.1. Grundlagen des COMPASS TCS-Systems

Das COMPASS TCS-System [Gru06] ist ein optisches Kontrollsystem. Die optischen Übertragung hat dabei zwei große Vorteile gegenüber der klassischen elektrischen Übertragung. Zum Einen werden die Elektronikschränke durch die optischen Verbindungen nicht elektrisch verbunden, was die Signalqualität verbessert. Zum Anderen ist die Dämpfung bei optischer Übertragung sehr gering, so daß deutlich längere Verbindung möglich sind. Da die längste Verbindung im Crystal-Barrel/TAPS-Experiment jedoch unter 100m beträgt wäre die Dämpfung auch bei elektrischer Übertragung kein Problem. Das Wegfallen von elektrischen Verbindungen ist jedoch auch im Crystal-Barrel/TAPS-Experiment ein signifikanter Vorteil.

Das COMPASS TCS-System hat drei wesentliche Komponenten. Der TCS-Master empfängt alle wichtigen Steuersignale (siehe Abbildung 4.8) und erzeugt daraus zwei Signale (Kanal A und Kanal B) über welche alle Signale und Informationen digital übertragen werden. Dabei wird zusätzlich noch ein 40 MHz Takt eincodiert. Beide Kanäle werden dann an ein zweites Modul (TTCex) geleitet, welche nur dafür zuständig ist diese beide Kanäle in optische Signale umzuwandeln. Diese Signale werden dann optisch zu den Subdetektoren geleitet. Auf jede COMPASS TDC-Karte ist nun eine TCS-Empfänger Karte aufgesteckt, welche diese optischen Signale empfängt, dekodiert und an die TDC-Karten weiterleitet. Zusätzlich zu den eigentlichen Informationen wird auf den Empfängerkarten noch der in das optische Signal eincodierte Takt extrahiert. Das hat den Vorteil, dass alle COMPASS TDC-Karten des Experiments mit dem gleichen phasenstabilen Takt betrieben werden und dadurch auch die Zeitauflösung auf allen diesen Karten identisch ist. Für die Signalverteilung an die Empfängerkarten werden optische Signalteiler benutzt. Diese verteilen das Eingangssignal passiv auf bis zu 32

[14] Abkürzung für COMPASS Accumulate, Transfer & Control Hardware (engl. für COMPASS Module zum Zusammenführen, Transferieren und Kontrollieren).

Ausgänge.

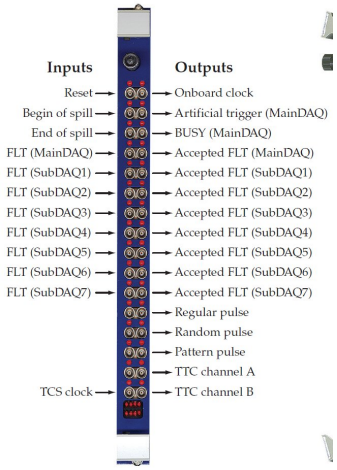


Abbildung 4.8.: **Der TCS-Master.** Der TCS-Master hat als Eingänge die Signale, für den Anfang und das Ende der Extraktion des Beschleunigers, das Triggersignal der ersten Triggerstufe, sowie die vorgegebene feste Frequenz auf welche die Signale aufmoduliert werden. Als Ausgangssignale steht zunächst ein *Busy*-Signal zur Verfügung, welches der Datenakquisition signalisiert, wenn das TCS-System beschäftigt ist. Ausserdem gibt es noch die beiden Kanäle A und B welche die Signale enthalten, die an das TTCex-Modul weitergeleitet werden. Bild aus [Gru06].

4.3.2. Anpassungen für das Crystal-Barrel/TAPS-Experiment

Um das TCS-System im Crystal-Barrel/TAPS-Experiment einsetzen zu können, mussten einige Anpassungen vorgenommen werden. Zum Einen wird im Crystal-Barrel/TAPS-Experiment im Gegensatz zum COMPASS-Experiment ein Zweistufentrigger verwendet. Die dadurch notwendigen Anpassungen werden im nächsten Abschnitt beschrieben. Des Weiteren sollten die Zählermodule nicht bei jedem Trigger sondern nur in festen Zeitabständen (Scaler-Ereignisse) ausgelesen werden. Diese Modifikation wird iletzten Abschnitt beschrieben. Zusätzlich wurde das Signal der ersten Triggerstufe noch auf einen TDC-Kanal von jedem Subdetektor gegeben. Dadurch wurde sichergestellt, dass an jedem Subdetektor zueinander zeitstabile Signale als Referenz zur Verfügung stehen.

Modifikation für den Zweistufentrigger

Durch den Zweistufentrigger des Crystal-Barrel/TAPS-Experiments wird von der ersten Triggerstufe zeitstabil zur Triggerentscheidung ein Signal erzeugt, welches Timerefe-

renz genannt wird. Dieses Signal wird an alle Detektoren weitergeleitet und startet das Prozessieren des Ereignisses in den Auslesekarten. Im Durchschnitt benötigt die zweite Stufe 6 μs um zu entscheiden, ob das Ereignis aufgenommen werden soll. Falls keine positive Entscheidung von der zweiten Stufe getroffen wird, muss die Ausleseelektronik auf das nächste Ereignis vorbereitet werden. Die meisten Auslesekarten besitzen dafür einen *Clear*-Eingang. Wenn dort ein Signal eintrifft, werden die bisher im Modul gespeicherten Daten verworfen und das Modul so zurückgesetzt, dass es das nächste Ereignis aufnehmen kann. Da dieses zweistufige Schema im COMPASS Experiment nicht existiert, ist dies mit dem TCS-System nicht direkt möglich. Um dieses Schema dennoch zu implementieren wurde benutzt, das das TCS-Systems die Möglichkeit bietet Befehle an die TCS-Empfängerkarten zu senden. Der TCS-Master besitzt dafür einen Puffer in den Befehle geschrieben werden können (Slow Command Buffer). Diese Befehle werden dann der Reihe nach über das TCS-System an die TCS-Empfänger übertragen. Einer dieser Befehle ist das Kommand BC1 (siehe Anhang D.4). Mit diesem Kommando kann ein Befehl zum Zurücksetzen der TDC-Module an die Empfängerkarten übertragen werden. Wenn die zweite Triggerstufe also keine positive Entscheidung trifft, wird ein *Fastreset*-Signal an den zuständigen Sync-Client gesendet. Der Prozess, welcher das TCS-System und den Trigger kontrolliert, schickt dann über das TCS-System das Kommando zum Zurücksetzten, bevor der Trigger erneut geöffnet wird.

Modifikation für die Auslese der Zählerkarten

Eine weitere Besonderheit des Crystal-Barrel/TAPS-Experiments sind die Scaler-Ereignisse, welche nur in festen Zeitabständen gelesen werden um die Totzeit der Datenakquisition gering zu halten. Da es mit dem TCS-System nicht möglich ist, bei bestimmten Ereignissen zusätzliche Informationen zu lesen, wird für die CATCH-Zählerkarten ein getrenntes System benutzt. Dabei wird ausgenutzt, dass die benutzten Zählerkarten nicht nur über das optische TCS-System gesteuert werden können, sondern auch mit ECL-Signalen über eine Reihe von Eingängen an der Vorderseite der Module. Dazu steht eine Kontrollbox zur Verfügung, welche ein Signal zum Zurücksetzen und ein Signal zum Auslesen empfängt und dieses an bis zu 9 CATCH-Zählerkarten weiterleitet. Das Signal des Anfangs der Extraktion des Beschleunigers wird dabei zum Zurücksetzen verwendet. Zur Auslese wird die vom Trigger gesendete Event-ID verwendet. Der Trigger wird dabei so konfiguriert, dass ein Kanal der Event-ID genau dann einen Wert zeigt, wenn es sich bei dem Ereignis um ein Scaler-Ereignis handelt. Dieser Kanal wird dann als Signal zum Auslesen benutzt. Das Gesamtsystem mit diesen Modifikationen ist in Abbildung 4.9 gezeigt.

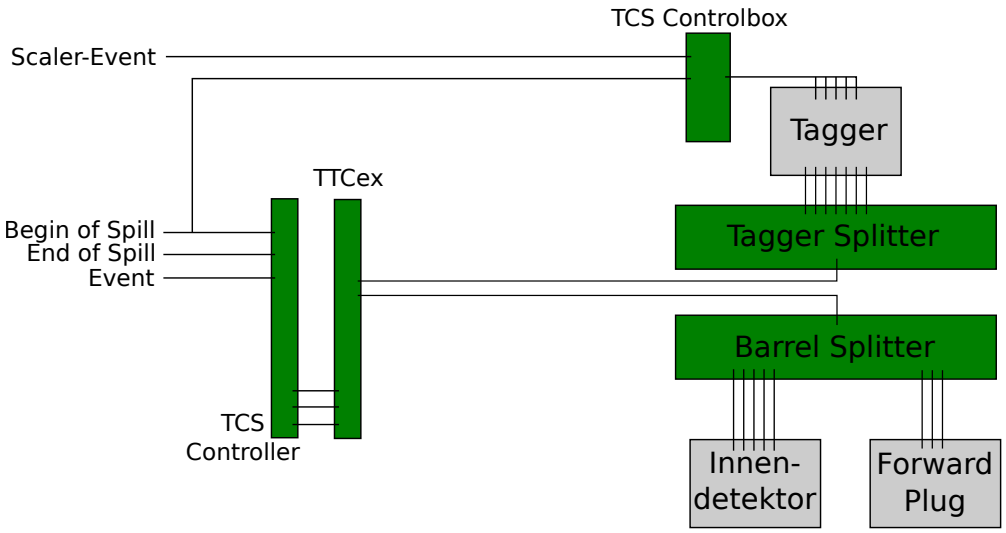


Abbildung 4.9.: **Das TCS System im Crystal-Barrel/TAPS-Experiment.** An den TCS-Master werden der Anfang und das Ende der Extraktion sowie das Triggersignal der ersten Triggerstufe gesendet. Die Signale werden dort mit einer festen Frequenz moduliert und an das TTCex-Modul geschickt. Dieses wandelt die Signale (Kanal A und B) in ein optisches Signal um, welches dann an zwei passive optische Splitter am Tagger und am Crystal-Barrel-Detektor weitergeleitet wird. Am Tagger wird das Signal dann an die TDC-Karten des Taggers, am Crystal-Barrel-Detektor an die TDC Karten des Innendetektors und des Forwardplugs weitergeleitet. Das Signal am Anfang der Extraktion wird zusätzlich mit einem *Scaler*-Ereignis-Signal an die TCS-Kontrollbox am Tagger geschickt. Von dort werden die Kontrollsignale als ECL-Signale an die Zählerkarten des Taggers geschickt.

5. Die Datenakquisition

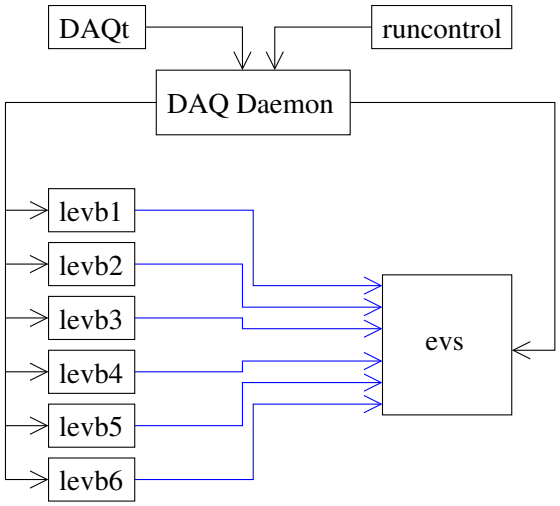


Abbildung 5.1.: **Überblick der einzelnen Komponenten der Datenakquisition.**

Die Datenakquisition besteht aus mehreren separaten Programmen die parallel auf verschiedenen Rechnern verteilt ausgeführt werden. Für jeden Subdetektor ist ein lokaler Eventbuilder (Kap 5.1) implementiert, der die Daten aus der Ausleseelektronik (ADCs, TDCs, Scaler, ...) ausliest und über das Netzwerk weiterreicht. Die Daten werden dann von einer zentralen Komponente gesammelt und auf Festplatten gespeichert. Diese Komponente wird als globaler Eventbuilder (Kap 5.3) bezeichnet. Gesteuert wird dieses System vom DAQ-Daemon (Kap 5.4.1), mit dem man sich entweder über ein Kommandozeilenprogramm (runcontrol) oder eine graphische Benutzeroberfläche (DAQt) verbinden kann. Ein Überblick der einzelnen Komponenten ist in Abbildung 5.1 gegeben. All hier vorgestellten Programme sind in der Programmiersprache *C++* geschrieben und benutzen die *boost*[1] Erweiterungen. Die DAQt basiert zusätzlich auf der *Qt*-Entwicklungsumgebung[2].

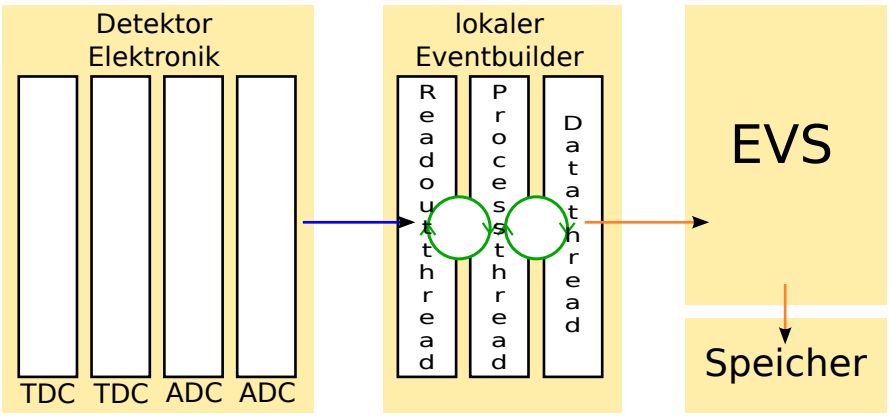


Abbildung 5.2.: **Überblick der einzelnen Komponenten der Datenakquisition.** Der *Readoutthread* liest über den VME-Bus (blau) die Informationen aus den Elektronikmodulen des Detektors. Diese werden dann über einen Ringpuffer (grün) zunächst an den *Processthread* und von diesem dann an den *Datathread* weitergegeben. Der *Datathread* schickt die Daten dann über das Netzwerk (orange) an den globalen Eventbuilder (EVS) der die Daten dann abspeichert.

5.1. Der lokale Eventbuilder

Für jede Detektorkomponente im Experiment gibt es einen lokalen Eventbuilder. Dieser ist dafür zuständig die Daten aus den Hardware-Modulen des Subdetektors auszulesen, vorzuverarbeiten und dann an den globalen Eventbuilder weiterzuschicken. Dafür werden drei voneinander unabhängige Prozesse benutzt, welche die Daten untereinander über Ringpuffer austauschen (Kapitel 5.1.6). Durch die Aufteilung in drei Prozesse, welche durch einen Puffer getrennt sind, muss die Datennahme nur für die Aufgaben welche wirklich eine Synchronität der Subdetektoren erfordern blockiert werden. Diese Aufgaben werden im *Readoutthread* (siehe Kapitel 5.1.1), dem ersten der drei Prozesse durchgeführt. Wenn der *Readoutthread* alle seine Aufgaben erfüllt hat und die Daten an den Puffer übergeben hat, wird die Datennahme sofort wieder freigegeben. Die weitere Aufgaben werden im *Processthread* (siehe Kapitel 5.1.2) durchgeführt und durch den *Datathread* (siehe Kapitel 5.1.3) an das Netzwerk übertragen. Die lokalen Eventbuilder werden mit der Bibliothek *daqbasesync* realisiert. Deren Hauptkomponenten befinden sich in den Klassen *CBaselevbSync*, *CReadoutThread*, *CProcessThread* sowie *CDataThread*. Im Folgenden werden diese Komponenten, ihr Datenaustausch untereinander, sowie die Steuerung der lokalen Eventbuilder, erklärt.

[1]Zusatzbibliotheken zur Erweiterung des C++ Standard - www.boost.org
[2]Plattformübergreifendes Framework zur Entwicklung von graphischen Oberflächen - qt.nokia.com

5.1.1. Der Readoutthread

Die Aufgabe des Readoutthreads ist es die Daten der Elektronikmodule der Subdetektoren zu lesen, wenn vom Trigger ein ‘‘Event’’-Signal an den zugehörigen Sync-Client geschickt wird. Im folgenden wird der Ablauf im Readoutthread beschrieben, wie er auch in Abbildung 5.3 gezeigt ist. Wenn die Datennahme gestartet wird werden zunächst die beiden Funktionen *PrepareRun* und *Reset* ausgeführt. In der Funktion *PrepareRun* werden dabei Daten aufgenommen, welche vor dem ersten Ereignis gespeichert werden sollen, wie zum Beispiel Pedestalwerte. Die Funktion *Reset* sorgt dafür, dass vor der eigentlichen Datennahme die Elektronik des jeweiligen Subdetektors korrekt konfiguriert und zurückgesetzt wird. Anschließend wartet der *Readoutthread* auf das ‘‘Event’’-Signal des Synchronisationssystems. Wenn dieses ein aufgetretenes Ereignis anzeigt wird zunächst die Datennahme blockiert indem über den Sync-Client das ‘‘Busy’’-Signal ausgeschaltet wird. Anschließend werden die ereignisspezifischen Informationen vom Sync-Client gelesen. Zu diesen zählt die Puffernummer, welche aus der Ereignisnummer gebildet und an alle Subdetektoren verteilt wird, sowie der Ereignistyp. Anschließend wird die Funktion *DataEvent* gestartet in welcher die Auslese der Elektronik des Subdetektors stattfindet. Falls der vom Trigger gesendete Ereignistyp ein spezielles Zählerereignis kennzeichnet wird anschließend noch die Funktion *ScalerEvent* aufgerufen. In dieser Funktion werden zusätzliche Informationen, wie zum Beispiel Zählerstände, welche nicht bei jedem Ereignis benötigt werden, gelesen. Die Daten werden dabei in den Funktionen *DataEvent* und *ScalerEvent* in einen Puffer geschrieben. Abschließend signalisiert der Prozess die Bereitschaft zur weiteren Datenaufnahme indem die Signale ‘‘busy’’ und ‘‘okay’’ gelöscht bzw. gesetzt werden. Nach dem nächsten Trigger wird dieser gesamt Prozess erneut gestartet. Zur weiteren Steuerung der Datennahme stehen zwei weitere Funktionen *BeforeWaitTrigger* und *AfterWaitTrigger* zur Verfügung. Diese beiden Funktionen werden direkt vor bzw. nach dem Warten auf die Triggerentscheidung ausgeführt. Dadurch wird es unter anderem einem der lokalen Eventbuilder ermöglicht den Trigger zu steuern. Die einzelnen Funktionen sind in Anhang 5.1.4 genauer beschrieben. Realisiert ist dieser Thread in der Klasse *CReadoutThread*.

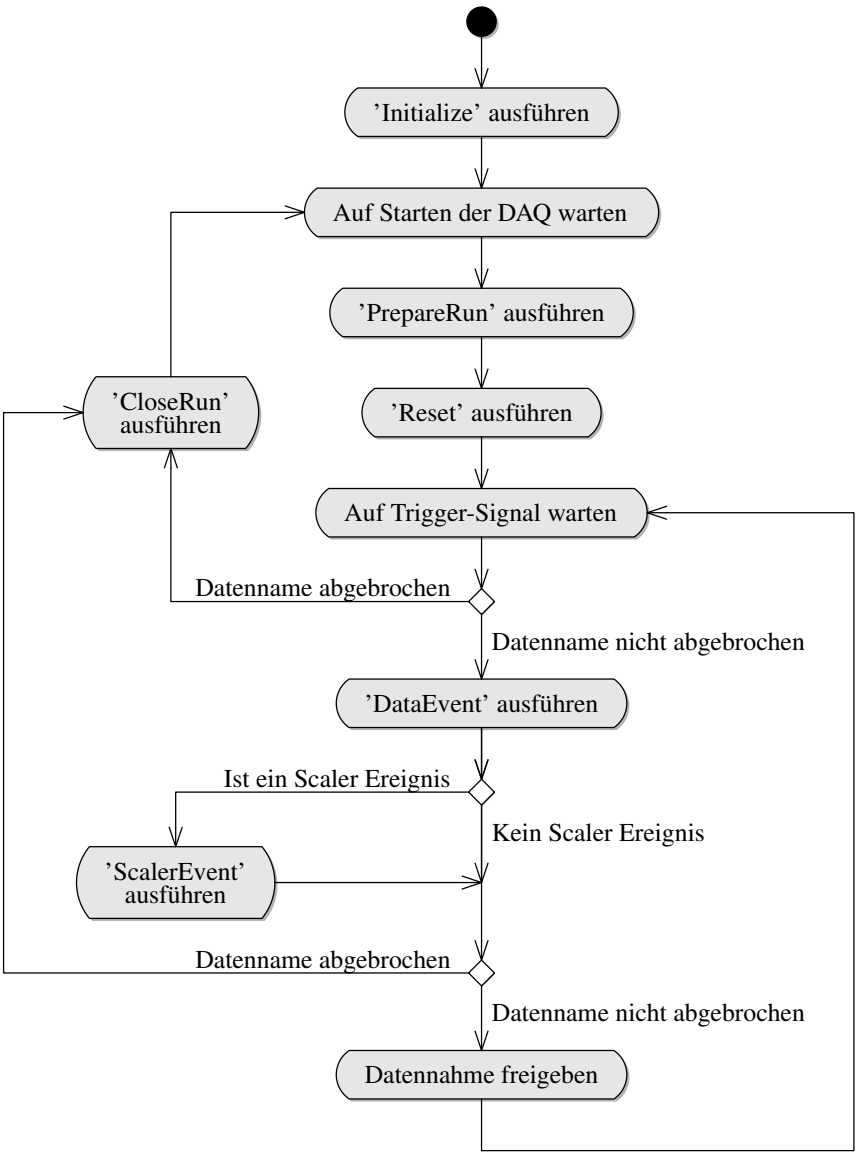


Abbildung 5.3.: **Der ReadoutThread.** Am Anfang führt der ReadoutThread die Funktion ''Initialize'' aus, dannach wartet er auf das Starten eines Datenblocks. Anschließend werden die Funktionen ''PrepareRun'' und ''Reset'' ausgeführt, bevor auf das erste Triggersignal gewartet wird. An dieser Stelle wird das erste Mal geprüft ob die Datennahme beendet wurde. Wenn dies nicht der Fall ist, wird die Funktion ''DataEvent'' und falls gefordert die Funktion ''ScalerEvent'' ausgeführt. Darin anschließend wird erneut geprüft ob die Datennahme beendet wurde. Falls dies nicht der Fall ist, wird die Datennahme über das Synchronisationssystem freigegeben und auf einen neuen Trigger gewartet. Falls die Datennahme beendet wurde wird die Funktion ''CloseRun'' ausgeführt und auf den Start des nächsten Datenblock gewartet.

5.1.2. Der Processthread

Da der Processthread parallel zum Readoutthread läuft und die Datennahme bereits vom Readoutthread wieder freigegeben wird, haben im Processthread durchgeführte Aufgaben keine direkte Auswirkung auf die Totzeit des Experiments. Daher können hier Aufgaben durchgeführt werden, welche eine gewisse Zeit benötigen und deshalb zu einer deutlichen Erhöhung der Totzeit der Datenakquisition führen würden, wenn sie im Readoutthread durchgeführt werden würden. Da der Processthread nicht vom Synchronisationssystem gesteuert wird, können dort keine Informationen des Ereignisses ausgelesen werden. Es ist jedoch möglich dort noch Änderungen an den aufgenommenen Daten vorzunehmen, wie zum Beispiel das Umsortieren von Kanälen einer Zählerkarte oder die Integralbildung der Werte eines FADCs[3]. Zusätzlich können auch Informationen gelesen werden, bei denen es nicht wichtig ist, ob sie dem korrekten Ereignis zugeordnet werden. Dies wären zum Beispiel Temperatur- oder Spannungswerte welche sich nicht auf der Zeitskala der Ereignisrate ändern.

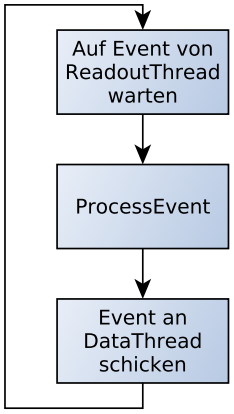


Abbildung 5.4.: Ablauf im ProcessThread. Der Ablauf ist im Text beschrieben.

Im Folgenden wird der Ablauf im Processthread beschrieben, wie er auch in Abbildung 5.4 gezeigt ist. Der Processthread wartet zunächst darauf, dass vom Readoutthread ein Puffer mit einem Ereignis gefüllt wird. Sobald dies der Fall ist, wird die Funktion ‘‘ProcessEvent’’ aufgerufen. In dieser Funktion können die oben beschriebenen Änderungen an den Rohdaten des Ereignisses vorgenommen werden. Nachdem die Funktion ‘‘ProcessEvent’’ beendet ist, wird das Datenobjekt an den Datathread weitergegeben. Realisiert ist dieser Thread in der Klasse *CProcessThread*.

5.1.3. Der Datathread

Der Datathread ist dafür zuständig die Daten an die zentrale Sammelstelle, den globalen Eventbuilder, weiterzugeben. Dazu baut der Datathread in jedem lokalen Eventbuilder beim Starten eine Verbindung zum globalen Eventbuilder auf. Sobald dies geschehen ist werden zunächst grundlegende Informationen über den Subdetektor übertragen, die nur einmal pro Datenblock anfallen. Teil dieser Informationen sind zum Beispiel

[3] Abkürzung für Flash Analog Digital Converter. Ein ADC welcher das Eingangssignal schnell in ein digitales Signal konvertieren kann. Dies ermöglicht das Abtasten des Eingangssignals mit einer festen Frequenz.

Pedestal[4]-Werte von ADC Kanälen und Informationen über den Aufbau der von dem lokalen Eventbuilder gesammelten Daten. Gesammelt werden diese Informationen in der Funktion *PrepareRun*. Zusätzlich wird noch eine eindeutige Identifikationsnummer für den jeweiligen Subdetektor übertragen. Während der eigentlichen Datennahme wartet der Datathread darauf, dass ein Ereignis vom Processthread bearbeitet wurde. Sobald dieser ein Ereignis in den Puffer zwischen diesen beiden Threads füllt, wird es vom Datathread an den globalen Eventbuilder über die bestehende Netzwerk-Verbindung weiter geschickt. Durch die Pufferung zwischen den Threads können die anderen beiden Threads unabhängig weiterlaufen, während der Datathread die gesammelten Daten überträgt. Dies ist besonders wichtig, da es bei der Übertragung über das Netzwerk auch zu kurzfristigen Störungen kommen kann, die in diesem Fall die Datennahme nicht beeinträchtigen. Dieser Thread ist in der Klasse *CDataThread* realisiert.

5.1.4. Beschreibung der Funktionen des lokalen Eventbuilders

In diesem Abschnitt werden die einzelnen Funktionen die im lokalen Eventbuilder zur Verfügung stehen genauer beschrieben werden.

Initialize

Die Funktion *Initialize* wird einmal beim Starten des lokalen Eventbuilders aufgerufen. Da der lokale Eventbuilder während einer Strahlzeit nur dann neu gestartet wird wenn ein Fehler aufgetreten ist, können in dieser Funktion keine Aufgaben ausgeführt werden, welche regelmäßig durchgeführt werden sollen. Die *Initialize*-Funktion wird daher typischerweise dafür genutzt um Variablen und Klassen anzulegen, die für den Betrieb benötigt werden. Hier werden zum Beispiel alle Klassen zur Steuerung und Auslese der Elektronikmodule angelegt. Zusätzlich können hier auch schon Konfigurationen vorgenommen werden, welche sich bei unterschiedlichen Konfigurationen der Datennahme nicht ändern. Dies sind üblicherweise zum Beispiel Schwellen von Diskriminatoren oder die Konfiguration von FPGA-Modulen.

Reset

Im Gegensatz zur *Initialize*-Funktion wird die *Reset*-Funktion vor dem Starten jedes Datenblocks ausgeführt. Um das regelmäßige Ausführen dieser Funktion und der Funktion PrepareRun zu garantieren wird die Dauer eines Datenblocks normalerweise auf einen Wert unter 30 Minuten gesetzt. Dadurch können hier außerdem Konfiguration ausgeführt

[4] Das Pedestal ist der Wert der vom ADC ausgegeben wird wenn kein Eingangssignal auf den jeweiligen Kanal gegeben wird. Dieser Wert wird zum Eichen der ADC Kanäle benötigt.

werden, welche regelmäßig durchgeführt werden sollen. Außerdem können hier alle Konfigurationen durchgeführt werden die nicht über die gesamte Strahlzeit gleichbleiben, sondern sich für jeden Datenblock ändern können. Dazu wird von dem Programm, welches die Datennahme steuert (siehe Kap. 5.4), eine für den lokalen Eventbuilder passende Konfigurationsdatei übergeben, welche dann in der Reset Funktion ausgewertet werden kann. Entsprechend dieser Konfiguration werden die Ausleseelektronik dann konfiguriert. Dadurch ist es zum Beispiel möglich für jeden Datenblock unterschiedliche Diskriminatorschwellen zu setzen. Des Weiteren werden in dieser Funktion noch alle Ausleseelektronik zurückgesetzt und für den Run vorbereitet.

PrepareRun

Ähnlich zur Funktion *Reset* wird auch diese Funktion vor dem Starten jedes Datenblock ausgeführt. Die *PrepareRun* Funktion wird jedoch ausgeführt, bevor der DataThread eine Verbindung zum globalen Eventbuilder aufgebaut hat. Somit können alle Informationen die in dieser Funktion von den Modulen bezogen werden, bei der Verbindung zum globalen Eventbuilder im ‘‘TableEvent’’ mitgeschickt werden. So kann vor jedem Datenrun zum Beispiel eine kurze Bestimmung der Pedestalwerte[5] durchgeführt werden und die Ergebnisse im TableEvent gespeichert werden. Außerdem ist es möglich hier eine Kanalzuordnung zu speichern (LUT[6]).

Dataevent

Diese Funktion wird bei jedem aufzunehmenden Ereignis aufgerufen. In ihr sollen alle Informationen aus den Elektronikmodulen des Subdetektors ausgelesen werden. Dabei wird zunächst mit der Funktion *getDataPtr()* ein Zeiger angefordert, der auf einen vorher reservierten Speicherbereich zeigt wohin die Daten geschrieben werden können. Nachdem die Daten dorthin geschrieben wurden werden noch die benötigten Informationen dazu gespeichert. Dies sind ein Name, eine Version, sowie die Größe der gespeicherten Daten. Dazu wird die Funktion *addData(Bankname, Bankversion, Bankgröße)* benutzt. Um die Daten besser zu strukturieren, können von einem lokalen Eventbuilder auch mehrere Datenblöcke mit unterschiedlichen Namen angelegt werden. Dazu wird nach dem ersten Aufruf der *addData*-Funktion ein weiteres Mal *getDataPtr()* ausgeführt, die Daten an den von dieser Funktion erhaltenen Ort geschrieben, und das ganze mit der Funktion *addData* abgeschlossen. Das Ganze kann prinzipiell beliebig oft wiederholt werden. Damit der reservierte Speicherplatz nicht überschritten wird kann die zur Verfügung stehende Größe

[5] Ein Pedestal ist der Wert den ein ADC aufnimmt, wenn er kein Signal am Eingang erhalten hat. Dieser Wert ist nötig zur Bestimmung der korrekten Eintrages.
[6] engl. für Lookup Table - Tabelle zum nachschauen

mit der Funktion *getFreeBufferSize()* abgefragt werden. Es muss dann sichergestellt werden, dass nicht mehr Bytes als dort angegeben an den reservierten Speicherbereich geschrieben wird.

Scalerevent

Im Gegensatz zur *DataEvent*-Funktion wird diese Funktion nur beim Eventtyp *Scalerevent* aufgerufen. Dieser Eventtyp wird in festgelegten Zeitabständen (typischerweise alle 50ms) vom Trigger gesetzt. Dadurch ist es möglich Informationen die nicht bei jedem Ereignis benötigt werden, wie zu Beispiel Zählerstände, nur in bestimmten Zeitabständen zu lesen, und dadurch die Totzeit zu reduzieren.

ModuloFunction

Ähnlich wie die *ScalerEvent*-Funktion wird diese Funktion nicht bei jedem Ereignis ausgeführt. Wann dies passiert wird jedoch bei der *ModuloFunction* nicht vom Trigger entschieden, sondern wird vom lokalen Eventbuilder festgelegt. In diesem wird eine Variable *modulo* definiert, wie oft die *ModuloFunction* ausgeführt wird. Sie wird dabei direkt vor der *BeforeWaitTrigger*-Funktion ausgeführt. Während die *ModuloFunction* ausgeführt wird ist der Trigger bereits freigegeben, so dass keine zusätzliche Totzeit durch diese Funktion generiert wird. Das nächste Ereignis kann jedoch abhängig von der Ereignisrate und der Dauer der *ModuloFunction* schon auftreten während die Funktion noch ausgeführt wird. Sie wird hauptsächlich für Aufgaben verwendet, die in festen Ereignisabständen durchgeführt werden. Dies sind zum Beispiel die Steuerung des Lichtpulsers (Kapitel 2.16) und des Bremsstrahlradiators (Kapitel 2.2).

CloseRun

Am Ende jedes Datenblocks wird diese Funktion ausgeführt. Dadurch können zum Beispiel am Ende eines Datenblocks zusammenfassende Informationen ausgegeben werden.

Uninitialize

Ebenso wie die *Initialize*-Funktion beim Starten des lokalen Eventbuilders gestartet wird, wird diese Funktion beim Beenden aufgerufen. Hier können die angelegten Klassen gelöscht werden.

5.1.5. Der Datenspeicher

In diesem Kapitel wird die Struktur, in welcher die Daten der lokalen Eventbuilder gespeichert werden, erläutert. Die dafür zuständige Klasse ist *CRawDataBuffer*. In dieser Klasse wird ein Speicherbereich reserviert, dessen Größe variabel an die benötigten Verhältnisse angepasst werden kann (siehe Anhang E.2). Die Größe beträgt im Normalfall 32 KBytes. In diesen Speicherbereich werden bei der Datennahme in den Funktionen *DataEvent*, *ScalerEvent* und *ProcessEvent* die Informationen der Subdetektoren geschrieben. Dabei werden die Daten für eine einfachere Verarbeitung in mehrere Bänke aufgeteilt. Eine Bank bezeichnet dabei einen Block von Daten der zusammengehörige Informationen enthält. Dabei wird innerhalb eines Subdetektors nach unterschiedlichen Teilkomponenten und Typ der Information (z.B. TDC oder ADC) unterschieden. Jeder Bank wird dabei nach einem festgelegten Schema ein aus 4 alphanumerischen Zeichen bestehender Name zugeordnet. Die ersten beiden Zeichen beschreiben dabei die Teilkomponente, die dritte Stelle den Banktyp (siehe Anhang E.3) und die letzte Stelle ist eine laufende Nummer, falls es mehrere Bänke von gleichem Typ und gleicher Teilkomponente gibt. Der Bankname "FPT0" würde zum Beispiel die erste Bank der TDC Informationen des Forwardplugs beschreiben. Die vergebenen Banknamen und ihre Inhalte sind in Anhang E.3 beschrieben. Zusätzlich zu den Datenbänken, die aus den Detektoren gelesen wurden, werden in den Speicherbereich noch einige Zusatzinformationen geschrieben. In Tabelle 5.1 ist die Struktur des Speicherbereiches des CRawDataBuffers aufgeführt.

Zusätzlich zu diesen Informationen werden noch für jede gespeicherte Bank die Position der Header, die Position der Daten, der Name der Bank und ob die Bank übers Netzwerk gesendet werden soll in einer getrennten Struktur mit dem Namen BufferManager gespeichert. Dies ist notwendig, damit der ProcessThread die Daten über den Banknamen erreichen und ändern kann. Des Weiteren werden noch einige Funktionen angeboten, die den Zugriff auf den Speicherbereich und die Informationen in der Klasse CRawDatabuffer ermöglichen. Die wichtigsten dieser Funktionen werden in Anhang E.1 erläutert.

5.1.6. Interprozesskommunikation

Damit die drei oben beschriebenen Prozesse (ReadoutThread, ProcessThread und DataThread) unabhängig voneinander agieren können und die Ereignisse von den Prozessen sequenziell abgearbeitet werden können, wird nicht nur ein Speicher der in Kapitel 5.1.5 beschriebenen Form benutzt, sondern es werden zunächst eine vorbestimmte Zahl von Speichern angelegt. Diese Speicher werden im folgenden Puffer genannt. Die Anzahl der Puffer beträgt im Normalfall 32, kann jedoch variabel an die vorliegenden Verhältnisse angepasst werden (siehe Anhang E.2). Um den Zugriff der Prozesse auf diese Puffer

CRawDataBuffer	Datasize (Kompletter Puffer in Bytes)			
	CPUid	Eventstatus	globalBufferNumber	localBufferNumber
	softInterruptCounter			
	hardInterruptCounter			
	Lifetime			
	DeadtimePrevEvent			
	EventTime			
	ReadoutTime			
CRawDataBank 1	header1 (Bankname)			
	header2 (Bankgröße ohne header1/2 in Bytes)			
	size (Bankversion | Bankgröße ohne header1/2 in 16 Bit Wörtern)			
	Daten			
	...			
CRawDataBank 2	header1			
	header2			
	size			
	Daten			
	...			
...				
CRawDataBank n	header1 (Bankname)			
	header2 (Bankgröße ohne header1/2 in Bytes)			
	size (Bankversion | Bankgröße ohne header1/2 in 16 Bit Wörtern)			
	Daten			
	...			

Tabelle 5.1.: **Struktur des CRawDataBuffer.** Eine Zeile entspricht 32 bit belegtem Speicher.

zu ermöglichen, wird zunächst ein Zeiger auf jeden dieser Puffer angelegt. Das hat den Vorteil, dass zwischen den Prozessen nur dieser Zeiger ausgetauscht werden muss, was deutlich schneller ist als, wenn der komplette Puffer übergeben werden müsste. Um nun sicherzustellen, dass die Puffer in der korrekten Reihenfolge abgearbeitet werden und den Prozessen eine Möglichkeit zu geben, schnell zu erkennen ob sie etwas zu bearbeiten haben, wurden drei Puffer geschaffen in denen die Zeiger gespeichert werden. Zunächst gibt es den *freeBuffer* in dem alle Zeiger auf Puffer gespeichert werden in denen keine Daten gespeichert sind. Im *usedBuffer* werden dann alle Zeiger gespeichert die auf Puffer zeigen, die vom DataThread mit Daten gefüllt wurden, aber noch nicht vom ProcessThread bearbeitet wurden. Der *sendBuffer* enthält dann diejenigen Zeiger, die schon vom ProcessThread prozessiert wurden. Alle drei Puffer basieren auf der Klasse *CPointerBuffer*, welche auf der Klasse *BoundedBuffer* der boost-Erweiterung basiert.

Im Folgenden wird der zeitliche Ablauf der Zugriffe der einzelnen Prozesse auf diese drei Objekte genauer beschrieben. Der Ablauf ist auch in Abbildung 5.5 schematisch gezeigt. Am Anfang sind alle Puffer noch nicht gefüllt, daher befinden sich alle Zeiger im *freeBuffer*. Der ReadoutThread entnimmt nun einen Zeiger aus dem *freeBuffer*,

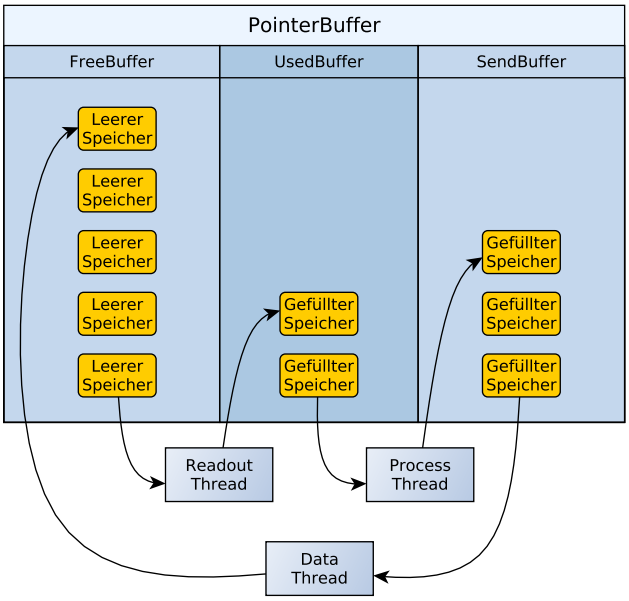


Abbildung 5.5.: **Schematische Darstellung des Datenaustausches zwischen den Prozessen.** Der zeitliche Ablauf ist genauer im Text beschrieben.

füllt seine Daten an den Puffer, auf welchen der Zeiger zeigt und hängt den Puffer an den *usedBuffer* an. Der ProcessThread warten in der Zwischenzeit darauf, dass im *usedBuffer* ein neuer Zeiger auftaucht. Sobald das der Fall ist, entnimmt er den Zeiger und bearbeitet den Inhalt. Anschließend hängt er den Zeiger dann an den *sendBuffer* an. Analog wartet der DataThread auf einen neuen Zeiger im *sendBuffer*. Nachdem er den Zeiger entnommen hat, sendet er die Daten aus dem Puffer, auf den der Zeiger verweist, und löscht den Inhalt mit der *reset()*-Funktion. Anschließend hängt er den Zeiger wieder an den *freeBuffer* an. Falls bei diesem Vorgang von einem Prozess noch nicht alle Zeiger verarbeitet wurden, werden trotzdem erst die anderen Puffer abgearbeitet, da neue Zeiger stets hinten angehängt werden. Dadurch wird sichergestellt, dass alle Puffer in der richtigen Reihenfolge bearbeitet werden.

5.1.7. Steuerung der lokalen Eventbuilder

Da die einzelnen lokalen Eventbuilder auf unterschiedlichen Rechnern über den gesamten Experimentbereich verteilt ausgeführt werden, müssen sie von einer zentralen Stelle gesteuert werden. Wie schon beim vorhergehenden Datenakquisitionssystem des CB-ELSA Experiments [Sch04] basiert diese Steuerung auf einem auf ASCII Kommandos basierenden Protokoll. Dabei werden Befehle im Textformat über das Netzwerk

übertragen, was es ermöglicht mit frei verfügbaren standardisierten Programmen Befehle an die lokalen Eventbuilder zu übertragen. Das alte System basierte jedoch auf der Common C++[7]-Erweiterung und war fest in die Strukturen der Datenakquisition eingebunden. Durch die im Rahmen dieser Arbeit erfolgte Neuimplementierung auf Basis der Boost[8]-Erweiterung, konnte die Geschwindigkeit der Datenübertragung erhöht und die Systembelastung verringert werden. Außerdem befinden sich die Klassen die zur Kommunikation verwendet werden nun in einer separaten Bibliothek, so dass sie von beliebigen Programmen verwendet werden können. Dadurch reduziert sich der Aufwand eine Kommunikation über das Netzwerk aufzubauen deutlich.

Die Implementierung besteht dabei aus einer Server-Klasse, die Befehle entgegen nehmen kann, sowie Antworten auf Anfragen liefert, sowie einer Client-Klasse mit der sich diese Befehle senden lassen und welche gegebene Antworten liest. Beide Klassen, *CTCPBoostClient* für den Client Teil, sowie *CTCPBoostServer* für den Server Teil, sind in der Bibliothek *CTCPBoostConnection* enthalten. Der Client Teil wird dabei hauptsächlich von den kontrollierenden Programmen der Datenakquisition benutzt (siehe Kapitel 5.4). Der Server-Teil wird vom lokalen Eventbuilder benutzt und ist so aufgesetzt, dass sich mehrere Klienten gleichzeitig mit dem Server verbinden und Befehle absetzen können. Um die für die Datenakquisition nötige Funktionalität herzustellen werden die von der *CTCPBoostServer*-Klasse empfangenen Kommandos an die Funktion *processCommand* des lokalen Eventbuilders weitergeleitet. Diese Funktion wertet dann die empfangenen Kommandos aus und führt je nach Inhalt unterschiedliche Funktionen aus. Von der Funktion *ProcessCommand* kann dabei entweder direkt eine Antwort zurückgegeben werden, oder, wenn ein Spezialkommando gegeben wird, wird diese Kommando an eine weitere Funktion weitergereicht.

Diese Spezialkommandos sind ‘‘SYNC’’, ‘‘SCAL’’ und ‘‘USER’’. Beim ‘‘SYNC’’-Kommando wird die Funktion *callSyncCommand* ausgeführt, welche Kommandos an das Synchronisationssystem geben kann. In der Funktion *callScalerCommand*, welche vom ‘‘SCAL’’-Kommando aufgerufen wird, können verschiedene Zähler ausgelesen werden. Beim Kommando ‘‘USER’’ wird das Kommando an die Funktion *parseCommands* des lokalen Eventbuilder weitergegeben, in welcher für jeden lokalen Eventbuilder beliebige Kommmandos definiert werden können. Alle allgemeinen Kommandos sowie die implementierten ‘‘USER’’-Kommandos einiger lokaler Eventbuilder sind in Anhang E.4 aufgeführt. Eine eventuelle Antwort der ausgeführten Befehle wird in einem einzeiligen Text über die Telnet-Schnittstelle zurück gegeben. Zusätzlich wird abhängig vom Erfolg

[7] Frei als Quellcode erhältliche portable Erweiterungsbibliotheken für die C++-Programmiersprache. https://www.gnu.org/software/commoncpp/

[8] Frei als Quellcode erhältliche portable Erweiterungsbibliotheken für die C++-Programmiersprache. https://www.boost.org

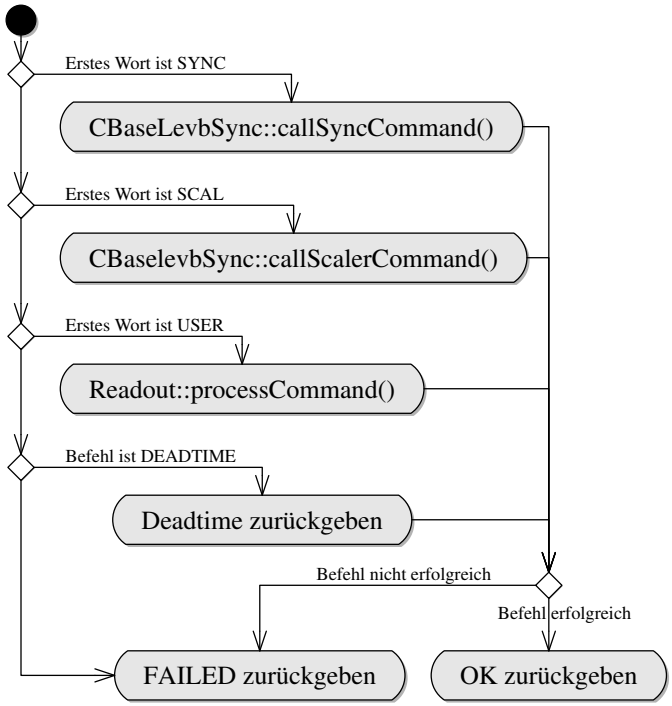


Abbildung 5.6.: **Schematische Darstellung des Ablaufs in der processCommand Funktion.** Zunächst wird geprüft ob der Befehl mit den Stichworten ‘‘SYNC’’, ‘‘SCAL’’ oder ‘‘USER’’ anfängt. In diesem Fall wird die entsprechende Funktion aufgerufen, die den Rest des Befehls auswertet. Anschließend, werden noch eigene Kommandos geprüft, hier Beispielhaft ‘‘DEADTIME’’ und direkt ausgewertet. Falls kein übereinstimmendes Kommando gefunden wurde wird ‘‘Failed’’ zurück gegeben Ansonsten wird je nach Erfolg des Kommandos ‘‘OK’’ oder ‘‘FAILED’’ zurückgegeben.

des Befehls immer der Text ‘‘OK’’ oder ‘‘FAILED’’ zurückgegeben. Falls vom Benutzer ein Kommando gegeben wird, das nicht implementiert ist, wird ebenfalls ein ‘‘FAILED’’ zurück gegeben.

5.2. Die lokalen Eventbuilder des Crystal-Barrel/TAPS-Experiments

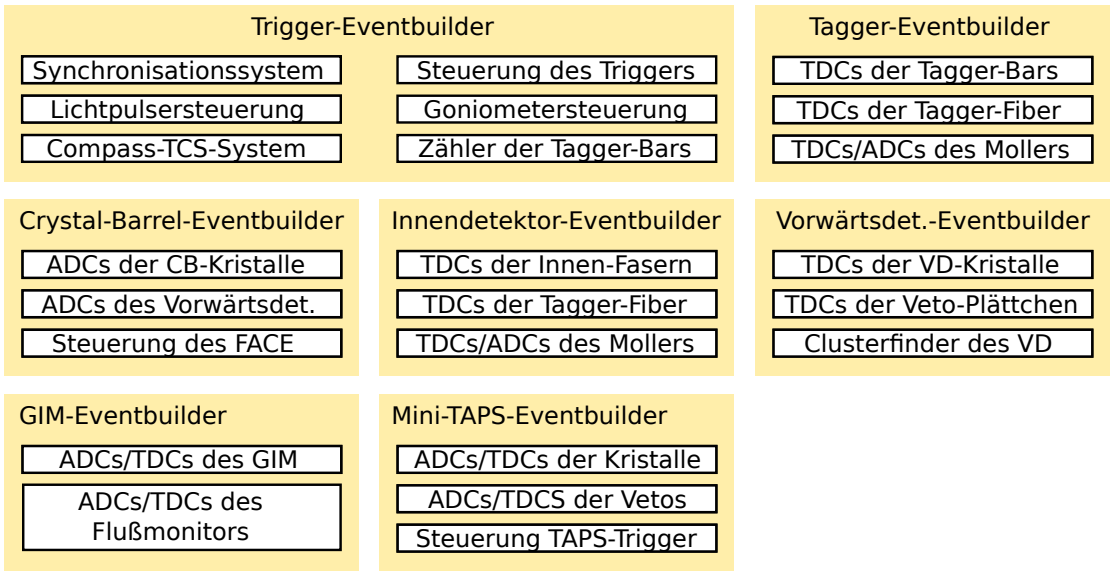


Abbildung 5.7.: **Überblick über die einzelnen lokalen Eventbuilder und ihre Aufgaben.**

Im diesem Kapitel werden die 13 lokalen Eventbuilder des Crystal-Barrel/TAPS-Experiments mit ihren speziellen Aufgaben beschrieben. Die 6 Eventbuilder der Mini-TAPS Detektors sind dabei zu einem Abschnitt zusammengefasst. Die Beschreibung der jeweiligen Detektoren befindet sich in Kapitel 2. Die Beschreibung der Datenstruktur der lokalen Eventbuilder ist in Anhang E.3 zu finden.

5.2.1. Trigger

Der Trigger-Eventbuilder steuert und konfiguriert das zentrale Triggermodul. Dabei wird zunächst vor jedem Datenblock in der Funktion *reset* die gewünschte Triggerkonfiguration geladen und danach die FPGA des Triggermoduls (siehe Kapitel 3) neu geschrieben und konfiguriert. Während der Datennahme wird dann, wie in Kapitel 4.1.2 beschrieben, die Puffernummer übertragen, der Status der anderen Eventbuilder überwacht, der Trigger geöffnet, sowie ein schnelles Zurücksetzen der an das optische TCS-System (siehe Kapitel 4.3) Komponenten angefordert. Dabei wird zunächst in der Funktion *BeforeWaitTrigger* gewartet bis alle Subdetektoren bereit zur Datennahme sind. Dann wird eine neue Puffernummer an die Subdetektoren verteilt und anschließend der Trigger geöffnet. In der Funktion *waitTrigger* wird nun abgewartet bis ein Trigger

ausgelöst wurde oder ein schnelles Zurücksetzen angefordert wurde. Im letzteren Fall wird dieses wie in Kapitel 4.3.2 beschrieben abgesetzt und der Trigger erneut geöffnet. Nach der Auslese der Daten werden diese Schritte wiederholt, falls die Datennahme nicht beendet wurde. Die Daten, die in diesem Subdetektor gelesen werden, sind zunächst die Triggerinformationen, welche aus einigen Zählern und verschiedenen Bitmustern bestehen. Diese Daten werden in die beiden Bänke *TRS0* und *TRC0* geschrieben. Der Aufbau dieser Bänke wird in Anhang E.3 beschrieben. Zusätzlich werden noch die Zähler der Tagger Latten gelesen (*TNS1*). Diese befinden sich bei der Trigger Elektronik, weil sie Treffer in den Detektoren nur zählen, während der Trigger auf ein passendes Ereignis wartet. Das dafür benötigte Signal wird von der Triggerelektronik erzeugt. Wenn dieses Signal zusätzlich noch zum Tagger geführt werden müsste, würde ein deutlich längere Verzögerung der Signale der Tagger-Latten benötigt. Die einlaufenden Signale sind dabei nicht Kanalweise sortiert, so dass im lokalen Eventbuilder noch eine Umsortierung erfolgt.

Des weiteren befindet sich im Trigger-Eventbuilder noch die Steuerung des Lichtpulsers und der Goniometerscheibe. Dies ist notwendig, da es für die Steuerung diese beiden Komponenten nach festen Ereigniszahlen Steuerbefehle gesenden werden müssen. Während eines Datenruns wird dafür die *ModuloFunction* benutzt, welche nach einer festgelegten Zahl von Ereignissen ausgeführt wird. In dieser Funktion wird dann die Filterkombination oder Goniometerposition geändert. Dies garantiert eine konstante Ereigniszahl für jede Einstellung. Die dafür notwendige Konfiguration wird dabei aus einer XML[9]-Datei eingelesen. Für die Goniometerscheibensteuerung wird eine Direktverbindung über das Netzwerk zum Steuergerät hergestellt. Die Steuermodule des Lichtpulsers befinden sich beim Eventbuilder des Forwardplugs und lassen sich über diesen steuern. Daher werden die Steuerbefehle zunächst an diesen Eventbuilder geschickt, der diese Befehle dann an die Steuerelektronik weiterleitet. Für das Goniometer werden die aktuellen Positionswerte in der Bank *GOC0* mit in die Daten geschrieben. Die Lichtpulserwerte werden direkt vom Eventbuilder des Forwardplugs geschrieben.

5.2.2. Tagger

Der Tagger Subdetektor besteht aus drei Teilkomponenten. Zunächst gibt es den Tagger selber, der noch einmal in Latten und Fasern unterteilt ist (Kapitel 2.3). Daneben gibt es noch den Møller-Detektor (Kapitel 2.4). Sowohl die Latten als auch die Fasern des

[9] eXtensible Markup Language - engl. für erweiterbare Auszeichnungssprache. Vom World Wide Web Consortium herrausgegebene Metasprache für die hierarchische Darstellung von Daten in Textdokumenten.

Taggers werden dabei von CATCH[10]-TDC-Karten digitalisiert. Es handelt sich dabei um Karten die auch vom Compass Experiment eingesetzt werden, und über ein eigenes optisches Trigger-Control-System verfügen (siehe auch Kapitel 4.3). Die ersten 2 TDC Karten digitalisieren dabei die 96 Kanäle der Latten, sowie 6 Kanäle des Møller-Detektors und zwei Zeitreferenzen, die vom Trigger abgeleitet werden. Auf den restlichen 4 TDC Karten befinden sich die 480 Kanäle der Tagger Fasern, sowie eine Zeitreferenz und ein Signal das aus der ELSA Teilchenstruktur abgeleitet wird.Die 480 Kanäle der Tagger Fasern werden zusätzlich noch auf CATCH-Scaler Karten geführt bevor sie an die TDC Karten geleitet werden. Die Eingänge der Scaler-Karten sind dabei nicht terminiert, so dass dies mit nur einem Kabel geschehen kann. Bei den CATCH-Scalerkarten handelt es sich um Module, die aus dem vorherigen CB-ELSA Experiment vor dem letzten Umbau übernommen wurden. Für den damaligen Aufbau wurden die Karten modifiziert, da dort kein optisches Trigger-Control-System zur Verfügung stand. Deswegen sind die CATCH Scaler Karten direkt an eine separate TCS-Kontrollbox angeschlossen, welche sich in einem NIM-Crate des Tagger befindet. Alle TDC Daten des Taggers und des Møller Detektors sind in der Bank *TNT0* gespeichert. Die Zähler Informationen der Tagger Fasern werden in der Bank *TNS2* gespeichert. Die 6 TDC Kanäle des Møller Detektors setzen sich aus den 4 Detektoren zusammen, sowie dem Møller-Triggersignal jeweils in Koinzidenz mit den beiden Polarisationsrichtungen der gestreuten Møller-Elektronen. Zusätzlich wird noch eine ADC-Karte (12 Kanal 10-bit ADC vom Typ 2249A der Firma Lecroy) mit den Energieinformationen der vier Detektoren gelesen, sowie zwei Zählerkarten (24-bit Zähler vom Typ 2551 der Firma Lecroy), die jeweils wieder mit einer der beiden Polarisationsrichtungen freigeschaltet werden. Da der CAMAC-Kontroller der für die Taggerelektronik zuständig ist nur maximal 16bit in einem CAMAC-Zyklus lesen kann, werden die oberen 8bit der 24-bit Zählerkarten in einem speziell dafür entwickelten Modul zwischengespeichert und in einem weiteren CAMAC-Zyklus gelesen. Die Zählerdaten werden in der Bank *TNS0*, die Energieinformationen in der Bank *MOA0* gespeichert.

5.2.3. Crystal-Barrel 1

Der Subdetektor zur Auslese des Crystal-Barrels, wurde aufgrund der hohen Zahl der Kanäle in zwei Teile geteilt um die Ausleserate zu erhöhen. Der erste Teil, der in diesem Abschnitt beschrieben wird, liest die zum Beschleuniger zeigende Hälfte des Crystal-Barrels aus. Da die andere Hälfte zwei Kristallringe mehr hat (siehe Kapitel 2.9) werden zusätzlich noch 40 Kanäle dieser Hälfte mit ausgelesen. Dadurch ist die Anzahl der

[10] Abkürzung für COMPASS Accumulate, Transfer & Control Hardware (engl. für COMPASS Module zum Zusammenführen, Transferieren und Kontrollieren).

ADC-Karten in beiden Hälften identisch, was die Auslesezeiten angleicht. Es werden dabei über 8 ADCs verteilt 670 Kanäle gelesen (630 aus der zum Beschleuniger zeigenden Hälfte, 40 aus der anderen Hälfte). Bei den ADCs handelt es sich um Dual-Range[11] FastBus[12]-ADCs vom Typ 1885F der Firma LeCroy. Die beiden Bereiche des ADCs werden dabei benutzt um eine verbesserte Energieauflösung für nierdrige Energien zu erreichen. Dazu wird jeder Kanal zweimal ausgemessen. Ein Kanal wird dabei direkt digitalisiert (Low-Range) der andere Kanal wird erst abgeschwächt bevor er digitalisiert wird (High Range). Die ADC-Karten haben dabei 96 Eingangskanäle und für beide Bereiche jeweils eine Auflösung von 12-bit.

Eine weitere Besonderheit diese Evenbuilders ist, dass sich die Elektronik in einem FastBus-Rahmen befindet um die ADC-Karten ansprechen zu können. Damit die VME-CPU mit diesen FastBus Karten kommunizieren kann, ist ein spezieller Einschub erforderlich. Es handelt sich dabei hier um das VME-zu-FastBus Interface SIS4100 der Firma Struck. Da es sich um FastBus Karten handelt ist noch ein zusätzliches Modul zur Verbindung mit der VME-CPU des Lokalen Eventbuilders nötig. In diesem Fall handelt es sich um das VME-zu-FastBus Interface SIS4100 der Firma Struck. Zusätzlich wird in diesem Eventbuilder noch der in Kapitel 3.4 beschriebene FACE gelesen und gesteuert. Die Bankstruktur der beiden Bänke *X1A0* (ADC Werte der Crystal-Barrel-Kristalle) und *FAC0* (Informationen des FACE) ist in Kapitel E.3 aufgeführt. Vor Beginn der Datennahme wird in diesem Subdetektor ein sogenanntes Table Event ausgeführt in dem die Pedestal Werte der 8 ADCs sowohl für den High-Range als auch den Low-Range bestimmt werden. Die Position und Breite dieser Pedestal werden dabei in der Bank *X1TB* gespeichert.

5.2.4. Crystal-Barrel 2

Mit der zweite Hälfte der Crystal-Barrel Auslese werden die Energieinformationen der restlichen Kanäle der vom Beschleuniger wegzeigenden Hälfte sowie der Kristalle des Forwardplugs gelesen. Es werden dabei ebenfalls 8 ADCs des Typs 1885F der Firma LeCroy verwendet um die 560 Kanäle der Crystal-Barrel-Hälfte und die 90 Kanäle des Forwardplugs zu digitalisieren. Der achte ADC ist dabei exklusiv mit den Kanälen des Forwardplugs bestückt. Ebenso wie bei der anderen Hälfte werden die FastBus Module über SIS4100 Module der Firma Struck angesteuert. Die Daten der ADCs werden in die Bank *X2A0* geschrieben. Analog zum Eventbuilder der anderen Crystal-Barrel-Hälfte gibt es auch hier wieder ein Table Event am Anfang jedes Datenruns, in dem die Pedestal Daten der beiden ADC Bereiche in die Bank *X2TB* gespeichert werden.

[11] engl. für zwei Bereiche
[12] Kommunikationsbus für Ausleseelektronik in der Teilchenphysik. Spezifiziert in IEEE 960-1986

5.2.5. Innendetektor

Ähnlich dem Tagger Subdetektor, werden die Signale des Innendetektors nach dem Diskriminator auf CATCH TDCs geführt. Die 513 Kanäle werden dabei von 5 TDC Modulen digitalisiert. Das fünfte TDC Modul liest dabei noch einige zusätzliche Kanäle (siehe Anhang E.3.5) zur Kontrolle des Triggersignals des Innendetektors aus. Gesteuert werden diese TDCs vom gleichen optischen Trigger-Control-System, welches auch TDCs der Tagger-Latten steuert.

Zusätzlich werden über einen vertikalen VME Bus zwei weitere VME-Überrahmen angesteuert, in denen sich 33 Diskriminatoren des Typs SIS3500 der Firma Struck befinden. Der vertikale VME Bus ist realisiert mit VIC[13] Modulen des Types 8251 der Firma CES[14].

5.2.6. Forwardplug

Der Eventbuilder liest die digitalisierten Informationen der 90 Kristalle und der 180 Veto-Plättchen des Forwardplugs (siehe Kapitel 2.10). Es werden dabei nur die TDC-Werte aufgenommen, da für die Kristalle die ADC-Werte vom Crystal-Barrel-Eventbuilder mit digitalisiert werden und die Veto-Plättchen keine nützlichen ADC-Werte liefern. Dabei werden diese Zeitinformationen aus zwei CATCH-TDC Modulen für die Plättchen und einem weiteren für die Kristalle gelesen. Die Ansteuerung dieser Module erfolgt wie beim Innendetektor und den Tagger-Latten über das optische Trigger-Control-System. Gespeichert werden die TDC-Informationen von Kristallen und Plättchen in der Bank *FPT0*. Zur Kontrolle der Strahlposition werden die Raten von Treffern in ausgewählten Veto-Plättchen mit einem Zähler aufgenommen. Diese Informationen werden in der Bank *FPS0* gespeichert.

Zusätzlich erfolgt im Forwardplug-Eventbuilder noch die Steuerung des Lichtpulser Systems (Kapitel 2.16). Der Forwardplug-Eventbuilder öffnet dabei einen Netzwerkport mit dem sich der Trigger-Eventbuilder verbindet. Über vorgegebene Steuerbefehle kann der Trigger-Eventbuilder dann zum Beispiel Filter des Lichtpulsers fahren und die Blitzlampe einschalten. Die Befehle der Lichtpulserkontrolle sind in Anhang E.4.1 beschrieben. Die Daten des Lichtpulsers sind in der Bank *LPC0* gespeichert. Die Struktur ist in Anhang E.3 aufgeführt.

Eine weitere Aufgabe des Forwardplug-Eventbuilders ist die Steuerung des Clusterfinders der Forwardplug-Kristalle (Kapitel 3.5). Dabei werden sowohl die Schwellen der

[13] vertical interconnect - engl. für vertikale Zwischenverbindung. Möglichkeit zwei VME-Rahmen so miteinander zu verbinden, dass sich nur in einem der beiden Rahmen ein Rechner befindet, der auf beide Rahmen zugreifen kann.

[14] Creative Electronic Systems

Diskriminatoren gesetzt, als auch welche Informationen auf den beiden Trigger-Leitungen geliefert werden. Es können dabei sowohl unterschiedliche Zahlen an Clustern ausgewählt werden, als auch spezielle Muster gewählt werden (z.B. Ein Treffer oben und einer unten, für Kosmische Untergrundstrahlung).

5.2.7. Mini-TAPS

Die Elektronik zur Auslese der 216 Kristalle und Veto Detektoren des Mini-TAPS Detektors (Kapitel 2.12) ist sehr umfangreich und wird deshalb auf 6 VME-Rahmen aufgeteilt. Um die Auslesegeschwindigkeit zu optimieren wird jeder dieser 6 Rahmen von jeweils einem lokalen Eventbuilder angesprochen. Die Auslese des Mini-TAPS Detektors ist deshalb auf 6 Eventbuilder verteilt. Jeder VME-Rahmen enthält dabei 14 Module des Typs CAEN V874B welche jeweils 4 bzw. 8 Kanäle digitalisieren können. Es werden dabei sowohl die Zeitinformation als auch die Energieinformation gespeichert. Die 216 Veto Kanäle werden dabei von den ersten beiden Eventbuildern ausgelesen, wobei die Module für diese Auslese jeweils 8 Kanäle digitalisieren können. Die Auslese der BaF Kristalle befindet sich in den restlichen vier Rahmen, wobei die Module in diesen Rahmen nur jeweils 4 Kanäle digitalisieren können, da bei den Kristallen jeder Kanal mehrfach über unterschiedlich lange Fenster, bzw. Schwellen digitalisiert wird. Die TDC und ADC Informationen werden in jedem Eventbuilder zusammen in eine Bank (*M1C0 - M6C0*) geschrieben. Der Inhalt dieser Bänke ist genauer in Anhang E.3 beschrieben. Die Auslese dieses Subdetektors ist genauer in einer Diplomarbeit [Kla12] beschrieben.

5.2.8. Gamma Intensity Monitor

Der Eventbuilder für den Gamma Intensitätsmonitor liest sowohl seine eigenen Daten als auch die Daten des Fluss Monitor Detektors (Kapitel 2.15) aus. Für beide Detektoren werden dabei die TDC Informationen über ein V1290 VME-Modul der Firma CAEN gelesen. Die ADC Informationen werden über ein V965 Modul gelesen, welches ebenfalls von CAEN ist. Zusätzlich dazu wird noch eine mit einer FPGA speziell für den GIM Detektor entwickelte GIM-ADC-Control Karte gesteuert. Die TDC Informationen beider Detektoren wird in der Bank *GIT0* gespeichert, die ADC-Informationen in der Bank *GIA0*. Zusätzlich werden analog zum Crystal-Barrel-Eventbuilder am Anfang eines Datenblockes die Pedastalwerte in der ADCs in die Bank *GITB* gespeichert.

5.3. Der globaler Eventbuilder

Nachdem die lokalen Eventbuilder jeweils einzelne Teile des Ereignisses produziert haben, müssen diese zu einem Gesamtereignis zusammengesetzt werden. Der für diese Aufgabe zuständige Prozess ist der globale Eventbuilder. Wie die bisherige Datenakquisition [Sch04] basiert das Konzept des globalen Eventbuilders darauf, dass für jeden lokalen Eventbuilder ein Prozess die Daten empfängt und diese anschließend in einem weiteren Prozess zu einem Gesamtereignis zusammengebaut werden. Dieses Ereignis wird dann sowohl gespeichert als auch an einen während der Datennahme laufenden Prozess zur Überwachung der Daten weitergeleitet.

Das bisherige Konzept wurde jedoch in einigen grundlegenden Punkten geändert und neu implementiert. Zunächst einmal war es nötig die mögliche Anzahl der lokalen Eventbuilder zu erhöhen, da sich diese im neuen Aufbau erheblich vergrößert hatte. Analog zu den lokalen Eventbuildern wurde zusätzlich sowohl die Kommunikation als auch die Verwaltung der einzelnen unabhängigen Prozesse des globalen Eventbuilders statt mit der bisher benutzten ‘‘commoncpp’’-Erweiterung mit der ‘‘boost’’-Erweiterung implementiert. Dies sorgt für eine deutlich geringere Prozessorlast, was sowohl höhere Datenraten als auch mehr lokale Eventbuilder ermöglicht. Ein weiterer Punkt, der verbessert wurde, ist der Speichervorgang. Dieser Vorgang wurde angepasst um neben dem Speichern in das ZEBRA[15]- und ROOT[16]-Dateiformat das Speichern in das CBDF[17]-Format zu ermöglichen. Zusätzlich können die Daten nun mit unterschiedlichen Algorithmen komprimiert werden. Die einzelnen Formate und Kompressionsalgorithmen mit ihren Vor- und Nachteilen sind in Kapitel 5.3.3 beschrieben. In den folgenden Kapiteln wird die Funktionsweise des globalen Eventbuilders genauer beschrieben. Dieser empfängt dabei zunächst die Daten der lokalen Eventbuilder (siehe Kapitel 5.3.1) prüft sie auf Synchronität und Vollständigkeit und baut sie dann zu einem kompletten Ereignis zusammen (siehe Kapitel 5.3.2). Anschließend wird dieses Gesamtereignis über einen Puffer an einen weiteren Prozess (*EventProcessThread*) weitergegeben, der die Ereignisse mit dem ausgewählten Format und Komprimierung auf den Datenträger speichert. Der Puffer sorgt dabei dafür, dass kurzfristige Verzögerungen beim Speichern der Daten keine Auswirkung auf die Datennahme haben. Zusätzlich werden vom *EventProcessThread* selektierte Ereignisse über das Netzwerk an einen Onlinemonitor weitergegeben. Dadurch ist es möglich die Daten schon während der Datennahme zu begutachten und sicherzustellen, dass keine Defekte in den Subkomponenten vorhanden sind.

[15] Am CERN entwickeltes System zum Speichern von Daten - ZEBRA Handbuch
[16] Am CERN entwickelte, objektorientierte freie Software zur Analyse von Daten `https://www.cern.ch`
[17] Von Ch. Funke entwickeltes Format zum Speichern der Daten des Crystal-Barrel/TAPS-Experiments [Fun13].

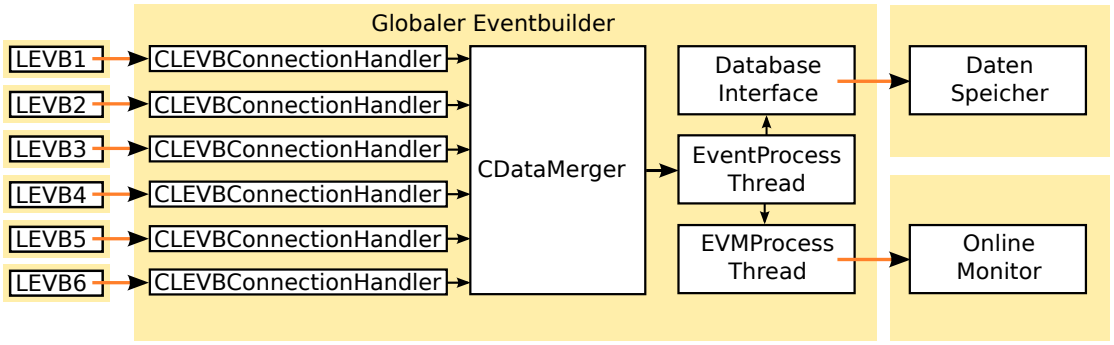


Abbildung 5.8.: **Schematische Darstellung des globalen Eventbuilder.** Die Daten der Lokalen Eventbuilder werden jeweils von einem LEVBConnection-Handler empfangen und an den CDataMerger weitergegeben. Dieser setzt die einzelnen Datenpackete zu einem Gesamtereignis zusammen und gibt dieses an den EventProcessThread weiter. Dieser leitet die Daten dann weiter an den Datenspeicher und den Onlinemonitor. Die in Orange dargestellten Verbindungen sind Netzwerkübertragungen.

5.3.1. Empfangen der Daten der LEVBs

Wenn die Datennahme für einen Datenblock gestartet wird, baut zunächst jeder lokale Eventbuilder eine Verbindung zum globalen Eventbuilder auf. Dort werden diese Verbindungen von der *LEVBConnectionHandler*-Klasse angenommen. Dabei wird für jeden lokalen Eventbuilder jeweils eine Instanz dieser Klasse angelegt. Der *LEVBConnection-Handler* ist dann während der Datennahme dafür zuständig die Daten von den lokalen Eventbuildern anzunehmen und in einen Puffer zu füllen. Aus Diesem holt sich dann der *DataMerger* (Kapitel 5.3.2) die empfangenen Daten zur Weiterverarbeitung. Falls der *DataMerger* kurzfristig zu lange zur Weiterverarbeitung braucht, ist durch diesen Puffer sichergestellt, dass weiterhin die Daten der lokalen Eventbuilder ohne Verzögerung angenommen werden können. Dadurch kann die Datennahme ungestört weiterlaufen. Analog zu den lokalen Eventbuilders besteht dieser Puffer aus mehreren *CRawDataBuffer* (siehe Kapitel 5.1.5). Dadurch können die in Form der *CRawDataBuffer* übertragenen Daten der lokalen Eventbuilder direkt gespeichert werden. Durch den Umbau des globalen Eventbuilders im Rahmen dieser Arbeit ist es jetzt auch möglich die Puffergröße einfach an die während einer Strahlzeit auftretenden Trigger- und Datenraten anzupassen (siehe Anhang E.2).

Eine weitere Aufgabe der *LEVBConnectionHandler*-Klasse ist es die Daten, die am Anfang des Datenblocks (vor der eigentlichen Datennahme) aufgezeichnet werden, zu speichern. Da für diese Daten kein Puffer erforderlich ist, ist im *LEVBConnectionHandler* ein Speicherbereich enthalten, in den die Daten gespeichert werden können. Zusätzlich

dazu überträgt jeder lokale Eventbuilder noch seine ID und die maximale Größe der Daten, die von diesem Eventbuilder bei jedem Ereignis übertragen werden. Durch die ID kann jeder *LEVBConnectionHandler* einem lokalen Eventbuilder zugeordnet werden, wodurch im Falle eines Problems beim Zusammenfügen der Ereignisse diese einem lokalen Eventbuilder zugeordnet werden können. Die maximale Größe dient dazu sicherzustellen, dass die übertragenen Daten auch gespeichert werden können. Dazu wird die Größe der Puffer im globalen Eventbuilder mit der maximal von den lokalen Eventbuildern übertragenen Größe verglichen.

Nachdem alle lokalen Eventbuilder ID, Größe und Startdaten übertragen haben, werden die Speicherbereiche der *LEVBConnectionHandler* vom *DataMerger* der Reihe nach ausgelesen und zu einem Gesamtereignis zusammengefügt. Dieses *TableEvent* wird dann von den nächsten Prozessen als erstes Ereignis gespeichert.

5.3.2. Zusammenbau der Ereignisse

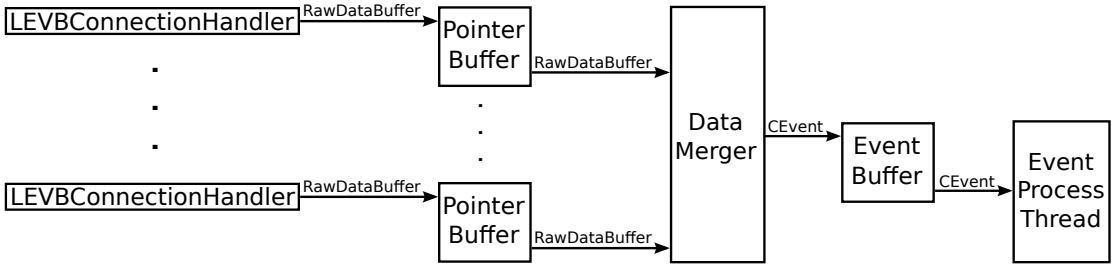


Abbildung 5.9.: **Schematische Darstellung des Datentransports des DataMerger.** Die vom LEVBConnectionHandler empfangenen Daten werden in Form eines RawDataBuffers in einem Zwischenspeicher geschrieben. Aus diesem holt sich der Datamerger der Reihe nach die einzelnen Datenpackete, setzt sie zusammen und gibt sie in Form eines CEvents an einen weiteren Puffer. Aus diesem holt sich anschließend der EventProcessThread das Gesamtereignis.

Die Hauptfunktion der Klasse *CDataMerger* ist es, die von den lokalen Eventbuildern angenommen Daten zu einem kompletten Ereignis zusammenzusetzen und dabei zu kontrollieren, ob es Probleme bei der Synchronisation gegeben hat. Dazu werden die Informationen, die das Synchronisationssystem aufgenommen hat (siehe Kapitel 4.1), ausgewertet. Diese Informationen werden von jedem lokalen Eventbuilder mitgeschickt und sind somit in jedem CRawDataBuffer enthalten. Um sicherzustellen, dass die Daten zum gleichen Ereignis gehören und keine Probleme mit dem Synchronisationssystem aufgetreten sind, werden der Reihe nach für jeden lokalen Eventbuilder die gespeicherten Daten aus dem Puffer des LEVBConnectionHandler abgefragt und die folgenden Werte

geprüft:

- Stimmt die Puffernummer des Ereignisses mit der des Trigger-Eventbuilders überein?
- Stimmt die lokal gezählte Puffernummer mit der vom Synchronisationssystem erhaltenen überein?
- Stimmt die per Software gezählte Anzahl der Ereignisse mit der vom Hardwarezähler überein?
- Stimmt die vom Hardwarezähler gezählte Anzahl der Ereignis mit der vom Trigger-Eventbuilder gezählten überein?
- Stimmt der Typ des Events mit dem Typ des Events des Trigger-Eventbuilders überein?

Falls alle diese Fragen positiv beantwortet werden können, werden die Daten dieses Eventbuilder an das Gesamtereignis angehängt. Wenn mindestens eine Frage nicht positiv beantwortet werden kann wird das Ereignis verworfen und eine Fehlermeldung ausgegeben.

Die Daten werden dabei in Form eines *CEvent* abgespeichert. In dieser Klasse werden die Daten jedes Subdetektors einzeln als *CRawDataBuffer* gespeichert. Die Anzahl der in jedem *CEvent* gespeicherten Puffer entspricht der Anzahl der Subdetektoren, da pro Subdetektor genau ein *CRawDataBuffer* gefüllt wird. Ein Zeiger auf dieses *CEvent* wird dann an einen Puffer (*CEventBuffer*) angehängt. Dieser Puffer kann weiter gefüllt werden, auch wenn der Prozess, der die Daten weiter verarbeitet, nicht schnell genug arbeitet. Dadurch wird dafür gesorgt dass sich kurzfristige Verzögerungen bei der Speicherung der Daten nicht auf den Empfang der Daten auswirken und damit die Datennahme nicht bremsen. Der gesamte Datentransport ist schematisch in Abbildung 5.9 gezeigt.

5.3.3. Speichern der Daten

Der Prozess die vom Experiment aufgenommenen Daten zu speichern beginnt damit, dass der EventProcessThread aus dem EventBuffer ein Ereignis entnimmt. Der Prozess erzeugt dann zunächst eine Kopie dieses in Form eines CEvents gespeicherten Ereignisses, welche an den Onlinemonitor weitergeleitet wird. Falls der Abtransport der Daten zum Onlinemonitor dabei zu langsam erfolgt, werden Ereignisse übersprungen bis wieder Daten an den Onlinemonitor übertragen werden können. Die wichtigen *ScalerEvent*-Ereignisse die nur in bestimmten Zeitabständen auftreten, werden dabei jedoch immer

gesendet. Für den eigentlichen Transport zum Onlinemonitor ist dabei die Klasse *EVMProcessThread* zuständig. Nachdem der *EventProcessThread* das Ereignis kopiert und weitergegeben hat, gibt er das Originalereignis zum Speichern weiter. Für die Speicherung der Daten stehen dabei mehrere Datenformate zur Verfügung. Zunächst ist dabei das ZEBRA-Dateiformat zu nennen. Dieses Format wurde vom CERN zur Speicherung von Experimentdaten entwickelt und ist auch das Format, dass für die bisher vom Crystal-Barrel/TAPS-Experiment aufgenommenen Daten gewählt wurde. Da das ZEBRA Format zu Zeiten von Bandlaufwerken entwickelt wurde und auf diese beim Design der Dateistrukturen Rücksicht genommen werden musste, werden in ZEBRA Dateien viele nicht benötigte Daten gespeichert, was die Dateigröße deutlich erhöht (siehe Tabelle 5.2). Als Alternative zum ZEBRA-Format stellt die DAQ noch das ROOT-Format zur Verfügung. Das ROOT-Dateiformat ist eine Dateistruktur, die von dem ebenfalls vom CERN entwickelten Programm ROOT benutzt wird um diverse Informationen, die während der Datenanalyse anfallen, zu speichern. Innerhalb dieses ROOT-Formats wurde für das Crystal-Barrel/TAPS-Experiment eine Struktur entwickelt, welche es erlaubt die anfallenden Daten der Subdetektoren in Form der gleichen Datenblöcken zu Speichern, welche auch im ZEBRA-Format benutzt werden. Das ROOT-Format hat dabei gegenüber dem ZEBRA-Format jedoch den Vorteil dass der Overhead deutlich kleiner ist (siehe Tabelle 5.2). Der Nachteil des ROOT-Formats ist jedoch, dass die gespeicherten Dateien nicht direkt mit einem einfachen Texteditor gelesen und kontrolliert werden können, sondern immer das ROOT-Programm benutzt werden muss um die Daten zu lesen. Als letzte Möglichkeit wurde noch ein eigens für das Crystal-Barrel/TAPS-Experiment entwickeltes Dateiformat mit dem Namen CBDF [Fun13] implementiert. Dieses Format wurde entwickelt um den Overhead weiter zu reduzieren. Es hat dadurch von allen drei Formaten die kleinsten Dateigrößen (siehe Tabelle 5.2). Zusätzlich ist es bei diesem Format wie beim ZEBRA-Format möglich die Dateien direkt mit einem Texteditor zu betrachten und zu kontrollieren. Neben den drei beschriebenen Dateiformaten können durch die flexible Struktur des globalen Eventbuilders sehr einfach weitere Formate implementiert werden.

Zusätzlich zu den unterschiedlichen Dateiformaten kann optional eine Komprimierung der Daten gewählt werden um die Datengröße weiter zu reduzieren. Dies ist insbesonders im Fall des ZEBRA Dateiformats notwendig um die Datennahme nicht zu beeinträchtigen. Dies liegt daran, dass durch die erhebliche Vergrößerung der Dateien die aus dem globalen Eventbuilder auslaufende Datenrate deutlich höher als die einlaufende Rate wird. Wenn die Daten mit der maximale Datenrate über das Gigabit-Netzwerk einlaufen, ist es somit nicht mehr möglich diese über das Gigabit Netzwerk abzuführen. Dadurch ist bei maximaler Datenrate eine Komprimierung zwingend erforderlich um die Datennahme

Dateiformat	Größe[MByte]
Rohdaten	2813
ZEBRA	4011
ROOT	3149
CBDF	2984

Tabelle 5.2.: **Datengrößen der möglichen Dateiformate.** Aufgeführt sind die Dateigrößen der verschiedenen Dateiformate einer unkomprimierten Datei aus einer Strahlzeit des Crystal-Barrel/TAPS-Experiments mit einem Niob-Target.

nicht zu beeinträchtigen.

Zur Kompression stehen dabei unterschiedliche Algorithmen zur Verfügung, die zusätzlich noch mit unterschiedlichen Kompressionsgraden ausgeführt werden können. Die dabei entstehenden Dateigrößen eines typischen Datenblocks aus einer Strahlzeit mit Niob-Target sind in Tabelle 5.3 aufgeführt. Das Problem von Algorithmen mit höherer Kompression sowie höheren Kompressionsgraden ist dabei dass die Berechnung der komprimierten Daten deutlich aufwendiger wird. Über das benutzte Gigabit-Netzwerk können die 2813 MByte Rohdaten in 25s übertragen werden. Damit die Komprimierung keinen Einfluss auf die Geschwindigkeit der Datennahme hat, muss die gesamte Datei demnach auch in unter 25s komprimiert werden. Tabelle 5.3 zeigt, dass nur nur der lzo Algorithmus bis zum Grad 6 dazu in der Lage ist die maximale Datenrate des Netzwerks zu komprimieren. Während der Niob-Strahlzeit bei der die Testdatei aufgenommen wurde betrug die minimale Zeit zum Schreiben der Dateien 1440 s. Dies würden alle Algorithmen schaffen. Nur beim lzma Algorithmus wären die höchsten Kompressionsgrade ab Grad 6 nicht mehr möglich.

Um höhere Kompressionsgrade oder aufwendigere Algorithmen zu benutzen müsste eine parallele Kompression implementiert werden. Da die Datenraten im Experiment jedoch nicht an die Maximalrate herankommen und die Daten bei der Offline-Analyse noch einmal erneut in einem anderen Format komprimiert werden können, war dies bisher nicht notwendig.

Algorithmus (Kompression-Grad)	Größe (1)	Zeit [s] (1)	Größe (3)	Zeit [s] (3)	Größe (6)	Zeit [s] (6)	Größe (9)	Zeit [s] (9)
lzo	45,6 %	22	46,1 %	23	46,1 %	23	35,2 %	1045
zip	37,2 %	66	34,9 %	88	31,8 %	148	31,6 %	594
bzip2	30,4 %	314	28,3 %	312	27,4 %	329	27,0 %	339
lzma	23,5 %	224	20,2 %	1091	18,9 %	1790	18,3 %	2259

Tabelle 5.3.: **Datengrößen der verschiedenen Kompressionsalgorithmen.** Es wurde eine Datei im ZEBRA-Format von 4011 MByte Größe mit verschiedenen Algorithmen komprimiert. Aufgetragen sind jeweils die Zeiten die zur Kompression nötig sind und die Größen der entstehenden Dateien relativ zur Originaldatei.

5.4. Kontrolle der Datenerfassung

Wie in den letzten Kapitel beschrieben besteht die Datenerfassung aus vielen voneinander unabhängigen Komponenten. Diese Komponenten müssen von einer zentralen Stelle gesteuert werden. Dafür wurde im Rahmen dieser Arbeit ein neues Konzept entwickelt. Es wurde dafür ein zentral laufender Dienst (DAQ Daemon) entwickelt. Dieser Dienst verbindet sich mit allen lokalen Eventbuildern und dem globalen Eventbuilder. Der Benutzer kann sich dann entweder mit einem Kommandozeilenprogramm (Runcontrol) oder über eine graphischen Oberfläche (DAQt) mit dem Daemon verbinden. Beide Programme können dann von dem DAQ Daemon den aktuellen Status abfragen. Außerdem können Änderungen an der Konfiguration der Eventbuilder übertragen werden, sowie die Datennahme gestartet oder gestoppt werden. Der zentrale DAQ Daemon stellt dabei sicher, dass sich beliebig viele Kontrollprogramme gleichzeitig gestartet sein können ohne dass es dadurch zu Problemen kommt. Dadurch kann die DAQ jederzeit von beliebigen Rechnern überwacht werden und wenn benötigt direkt vor Ort an den Subkomponenten gestartet werden. Der DAQ Daemon wird in Kapitel 5.4.1 näher beschrieben. Die Kontrollprogramme werden in Kapitel 5.4.2 (Runcontrol) und Kapitel 5.4.3 (DAQt) genauer beschrieben.

5.4.1. DAQ Daemon

Der DAQ Daemon ist der zentrale Dienst zur Kontrolle und Steuerung der Datenerfassung. Er benutzt zur Kommunikation die gleichen auf Boost aufbauenden TCP Klassen wie die lokalen Eventbuilder (CControlSessionBoost und CTCPBoostConnection). Er besteht dabei aus zwei Teilen (DAQControlDaemon und CDAQStatusThread). Der CDAQStatusThread überwacht dabei ständig den Status der einzelnen DAQ Komponenten, wie

zum Beispiel ob die einzelnen lokalen Eventbuilder noch aktiv sind und was die aktuelle Eventnummer ist. Der DAQControlDaemon beinhaltet die TCP-Kommunikation, stellt darüber die Informationen des StatusThreads zur Verfügung, und ermöglicht weitere Funktionen, wie zum Beispiel das Starten und Stoppen von Runs.

5.4.2. Runcontrol

Die Runcontrol ist ein Kommandozeilenprogramm zur Steuerung und Überwachung der Datenerfassung. Es verbindet sich dazu mit dem DAQ Daemon. Über die Runcontrol ist es möglich die Datenakquisition direkt am Detektor zu starten und zu stoppen. Man kann außerdem wählen, welche Detektoren ausgelesen werden sollen, welcher Trigger benutzt werden soll, sowie einen Namen für die Datei wählen, in die die Daten geschrieben werden. Dabei werden eine laufende run Nummer und der gewählte Trigger mit in den Dateinamen geschrieben. Für die effiziente Datennahme von mehreren Dateien am Stück gibt es einen Autopilot Modus, der nach einer vorgewählten Anzahl von Events eine neue Datei schreibt. Die runcontrol zeigt den Status der einzelnen Subdetektoren an, und gibt, wenn die Datennahme läuft die aktuelle Eventnummer an. Wenn Daten von nur einem Subdetektor aufgenommen werden sollen, kann sich die Runcontrol auch mit der Standalone DAQ (Kapitel 5.5) verbinden. In Abbildung 5.10 ist das Menü der Runcontrol gezeigt.

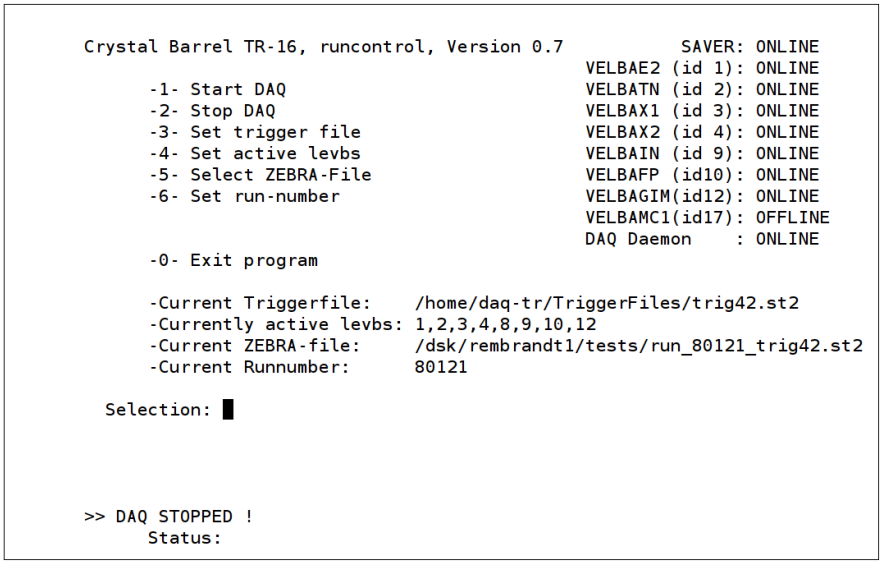


Abbildung 5.10.: **Die Runcontrol Oberfläche.** Am rechten Rand ist der Status der einzelnen Subdetektoren gezeigt. In der Mitte oben sind die möglichen Auswahlpunkte zu sehen. Unten ist der Status und die einzelnen Konfigurationen sichtbar.

5.4.3. Graphische Kontrolloberfläche DAQt

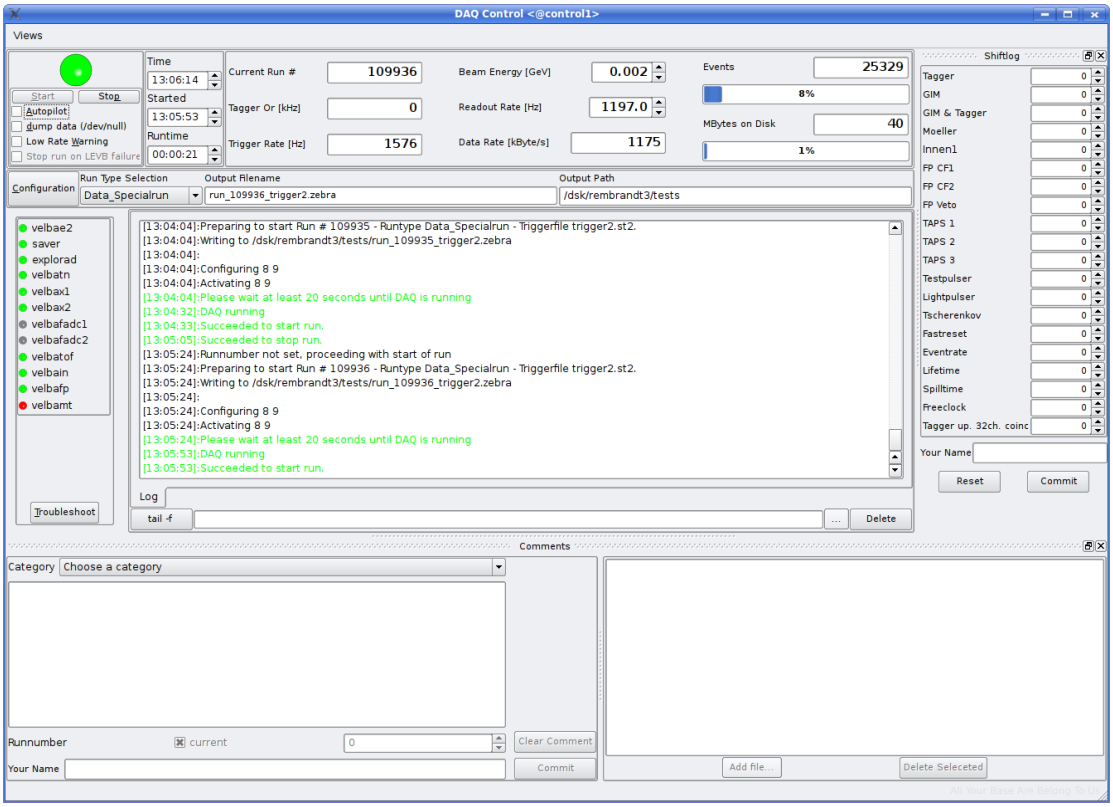


Abbildung 5.11.: **Die DAQt Oberfläche.**

Um die Kontrolle während einer Strahlzeit zu vereinfachen wurde in Zusammenarbeit mit Damian Piontek eine grafische Oberfläche entwickelt. Dabei wurde die Qt[18]-Entwicklungsumgebung benutzt, woraus sich auch der Name DAQt ableitet. Mit diesem Programm können zunächst analog zur Runcontrol die grundlegenden Kontrollfunktion der DAQ durchgeführt werden. So kann die Datenerfassung gestartet, gestoppt und überwacht werden. Dazu verbindet sich auch die DAQt mit dem DAQ Daemon (Kapitel 5.4.1). Man kann ebenfalls die Konfiguration der lokalen Eventbuilder (Kapitel 5.1), des Triggers (Kapitel 3) und des globalen Eventbuilders (Kapitel 5.3) ändern.

Zusätzlich zu dieser Grundfunktionalität bietet die DAQt jedoch entscheidende weitere Funktionalität welche im folgenden aufgeführt werden.

In der DAQt können zunächst die oben genannten Konfigurationen in Profilen gespeichert werden. Dies ermöglicht es Profile für unterschiedliche Anforderungen anzulegen

[18] Entwicklungsumgebung zur plattformübergreifenden Entwicklung grafischer Benutzeroberflächen und Programme. www.qt.io/

und während der Datennahme nur diese Profile zu wechseln. Dies beschleunigt zunächst einmal das Wechseln zwischen unterschiedlichen Messprofilen enorm. Zusätzlich ist dieses Verfahren auch weniger fehleranfällig. Ein weiteres Feature ist der ‘‘Autopilot’’. Dieser ermöglicht es am Ende eines Datenblocks den nächsten automatisch zu starten. Dabei kann vorher auch das Konfigurationsprofil für den nächsten Block gewechselt werden. Durch die Automatisierung die dieses Feature mit sich bringt verringert sich die Zeit zwischen zwei Datenblöcken, in der keine Daten gespeichert werden können.

Die Kontrolle der Eventbuilder wird ebenfalls durch die DAQt deutlich vereinfacht. So werden neben dem Status der Eventbuilder in Grün und Rot, auch alle Fehlermeldungen, die von den Eventbuildern ausgegeben werden übersichtlich in der Mitte der DAQt dargestellt. Im Falle eines Fehlers können die Eventbuilder auch in der DAQt über das ‘‘Troubleshoot’’-Interface durch einen Klick neugestartet werden.

Um eine Strahlzeit besser zu dokumentieren bietet die DAQt einige Werkzeuge. Zunächst einmal werden bei jedem Start und Stop der Datenakquisition eine Reihe relevanter Daten in die ‘‘Rundatabase’’ (siehe Kapitel 5.4.4) geschrieben. Dies sind unter anderem die jeweiligen Zeitpunkte, die Konfigurationsdateien und die Anzahl der gespeicherten Ereignisse. Die Personen die während der Datennahme die Datenerfassung kontrollieren können zusätzlich auch manuell Kommentare zur Datenbank hinzufügen, diese werden dann von der Datenbank je nach gewähltem Kategorie an eine festgelegte E-Mail Adresse geschickt. So können die zuständigen Experten zeitnah über Probleme informiert werden. Die letzte Möglichkeit zur Dokumentation ist das Speichern der aktuellen Raten im Experiment. Diese Raten werden im Kontrollraum auf einem Monitor angezeigt und können mit der DAQt in die ‘‘Rundatabase’’ gespeichert werden.

In Strahlzeiten mit linear polarisierten Photonen wird ein Diamant zur Erzeugung dieser Photonen benutzt. Dieser Diamant muss mit einem Goniometer (Kapitel 2.2) auf vorbestimmte Positionen bewegt werden. Die Steuerung des Diamanten wird von der DAQt ebenfalls automatisiert. So kann nach einer festgelegten Anzahl von Datenblöcken automatisch zwischen zwei Positionen gewechselt werden. Es kann jedoch auch eine von beide Positionen manuell für den nächsten Datenblock gewählt werden.

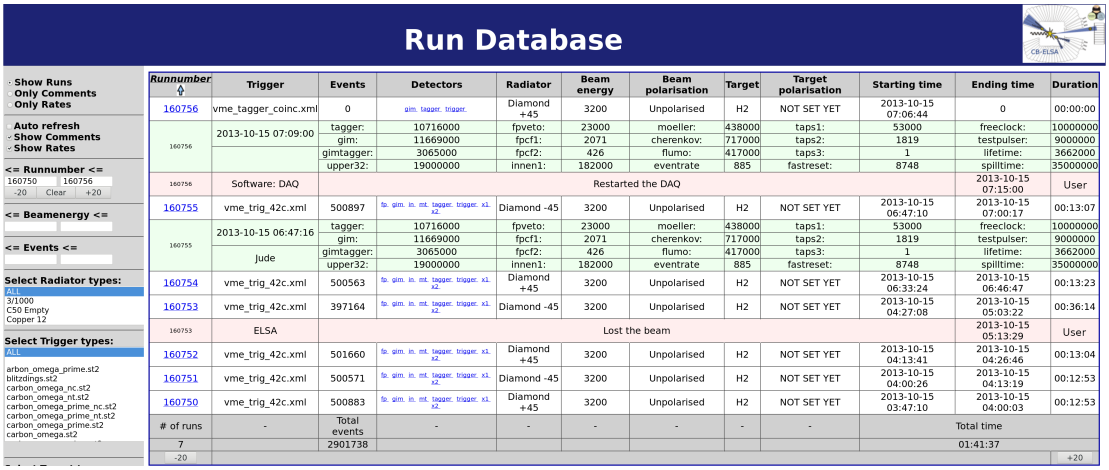
Die DAQt ist dabei auf einen parallelen Betrieb ausgelegt. Das heißt sie kann gleichzeitig auf mehreren Rechnern gestartet werden. Dadurch kann der Status der Datenerfassung einfach an unterschiedlichen Stellen parallel überwacht werden und die Datenerfassung kann auch an unterschiedlichen Stellen kontrolliert werden. Um sicherzustellen das die Konfiguration aller gestarteten Instanzen der DAQt identisch ist, wird diese in einer Postgresql[19]-Datenbank gespeichert. Diese Datenbank wird dann von allen DAQt-Instanzen gelesen.

[19] freies objektrelationales Datenbankmanagementsystem. www.postgresql.org/

5.4.4. Rundatabase

Die Informationen über alle gespeicherten Datenblöcke werden von der DAQt in einer Datenbank gespeichert. Diese Rundatabase ist in einer Postgresql-Datenbank implementiert. Darin werden beim Start und Stop der Datenblöcke eine Reihe von Informationen eingetragen. Zusätzlich werden manuell in die DAQt eingegebene Kommentare gespeichert und per E-Mail weitergeleitet. Als letztes können noch eine Reihe von Raten in die Datenbank eingetragen werden.

Für den Benutzer werden die gespeicherten Einträge über ein Web-Interface dargestellt. Dieses ist in PHP[20] realisiert. In Abbildung 5.12 ist ein Ausschnitt dieser Seite gezeigt.



Run Database

· Show Runs
Only Comments
Only Rates

Auto refresh
· Show Comments
· Show Rates

<= Runnumber <=
160750 160756
-20 Clear +20

<= Beamenergy <=

<= Events <=

Select Radiator types:
ALL
3/1000
C50 Empty
Copper 12

Select Trigger types:
ALL
arbon_omega_prime.st2
blitzdings.st2
carbon_omega_nc.st2
carbon_omega_nt.st2
carbon_omega_prime_nc.st2
carbon_omega_prime_nt.st2
carbon_omega_prime.st2
carbon_omega.st2

Runnumber	Trigger	Events	Detectors	Radiator	Beam energy	Beam polarisation	Target	Target polarisation	Starting time	Ending time	Duration
160756	vme_tagger_coinc.xml	0	gim, tagger, trigger,	Diamond +45	3200	Unpolarised	H2	NOT SET YET	2013-10-15 07:06:44	0	00:00:00
160756	2013-10-15 07:09:00	tagger:	10716000	fpveto:	23000	moeller:	438000	taps1:	53000	freeclock:	10000000
		gim:	11669000	fpcf1:	2071	cherenkov:	717000	taps2:	1819	testpulser:	9000000
		gimtagger:	3065000	fpcf2:	426	flumo:	417000	taps3:	1	lifetime:	3662000
		upper32:	19000000	innen1:	182000	eventrate	885	fastreset:	8748	spilltime:	35000000
160756	Software: DAQ	Restarted the DAQ								2013-10-15 07:15:00	User
160755	vme_trig_42c.xml	500897	fp, gim, in, mt, tagger, trigger, x1, x2,	Diamond -45	3200	Unpolarised	H2	NOT SET YET	2013-10-15 06:47:10	2013-10-15 07:00:17	00:13:07
160755	2013-10-15 06:47:16	tagger:	10716000	fpveto:	23000	moeller:	438000	taps1:	53000	freeclock:	10000000
		gim:	11669000	fpcf1:	2071	cherenkov:	717000	taps2:	1819	testpulser:	9000000
	Jude	gimtagger:	3065000	fpcf2:	426	flumo:	417000	taps3:	1	lifetime:	3662000
		upper32:	19000000	innen1:	182000	eventrate	885	fastreset:	8748	spilltime:	35000000
160754	vme_trig_42c.xml	500563	fp, gim, in, mt, tagger, trigger, x1, x2,	Diamond +45	3200	Unpolarised	H2	NOT SET YET	2013-10-15 06:33:24	2013-10-15 06:46:47	00:13:23
160753	vme_trig_42c.xml	397164	fp, gim, in, mt, tagger, trigger, x1, x2,	Diamond -45	3200	Unpolarised	H2	NOT SET YET	2013-10-15 04:27:08	2013-10-15 05:03:22	00:36:14
160753	ELSA	Lost the beam								2013-10-15 05:13:29	User
160752	vme_trig_42c.xml	501660	fp, gim, in, mt, tagger, trigger, x1, x2,	Diamond +45	3200	Unpolarised	H2	NOT SET YET	2013-10-15 04:13:41	2013-10-15 04:26:46	00:13:04
160751	vme_trig_42c.xml	500571	fp, gim, in, mt, tagger, trigger, x1, x2,	Diamond -45	3200	Unpolarised	H2	NOT SET YET	2013-10-15 04:00:26	2013-10-15 04:13:19	00:12:53
160750	vme_trig_42c.xml	500883	fp, gim, in, mt, tagger, trigger, x1, x2,	Diamond +45	3200	Unpolarised	H2	NOT SET YET	2013-10-15 03:47:10	2013-10-15 04:00:03	00:12:53
# of runs	-	Total events	-	-	-	-	-	-	Total time		
7		2901738							01:41:37		
-20											+20

Abbildung 5.12.: **Das Web-Interface der Rundatabase.** Hier werden die Einträge der Rundatabase zu den einzelnen Datenblöcken gezeigt. Zusätzlich werden die manuell eingegebenen Kommentare in Rot und die gespeicherten Zählerstände in Grün angezeigt. Durch Anklicken der Nummer des Datenblocks werden weitere Details zum Datenblock gezeigt. Hinter den Detektor-Namen sind die jeweiligen Konfigurationsdateien aufgeführt. Auf der linken Seite kann zusätzlich noch mit Filtern festgelegt werden, welche Datenblöcke angezeigt werden.

[20] rekursives Akronym für PHP: Hypertext Preprocessor - Freie Skript-Sprache zum Erstellen von dynamischen Webseiten.

5.5. Standalone DAQ

Um die Funktionalität der einzelnen Subdetektoren auch unabhängig testen zu können wurde ein eigenständiges Datenakquisitionssystem entwickelt. Dieses verzichtet auf die Synchronisationskomponente der Haupt-DAQ, da bei einem einzelnen Subdetektor die Daten nicht mehr zusammengesetzt werden müssen und damit automatisch synchron sind. Dafür wird jedoch die Komponente die Daten im ZEBRA Format auf den Festplattenspeicher schreibt in den lokalen Eventbuilder integriert. Zusätzlich wird das Synchronisationsmodul mit einer neuen Firmware geladen und fungiert dann als Triggermodul. Dadurch ist der Einsatz die separate Auslese des Subdetektors ohne zusätzliche Hardwarekomponenten möglich. Zusätzlich ermöglicht die Standalone-DAQ es auch in Laboraufbauten unabhängig eine simple Datenakquisition aufzubauen die, falls dies nachher erwünscht ist, auch einfach in das Gesamtexperiment integriert werden kann. Zum Betrieb der Standalone DAQ wurde eine modifizierte Klassensammlung *baselevb_standalone* entwickelt, die aus den Komponenten *CReadoutThread* und *CProcessThread* des Gesamtsystems besteht, hier *CReadoutThreadStandlone* und *CProcessThreadStandalone* genannt. Zusätzlich wurde die Klasse *CDataThread* modifiziert, so dass die Daten nicht mehr über das Netzwerk weitergeleitet werden sondern direkt auf die Festplatte geschrieben werden. Diese modifizierte Klasse nennt sich *CDataThreadStandalone*. Kontrolliert wird das System von der *Runcontrol_SA*, welche auf der Klasse *DAQControlCenter_SA* aufbaut.

6. Messungen zur Leistungsfähigkeit der Datenakquisition

Die Leistungsfähigkeit eines Datenakquisitionssystems wird bestimmt durch die sogenannte Totzeit. Die Totzeit einer Datenakquisition ist die Zeit in der sie auf auftretende Ereignisse nicht reagiert. Dies ist typischerweise die Zeit zwischen dem Moment, in dem ein Ereignis akzeptiert wird, und dem Moment, in dem der Trigger wieder bereit ist ein neues Ereignis zu akzeptieren. Während dieser Zeit ist das Experiment nicht sensitiv auf neue Ereignisse, daher sollte diese Zeit minimiert werden. Die Totzeit setzt sich aus mehreren Komponenten zusammen. Der Hauptteil der Totzeit fällt an, durch die Auslese der digitalen Informationen aus den Modulen der einzelnen Subdetektoren. Dies geschieht für die verschiedenen Komponenten parallel, so dass der gesamte Anteil der Totzeit, welcher durch die Auslese entsteht, einzig aus der Auslesezeit des langsamsten Detektors bestimmt wird. Dieser Anteil wird im Kapitel 6.2 näher aufgeführt. Die zusätzlich noch entstehenden Totzeiten, welche zum Beispiel aus der Steuerung des Triggers resultieren, sind in Kapitel 6.1 aufgeführt. Ein weitere Punkt, der die Totzeit beeinflussen kann, ist die Zeit die benötigt wird die entstehenden Daten über das Netzwerk zu transportieren und letztlich auf einen Datenspeicher zu speichern. Dieser Vorgang erfolgt parallel zur restlichen Auslese und erzeugt somit keine zusätzliche Totzeit, wenn die Daten schneller transferiert und gespeichert werden können, als sie produziert werden. Dieser Vorgang wird in Kapitel 6.3 betrachtet. In Kapitel 6.4 wird eine Methode untersucht um die Leistungsfähigkeit der Datenakquisition weiter zu verbessern, indem die Zeit ausgenutzt wird in der der Teilchenbeschleuniger keinen Strom an das Experiment extrahiert. Die Eignung der hier vorgestellten Datenakquisition für das Crystal-Barrel/TAPS-Experiment wird in Kapitel 6.5 untersucht.

6.1. Totzeit des Datenakquisitionssystems

Für den einfacheren Fall eines einstufigen Triggers (Kapitel 3) teilt sich die Zeit während der Datennahme in drei Bereiche auf (siehe Abbildung 6.1), diese sind die Vorbereitungszeit, die Lifetime und die Auslesezeit. Die Vorbereitungszeit ist die Zeit vor dem Öffnen des Triggerfensters, während der alle elektronischen Module für die Datenaufnahme

Abbildung 6.1.: **Schematische Darstellung der Abläufe bei einem einstufigen Trigger.** Bei einem einstufigen Trigger kommt zunächst die Vorbereitungszeit, gefolgt von der Lifetime und der Auslesezeit. Anschließend beginnt der Vorgang wieder mit der Vorbereitungszeit. Die Vorbereitungszeit und die Auslesezeit bilden zusammen die Totzeit.

vorbereitet werden. Sie muss groß genug gewählt werden um auch dem langsamsten Modul die Möglichkeit zu geben, (wieder) bereit für die Aufnahme eines weiteren Ereignisses zu sein. Um dies für die ADC-Module des Crystal-Barrel-Detektors sicherzustellen wurde eine Vorbereitungszeit von 5 µs gewählt. Die Lifetime ist die Zeit während der das Triggersystem auf ein spezifiziertes Ereignis wartet. Sie entspricht der Zeitspanne bis ein Ereignis die Triggerbedingung erfüllt und ist damit abhängig von der gewählten Triggerbedingung und kann von wenigen Nanosekunden bis zu beliebig langen Zeiten dauern. Die Auslesezeit ist die Zeit, die benötigt wird um die Informationen aus den Elektronikmodulen auszulesen und wird später in diesem Kapitel genauer betrachtet. Als Totzeit bezeichnet man den Teil, während der das Experiment nicht sensitiv auf neue Ereignisse ist. Die Totzeit ist somit die Summe aus Vorbereitungszeit und Auslesezeit.

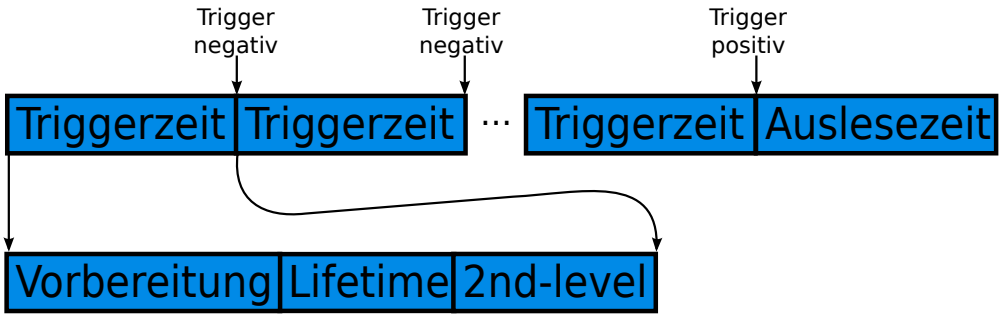


Abbildung 6.2.: **Schematische Darstellung der Abläufe bei einem zweistufigen Trigger.** Bei einem zweistufigen Trigger kommt zunächst ebenfalls die Vorbereitungszeit gefolgt von der Lifetime. Danach folgt die Zeit die für die Entscheidung der zweiten Stufe nötig ist. Falls die zweite Stufe keine positive Entscheidung trifft folgen anschließend wieder Vorbereitungszeit, Lifetime und die Entscheidungszeit der zweiten Stufe. Falls eine positive Entscheidung von der zweiten Stufe erfolgt, beginnt die Auslesezeit und der gesamte Vorgang beginnt von vorne.

Im Falle eines zweistufigen Triggers sieht der Ablauf anders aus (siehe Abbildung 6.2). Nachdem die erste Stufe des Triggers am Ende der Lifetime eine positive Entscheidung getroffen hat, wird die zweite Stufe aktiv und prüft ihre Bedingung. Im Crystal-

Barrel/TAPS-Experiment dauert diese Phase (2nd-level-Entscheidung) im Mittel 6 µs und maximal 10 µs (Siehe Kapitel 3.4). Nach 10 µs wird die Phase mit einer negativen Entscheidung abgebrochen, da dann nur noch selten eine positive Entscheidung erreicht werden kann. Wenn die zweite Stufe kein passendes Ereignis gefunden hat, folgt anschließend erneut die Vorbereitungszeit, bevor die erste Stufe wieder aktiv wird. Falls die zweite Stufe ein passendes Ereignis gefunden hat, folgt anschließend die Auslesezeit. Die Totzeit setzt sich hier also aus den 2nd-level Entscheidungszeiten, den Vorbereitungszeiten sowie der Auslesezeit zusammen.

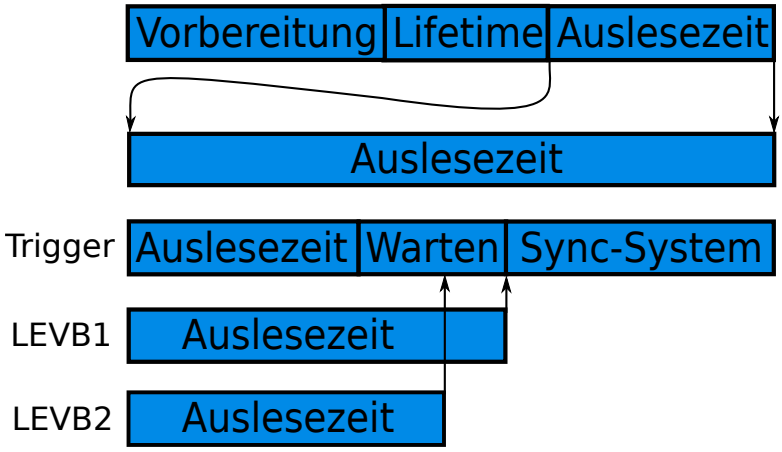


Abbildung 6.3.: **Schematische Darstellung der zeitlichen Abläufe im Trigger-LEVB.** Die Auslesezeit teilt sich beim Trigger LEVB in drei Teile auf. Die eigentliche Auslesezeit der Module des Triggers, eine Wartezeit bis alle anderen LEVBs mit ihrer Auslesezeit fertig sind, sowie einer Zeit zum Steuern von Sync-System und Trigger.

Die Auslesezeit lässt sich dabei noch genauer betrachten (Abbildung 6.3). Da der Trigger-Eventbuilder (Kapitel 5.1) die Auslese kontrolliert, ist er auch für die Dauer verantwortlich. Dabei entsteht zuerst die eigene Auslesezeit der Module des Triggers, danach wartet der Eventbuilder auf die Auslese der anderen Detektoren. Die dafür benötigte Zeit wird dabei, durch die parallele Auslese, nur vom langsamsten Detektor bestimmt. Anschließend muss das Synchronisationssystem und der Trigger wieder auf das nächste Ereignis vorbereitet werden. Die Totzeit, die durch die Auslesezeit der Detektoren entsteht (auch die des Trigger Eventbuilders) beträgt dabei minimal etwa 14 µs. Diese Zeit entsteht durch die Auslese von Werten und der Steuerung des Synchronisationssystems über den VME-Bus. Durch die Vorbereitungszeit entsteht nochmal eine Totzeit von 5 µs. Die Zeit die für die zusätzliche Steuerung des Synchronisationssystems im Trigger-LEVB nötig ist, sowie die Zeit zum Steuern des Triggers beträgt zusammen minimal

etwa 6,5 µs. Die entstehende Totzeit hängt dabei vom Typ des verwendeten VME-CPU Modulen ab, da die Zeit, die für einen VME-Zugriff benötigt wird, bei Verwendung eines anderen Moduls teilweise deutlich höher sein kann. Der Grund dafür sind große Unterschiede in der Art der Anbindung des VME-Controllers an die CPU. Rechnerisch ergibt sich eine minimale Totzeit von $5\,\mu s + 14\,\mu s + 6{,}5\,\mu s = 25{,}5\,\mu s$. Bei Verwendung einer Triggerentscheidung, die keine signifikante Lifetime erzeugt, ergibt sich damit eine maximal mögliche Ereignisrate von etwa 39 kHz. Da es sich bei dem verwendeten Linux Betriebssystem um kein Realtime[1] System handelt, ergibt sich im Mittel eine etwas höhere Totzeit. Die gemessene Ereignisrate von 33 kHz und damit eine Totzeit von 30,3 µs ist um etwa 5 µs höher. Diese zusätzliche Totzeit verteilt sich auf die schon erwähnten Verzögerungen durch das Betriebssystem sowie auf Verzögerungen durch Speicherzugriffe im Programmablauf. Da diese beiden Effekte von der Last des Systems und dem Typ des VME-Moduls abhängen, werden für die Abschätzung der Maximalraten im nächsten Abschnitt die idealen Werte als minimal mögliche Totzeit angenommen.

6.2. Totzeit durch die Auslese der einzelnen Subdetektoren

Zusätzlich zu der im vorhergehenden Abschnitt bestimmten minimalen systembedingten Totzeit ergibt sich noch eine weitere Totzeit durch die Auslese der einzelnen VME-Module der Subdetektoren. Hinzu kommt noch eine Totzeit durch in festen Zeitabständen stattfindende Auslese von weiteren Zählermodulen. Da diese, mit den aktuellen Einstellungen, jedoch nur all 50 ms ausgelesen werden, ist die Auswirkung auf die mittlere Totzeit zu vernachlässigen. Um die Dauer der Totzeit durch die Auslese zu bestimmen, wird während der Datennahme bei jedem Ereignis für jeden Detektor die Dauer der Auslese-Funktion gespeichert. Da die Dauer der Auslesezeit von der Anzahl der angesprochenen Detektoren in jeder Subkomponente abhängt, ist die Auslesezeit von der verwendeten Triggerentscheidung (Kap. 3.1) sowie auch von dem verwendeten Target (Kap. 2.6 und 2.7) und dem extrahierten Strahlstrom abhängig. Das Target spielt dabei vor allem eine Rolle, da sich durch unterschiedliche Konfigurationen die Untergrundsituation ändert und damit auch die Anzahl der zusätzlich zum guten Ereignis auftretenden Einträge. In den Abbildungen 6.4 und 6.5 sind die durch die Auslese der Detektoren zusätzlich entstehende Totzeiten für zwei repräsentative Konfigurationen aufgeführt.

In Abbildung 6.4 sind die Werte für den Datenblock 158291 aus der ‘‘August 2013’’-Strahlzeit gezeigt. Dabei wurde ein Flüssigwasserstoff-Target und ein linear polarisierter Photonstrahl verwendet. Die Triggerbedingung hat mindestens 3 nachgewiesene Teilchen

[1] Betriebssystem bei dem alle Operationen von Anwendungsprogrammen und das Verarbeiten der Signale von Hardware-Schnittstellen eine maximale vorbestimmte Laufzeit haben.

gefordert. Der extrahierte Elektronenstrom war etwa 640 nA.

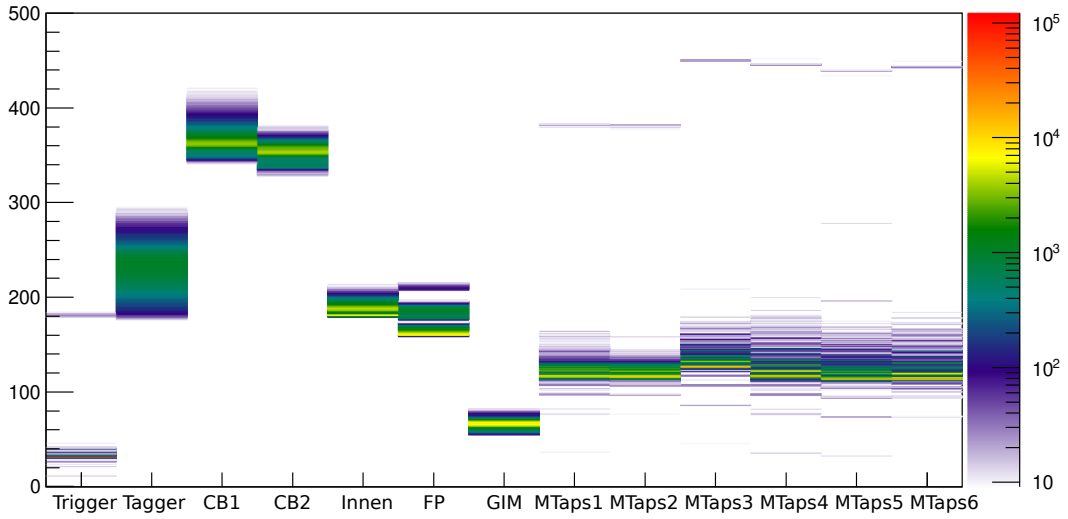


Abbildung 6.4.: **Die Totzeit der einzelnen Subdetektoren.** In dieser Abbildung wurde ein Trigger verwendet der mindestens drei nachgewiesene Teilchen verlangt. Als Target wurde Flüssigwasserstoff verwendet.

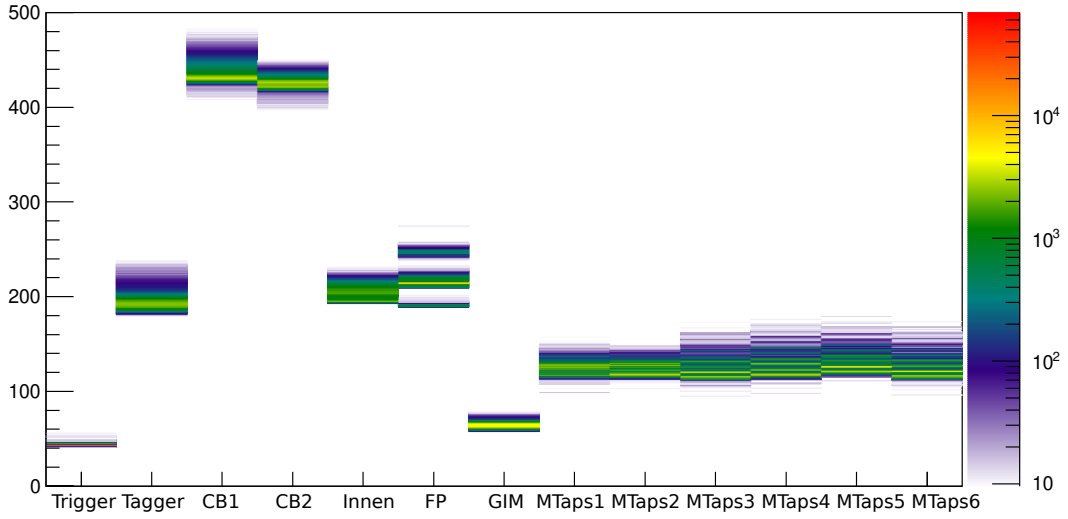


Abbildung 6.5.: **Die Totzeit der einzelnen Subdetektoren.** In dieser Abbildung wurde ein Trigger verwendet der mindestens drei nachgewiesene Teilchen verlangt. Als Target wurde Niob verwendet.

In Abbildung 6.5 sind die Werte für den Datenblock 163211 aus der "November 2013"-Strahlzeit gezeigt. Dabei wurde ein Niob-Target und ein nicht polarisierter Photonstrahl verwendet. Die Triggerbedingung hat mindestens 3 nachgewiesene Teilchen gefordert. Der extrahierte Elektronenstrom war etwa 120 nA.

Die mittleren Totzeiten für beide Datenblöcke sind in der folgenden Tabelle aufgeführt:

Detektor	Totzeit LH	Totzeit Niob
Trigger	32,1 µs	43,7 µs
Tagger	230,1 µs	193,7 µs
CB1	366,1 µs	432,9 µs
CB2	354,7 µs	426,1 µs
Innen	188,1 µs	205,1 µs
FP	169,3 µs	216,1 µs
GIM	67,0 µs	64,9 µs
Mtaps1	121,2 µs	124,8 µs
Mtaps2	120,5 µs	124,5 µs
Mtaps3	130,4 µs	123,0 µs
Mtaps4	122,2 µs	126,3 µs
Mtaps5	118,3 µs	129,1 µs
Mtaps6	119,3 µs	124,1 µs

Es fällt auf, dass in der Niob-Strahlzeit die Auslesezeit des Tagger-Eventbuilders deutlich niedriger war. Dies begründet sich dadurch das während dieser Strahlzeit die 12 Taggerlatten mit den höchsten Raten nicht mit einer Hochspannung versorgt waren, während es in der Flüssigwasserstoff-Strahlzeit nur 2 Latten waren. Dadurch war die Anzahl der auszulesenden Wörter während der Niob-Strahlzeit auch deutlich niedriger. Ebenfalls fällt auf, dass die Forwardplug, Innendetektor und Crystal-Barrel-LEVBs in der Niob-Strahlzeit eine deutlich längere Totzeit haben. Durch die höhere Dichte und Massenzahl des Niob-Targets gibt es hier trotz des deutlich niedrigeren Strahlstroms eine erhöhte Ereignisrate und damit auch mehr Einträge in den Detektoren. In beiden Fällen lässt sich deutlich erkennen, dass die Auslese des Crystal-Barrel-Detektors selber bei weitem die längste Zeit in Anspruch nimmt. Dies liegt vor allem an der langen Konversionszeit der verwendeten LeCroy Fastbus 1885F Modulen von 265 µs sowie der Transferzeiten über das Fastbus- und das VME-Interface. Die mittlere Gesamtauslesezeit beträgt für den langsameren der beiden Crystal-Barrel-Hälften etwa 433 µs bzw. 366 µs. Dazu muss noch die in Kapitel 6.1 bestimmte Totzeit von 25,5 µs addiert werden, woraus sich ein Wert für die Gesamttotzeit von 458,5 µs bzw. 391,5 µs ergibt. Damit wäre die maximale Rate bei vernachlässigbarer Lifetime 2183 Hz bzw. 2554 Hz. Wenn diese Module durch schnellere Module ersetzt würden, wäre das nächste Limit durch die Gruppe von Tagger, Innendetektor und Forwardplug gegeben. Durch den langsamsten dieser drei Detektoren (den Tagger in der LH-Strahlzeit) berechnet sich die Totzeit zu 230,1 µs + 25,5 µs = 255,6 µs und daraus die maximale Rate zu 3912 Hz.

Die Auslese dieser drei Detektoren besteht aus Catch-TDC Modulen. Auch hier könnte durch einen Austausch ein Geschwindigkeitsgewinn erreicht werden. Damit wäre dann der Mini-Taps Detektor der langsamste Detektor. Die Totzeit dieses Detektors berechnet sich zu $130{,}4\,\mu s + 25{,}5\,\mu s = 155{,}9\,\mu s$ und die maximale Rate zu 6452 Hz. Um ausreichend Spielraum für zusätzliche Verbesserungen zu haben wurde als maximale Datenrate, für welche die Datenakquisition nach Modernisierung der Ausleseelektronik ausgelegt sein muss, eine Rate von 12 kHz gewählt.

6.3. Transfer unterschiedlicher Bankgrößen

Neben der Zeit die zur Auslese der Detektoren benötigt wird, kann auch der Abtransport und die Speicherung der Daten zu einer Erhöhung der Totzeit führen. Da der Transport der Daten parallel zur restlichen Datennahme läuft, kann er jedoch nur einen Einfluss haben, wenn die Daten schneller produziert werden, als sie über das Netzwerk transferiert werden können. Die Produktionsdatenrate hängt dabei zum einen von der Größe der einzelnen Ereignisse und zum anderen von der Ereignisrate ab. Die Datenrate des Abtransports wird durch drei Faktoren bestimmt. Diese sind die Geschwindigkeit des verwendeten Netzwerks, der Zeitverlust durch eine eventuell durchgeführte Komprimierung der Daten, sowie die Geschwindigkeit des Datenspeichers, der die zu speichernden Dateien aufnimmt.

Der wesentlichste Punkt ist die Geschwindigkeit des verwendeten Netzwerkes. Beim Crystal-Barrel/TAPS-Experiment sind die einzelnen Komponenten durchgehend mit Gigabit Ethernet angebunden, welches eine maximale Übertragung von 10^9 Bits in der Sekunde erlaubt. Dabei ist jedoch nicht die gesamte Bandbreite nutzbar, ein Teil geht dadurch verloren, dass die Daten für die verwendeten Übertragungsprotokolle in Zusatzinformationen gekapselt werden müssen. Bei einer Standardkonfiguration reduziert sich die Übertragung damit auf 941.482.440 Bits pro Sekunde. Damit wäre pro Sekunde theoretisch der Transport von 29.421.321 Wörtern mit 32 Bit Größe möglich.

Im Folgenden wurde nun untersucht, ob diese Rate auch mit der Datenakquisition erreicht werden kann. Dazu wurde nur der Trigger Eventbuilder benutzt, der künstlich auf die geforderte Ausleserate von 12 kHz begrenzt wurde. Dadurch würde sich für das benutze Gigabit Netzwerk eine Eventgröße von maximal etwa 2450 Wörtern mit 32 bit Größe ergeben. Es wurde nun eine Messreihe durchgeführt bei der von einem lokalen Eventbuilder eine festgelegt Anzahl von Wörtern übertragen wurde. Die Anzahl wurde dabei von 0 32-bit Wörtern bis 4000 Wörter in Schritten von 200 Wörtern erhöht. Diese Messung wurde für jede Größe mehrfach wiederholt und die Mittelwerte dieser Messwerte mit ihrer Standardabweichung in Abbildung 6.6 aufgetragen. Um zunächst ausschließlich

die Leistungsfähigkeit der Datenübertragung zu bewerten, wurden die Daten dabei weder komprimiert noch gespeichert.

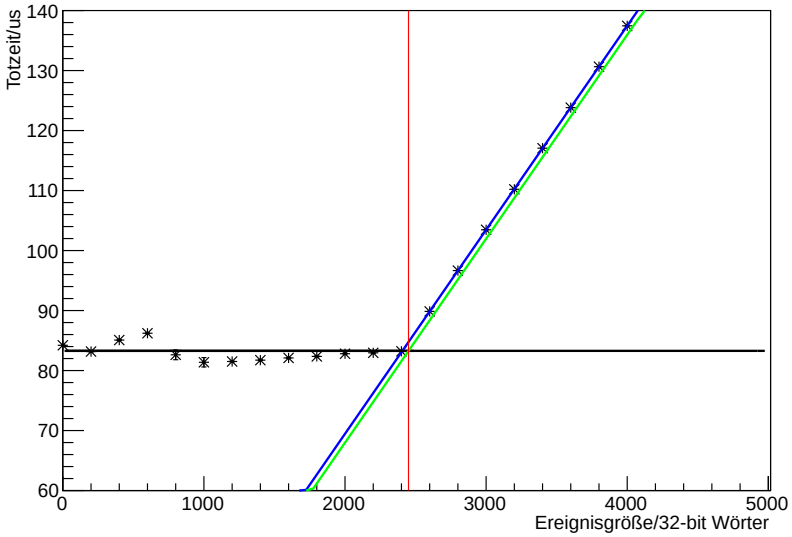


Abbildung 6.6.: **Abhängigkeit der Totzeit von der Eventgröße.** Die Totzeit bleibt konstant (schwarz) bis zur roten Linie ab der die Daten nicht mehr schnell genug über das Netzwerk abgeführt werden können. Ab da steigt sie linear an (blau). In grün dargestellt ist das berechnete Limit, dass sich aus der Bandbreite ergibt.

Es wurde nun zunächst der Bereich bis 2500 Wörter einer konstanten Funktion angepasst (schwarze Linie). Daraus ergab sich ein Mittelwert von $(83{,}29 \pm 0{,}02)$ µs. Die mittlere Rate war damit $(12\,006 \pm 3)$ Hz. Mit der oben bestimmten Rate von 29 421 321 Wörtern pro Sekunde ergab sich damit ein theoretisches Limit von $(2450{,}6 \pm 0{,}7)$ Wörtern die ohne Verzögerung übertragen werden können. Die Zahl wurde mit der senkrechten roten Linie markiert. Zusätzlich lässt sich aus der Rate von 29 421 321 Wörtern pro Sekunde abhängig von der Eventgröße eine Übertragungszeit berechnen. Diese Funktion wurde als grüne Linie in den Graphen eingezeichnet. Als letztes wurde noch an den Bereich ab 2500 Wörtern eine lineare Funktion angepasst, welche als blaue Line dargestellt ist. Die Datenpunkte folgen dabei mit leichten Abweichungen bis zur roten Linie der konstanten Funktion und steigen danach linear an. Das χ^2 der angepassten linearen Funktion betrug dabei 2,7. Der durch das Netzwerk bedingte Anstieg ist also gut mit der theoretischen Erwartung einer linearen Funktion vereinbar. Die angepasste blaue Funktion ist auch nur etwa 43 Wörter links neben der mit der maximal möglichen Übertragungsrate berechneten grünen Funktion, die Maximum wird also fast erreicht.

Abschließend wurde die Auswirkung der Kompression auf die Totzeit untersucht. Dazu wurde die Messung wiederholt, wobei nur zusätzlich eine Kompression mit dem gzip[2] Algorithmus hinzugefügt wurde. Gzip benutzt den Deflate-Algorithmus implementiert, welcher eine Kombination aus LZ77[3] und Huffman-Kodierung[4] ist. Dabei wurde die niedrigste Kompressionstufe eins benutzt, die am schnellsten kodiert, aber auch den niedrigste Kompressionsgrad erreicht. Das Ergebnis der Messung sieht man in Abbildung 6.7. Zum Vergleich sind die beiden angepassten Kurven aus 6.6 eingezeichnet. Im Gegensatz

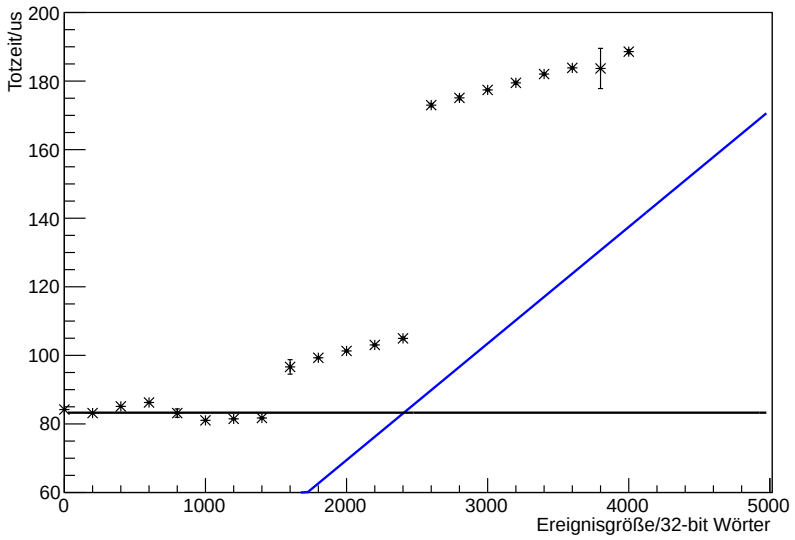


Abbildung 6.7.: **Abhängigkeit der Totzeit von der Eventgröße bei Komprimierung der Daten mit einer langsamen CPU.** In schwarz und blau dargestellt sind die angefitteten Kurven aus Abbildung 6.6.

zur Abbildung 6.6 ist hier die Totzeit nicht bis zur Grenze von 2450 Wörtern annähernd konstant, es ist stattdessen bereits ab 1600 Wörtern eine deutliche Abweichung zu erkennen, welche ab 2600 Wörtern nocheinmal deutlich zunimmt. Die Ursache dieser Sprünge ist, dass die Kompression der Daten von einem einzelnen Rechenprozess durchgeführt wird und der im Experiment für den saver Prozess verwendete Rechner (AMD Opteron 2222) unter Volllast bei einer Rate von 12 kHz nicht mehr als 1600 Wörtern pro Event komprimieren kann. Für die im Experiment auftretenden Datenmengen wäre dies wie sich später zeigt (Kapitel 6.5) ausreichend, um zu zeigen das es auch möglich ist die vom

[2] Frei erhältliches Open Source Kompressionsprogramm. www.gnu.org/software/gzip

[3] Lempel-Ziv 1977 Algorithmus. Dieser Algorithmus findet Wiederholungen von Abschnitten und markiert diese.

[4] Die Huffman-Kodierung ersetzt häufig vorkommende Zeichen durch Kodierungen, die weniger Speicherplatz als ein Zeichen verbrauchen.

Netzwerk unterstützte maximale Datenrate zu komprimieren, wurde diese Messung mit einem leistungfähigeren Rechner (Intel® Core™ i7-4790) wiederholt. Die Ergebnisse sind in Abbildung 6.8 gezeigt. Zum Vergleich sind wieder die beiden angepassten Kurven aus Abbildung 6.6 eingezeichnet. Eine quantitative Auswertung zeigt, das bis zum Abknick-

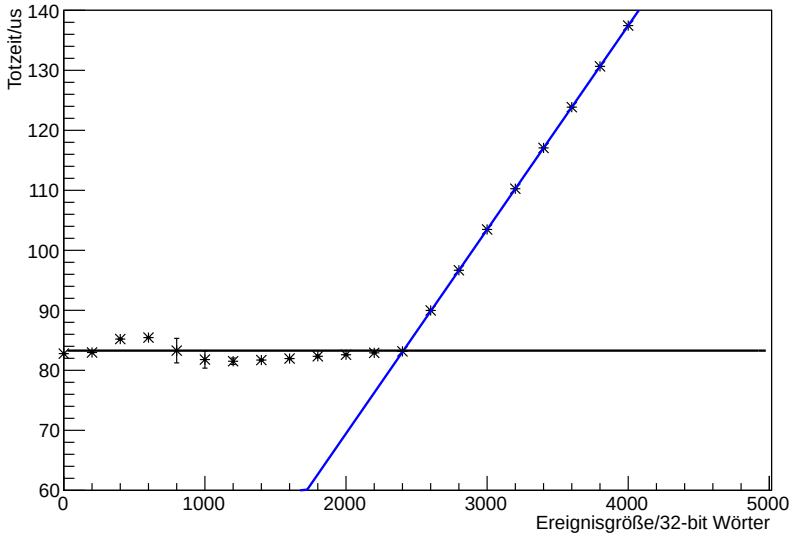


Abbildung 6.8.: **Abhängigkeit der Totzeit von der Eventgröße bei Komprimierung der Daten mit einer schnellen CPU.** In schwarz und blau dargestellt sind die angefitteten Kurven aus Abbildung 6.6.

punkt bei 2400 Wörtern die Werte mit Kompression im Rahmen der Messfehler mit den Werten ohne Kompression übereinstimmen. Das Komprimieren der Daten hat daher mit eine Rechner, der die nötige Leistungsfähigkeit besitzt, keine Auswirkung auf die Totzeit der Datennahme. Abschließend wurden die Daten noch auf Datenträger gespeichert. Es wurde dabei ein Festplattenverbund von 8 aktuellen Festplatten benutzt. Die Bandbreite der einzelnen Festplatten wurden einzeln mit $140\,\mathrm{MB\,s^{-1}}$ gemessen. Dies liegt mit $1120\,\mathrm{Mbit\,s^{-1}}$ schon über der maximalen Transferrate des Gigabit-Interfaces. Durch den Plattenverbund erhöht sich die Bandbreite der Festplatten noch einmal um einen Faktor von bis zu 7. Das Speichern auf dem Datenträger zeigte daher keinen nennenswerten Effekt. Bei der Verwendung von einer einzelnen Festplatte mit einer Bandbreite die in der Nähe oder unter der Bandbreite des Gigabit-Interfaces liegt könnte durch die Speicherung durchaus ein zusätzlicher Effekt entstehen. Es muss also sichergestellt sein, das es sich bei dem Ort auf den die Daten gespeichert werden um ein dafür ausgelegtes Ziel handelt. Bei Verwendung eines schnelleren Netzwerks muss zum Beispiel auch die Bandbreite der Festplatten angepasst werden.

6.4. Auswirkung der Änderung der Puffergrößen

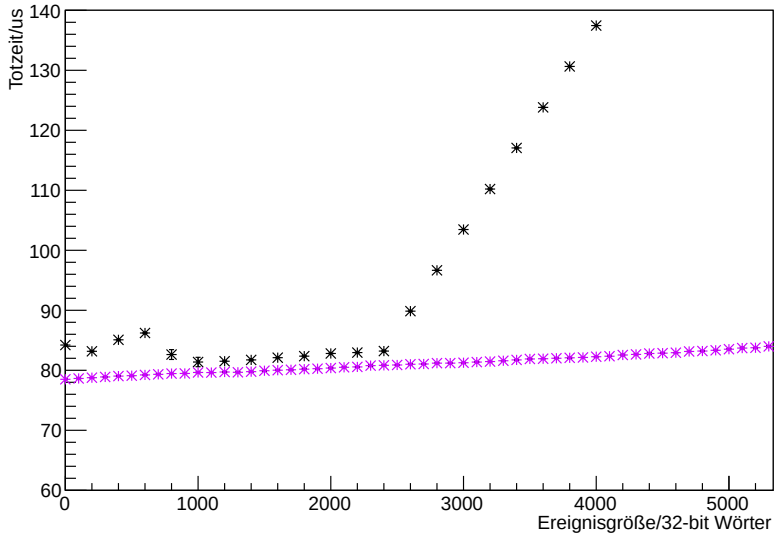


Abbildung 6.9.: **Abhängigkeit der Totzeit von der Eventgröße ohne Transport über das Netzwerk.** Die schwarzen Datenpunkte entsprechen denen aus Abbildung 6.6. In violett dargestellt sind die Datenpunkte ohne Transport über das Netzwerk.

Aufgrund des vom ELSA Beschleuniger verwendeten Beschleunigertypes (siehe Kapitel 2.1) wird an das Crystal-Barrel/TAPS-Experiment kein kontinuierlicher Strahl extrahiert. Stattdessen befindet sich zwischen zwei Extraktionen eine Pause von mehreren Sekunden, während der keine neuen Ereignisse aufgenommen werden. Wenn nun Daten, die während der Extraktion nicht übertragen werden können, zwischengespeichert werden, kann diese Pause zur Übertragung dieser Daten genutzt werden. In der normalen Konfiguration gibt es in den lokalen Eventbuildern 32 Puffer in denen die Daten vor dem Transport über das Netzwerk zwischengespeichert werden. In jeden Puffer kann dabei ein Ereignis gespeichert werden. Dadurch können kurze Lastspitzen im Netzwerk abgepuffert werden. Wenn man nun die Anzahl der Puffer vergrößert, sollte es durch Ausnutzung der Pause zwischen zwei Extraktionen möglich sein mehr Wörter pro Event zu übertragen, als prinzipiell durch die Gigabit-Ethernet-Grenze ermöglicht werden. Bevor dies jedoch getestet werden konnte, musste zunächst das Verhalten bei der Standard-Pufferzahl von 32 genauer betrachtet werden. In Abbildung 6.9 ist neben den schon bekannten Datenpunkten (schwarz) aus Abbildung 6.6 noch das Verhalten der Totzeit in Abhängigkeit von der Datengröße aufgetragen, wenn der Transport über das Netzwerk komplett ausgeschaltet wird. Es

sind dabei drei Bereiche zu sehen. Im ersten Bereich bis 1000 Wörtern pro Ereignis ist eine Erhöhung der Totzeit gegenüber der Kurve ohne Transport über das Netzwerk zu beobachten. Der Grund dieser Erhöhung ist vermutlich, dass kleine Datenpakete mit geringerer Effizienz übertragen werden, woraus sich eine geringere mittlere Datenrate ergibt und damit auch eine höhere Totzeit. Im Bereich von 1200 bis 2400 Datenwörtern folgen die schwarzen Datenpunkte den violetten, es entsteht in diesem Bereich also keine zusätzliche Totzeit durch den Transport über das Netzwerk. Im Bereich ab 2600 Datenwörtern weichen die schwarzen Datenpunkte dann deutlich ab, auf Grund der in Kapitel 6.3 betrachteten Limitierung durch die maximale Übertragungsrate des Gigabit-Netzwerkes. Im Folgenden wurden nun die Auswirkungen untersucht, wenn die Anzahl

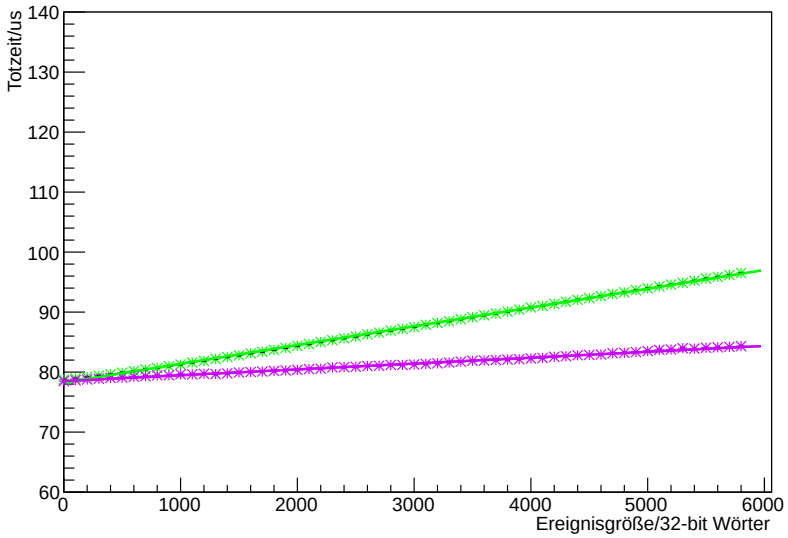


Abbildung 6.10.: **Auswirkung der Puffergröße auf die Abhängigkeit der Totzeit von der Eventgröße ohne Transport über das Netzwerk.** Dargestellt ist die Abhängigkeit der Totzeit von der Eventgröße, bei der Verwendung von 32 (violett) bzw. 20.000 (grün) Zwischenspeichern. Der Transport über das Netzwerk wurde in beiden Fällen ausgeschaltet.

der Zwischenspeicher deutlich auf 20.000 erhöht wurde. In Abbildung 6.10 ist zunächst der Unterschied dargestellt, wenn der Transport über das Netzwerk ausgeschaltet ist. Sowohl bei 32 Puffern, als auch bei 20.000 ist eine leichte Steigung zu erkennen, welche daraus entsteht, dass das Kopieren der Daten in den Zwischenspeicher eine gewisse Zeit erfordert. Um den Unterschied der beiden Steigungen zu erklären, wurde jeweils eine linear Funktion an die Datenpunkte angepasst. Für die Steigung ergab sich dabei ein Wert von $(1{,}0009 \pm 0{,}0035)\ \frac{\text{ns}}{32\text{bit}-\text{Wort}}$ bei 32 Puffern, beziehungsweise $(3{,}12941 \pm 0{,}0050)\ \frac{\text{ns}}{32\text{bit}-\text{Wort}}$

bei 20.000 Puffern. Da jedes 32 bit-Wort eine Größe von 4 Bytes besitzt ergibt sich daraus eine Datenrate von $(7993 \pm 28)\,\mathrm{MiB\,s^{-1}}$ [5] bei 32 Puffern, sowie $(2556 \pm 4)\,\mathrm{MiB\,s^{-1}}$ bei 20.000 Puffern. Der deutliche Unterschied dieser beiden Transferraten entsteht dadurch, dass im Fall von 32 Puffern die verwendete CPU einen schnelleren Speicherbereich (L2-Cache) benutzen kann. Dieser Bereich hat bei dem verwendeten Rechner (Intel® Mobile Core™ 2 T7400) eine Größe von 4048 KByte. Bei 32 Puffern a 32 KByte ergibt sich eine Gesamtgröße von 1024 KByte, was in den Bereich hineinpasst. Bei 20.000 Puffern ergibt sich eine Gesamtgröße von 640.000 KByte. Für einen Speicherbereich dieser Größe muss der Hauptspeicherbereich (RAM) benutzt werden.

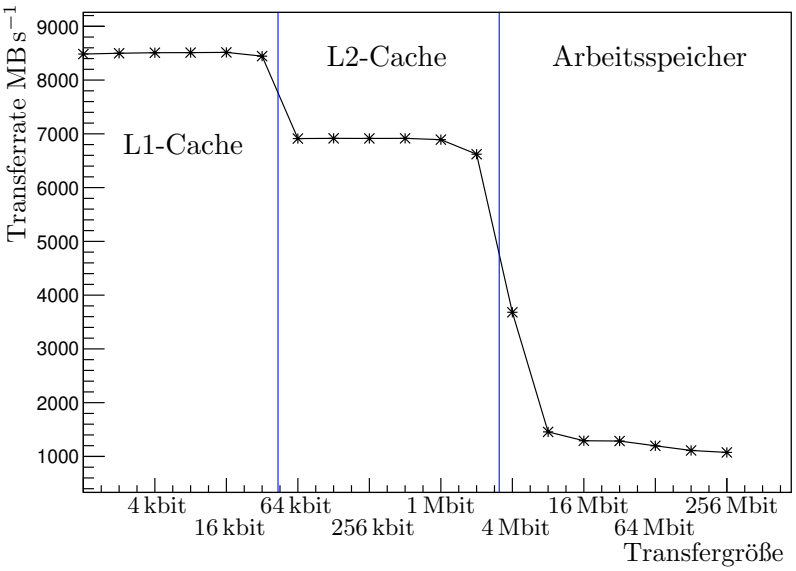


Abbildung 6.11.: **Übertragungsgeschwindigkeiten in den Speicher bei unterschiedlichen Transfergrößen.** Dargestellt ist das Ergebnis eines Testprogramms zur Ermittlung der Übertragungsgeschwindigkeiten. Im Bereich bis 32 bit kann der schnellste Teil des Speichers benutzt werden (L1-Cache), dann wird der L2-Cache benutzt. Ab 2048 bits kann dieser Bereich nicht mehr ausschließlich verwendet werden. Ab 16384 bits wir dann nur noch der langsamste Bereich (RAM) benutzt.

Um dies zu testen wurde mit dem Tool "Ramspeed"[6] die Zeit, die zum Schreiben von unterschiedlich großen Bereichen gebraucht wird, bestimmt (Abbildung 6.11). Deutlich zu erkennen sind drei verschieden schnelle Bereiche des Arbeitsspeichers. Bei zu schreibenden Größen bis 32 kbit kann hauptsächlich der schnellste Speicherbereich (L1-Cache) benutzt

[5] $1\ \mathrm{MiB} = 1 \cdot 10^6$ Bytes

[6] Freies Tool zum Testen der Geschwindigkeit von Speicherzugriffen. Entwickelt von der Firma Alasir. http://alasir.com/software/ramspeed/

werden. Dieser hat bei der verwendeten CPU eine Größe von 2x32 kB. Ab einer Größe von 64 kbit muss zusätzlich der etwas langsamere L2-Cache Bereich hinzugezogen werden. Ab 2048 kbit reicht dieser Bereich dann ebenfalls nicht mehr aus und es muss zusätzlich auch der normale Arbeitsspeicher verwendet werden. Der L2-Cache hat bei der verwendeten CPU eine Größe von 4 MB. Ab 16 384 kbit hat die Verwendung der schnelleren Cache Bereiche dann keine messbare Auswirkung mehr auf die Transfergeschwindigkeiten. Dieser Bereich beschreibt dann den Zugriff auf den Arbeitsspeicher. Dass die Größen der Caches nicht komplett erreicht werden, liegt daran, dass der Cache nicht komplett für ein vom Benutzer gestartetes Programm zur Verfügung stehen.

Der oben ermittelte Wert bei 20.000 Puffern von 2556 $\mathrm{MiB\,s^{-1}}$ entspricht 2438 $\mathrm{MB\,s^{-1}}$ und liegt etwas oberhalb des ermittelten Wertes für den Arbeitsspeicher. Es kann also durch geschickte Benutzung der Caches durch die CPU der Vorgang weiterhin beschleunigt werden. Der ermittelte Wert bei 32 Puffern von 7993 $\mathrm{MiB\,s^{-1}}$ entspricht 7623 $\mathrm{MB\,s^{-1}}$ und liegt damit zwischen den gemessenen Werten beim Zugriff auf den L1- und L2-Cache. Es kann also auch hier durch geschickte Verwendung des L1-Cache der Zugriff beschleunigt werden.

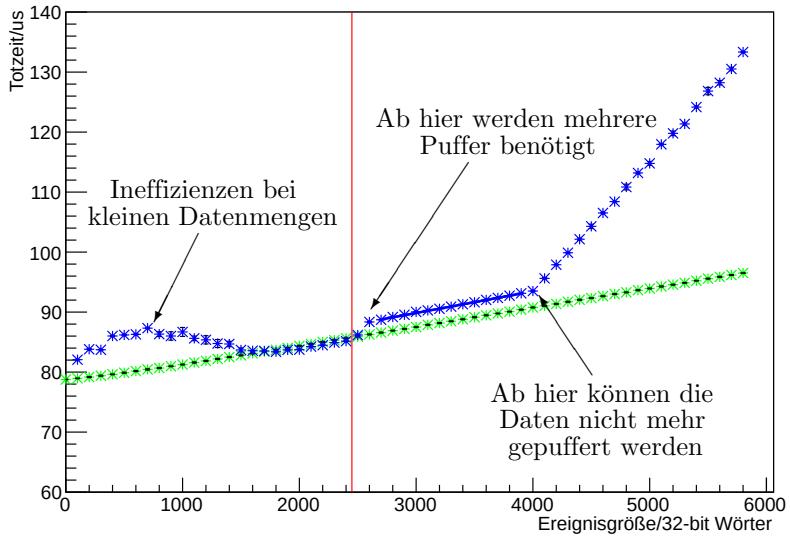


Abbildung 6.12.: **Abhängigkeit der Totzeit von der Eventgröße bei 20.000 Puffern.** Die blauen Datenpunkte beschreiben das Verhalten bei 20.000 Puffern mit Übertragung über das Netzwerk. In grün dargestellt sind die Datenpunkte ohne Transport über das Netzwerk.

Nun wurde der Datentransport über das Netzwerk wieder eingeschaltet und das Ergebnis zunächst mit den Datenpunkten bei 20.000 Puffern ohne Datentransport

verglichen (Abbildung 6.12). Es wurde wieder die Kompression abgeschaltet und die Daten nicht gespeichert. Bei der Datenerzeugung wurde dabei ein Signal verwendet, das eine typische Extraktionszeitstruktur (siehe auch Kapitel 2.1) nachempfindet. Diese Signal simuliert eine 1,8 Sekunden Extraktionspause, gefolgt von einer Extraktionszeit von 4 Sekunden. Analog zum dem Ergebniss bei 32 Puffern gibt es wieder einen Bereich bei kleinen Datengrößen, in dem eine erhöhte Totzeit zu beobachten ist. Im Bereich von 1600 bis 2400 folgen die Datenpunkte dann den Werten ohne Netzwerktransport. Ab dem Bereich wo die Daten ohne Puffer nicht mehr schnell genug übertragen werden können (rote Linie), ist eine zusätzliche Totzeit zu beobachten. Ab diesem Punkt können die Datenpakete nicht mehr direkt über das Netzwerk gesendet werden. Da hierdurch mehr als ein 32 kB Puffer gleichzeitig benutzt wird, kann der L1-Cache von 2x32 kB nicht mehr so effektiv genutzt werden. Dadurch entsteht eine weitere Totzeit. Ab einer Datengröße von 4000 Wörtern reicht der Puffer dann nicht mehr um die Daten zwischenzuspeichern und die Datenpunkte knicken erneut ab.

Dass der Zwischenspeicher nicht mehr ausreicht kann dabei zwei Gründe haben. Zum Einen kann die Pufferzahl von 20.000 nicht mehr ausreicht um die nicht übertragenen Ereignisse zwischenzuspeichern. Zum Anderen könnte auch der Zwischenspeicher in der Pause nicht mehr vollständig geleert werden. Im Folgenden soll nun die Eventgröße bestimmt werden ab der diese Effekte auftreten. Dazu muss zunächst die Rate bestimmt werden mit der Daten abhängig von der Eventgröße produziert werden. Da die oben betrachteten Effekte des Arbeitsspeichers die Ereignisrate abhängig von der Eventgröße reduzieren, müssen diese Effekte mit betrachtet werden. Dazu wurde in Abbildung 6.12 im Bereich zwischen 2700 und 3900 Wörtern pro Ereignis, in welchen beide Effekte aktiv sind, eine lineare Funktion an die Daten angepasst. Das Ergebniss war eine Totzeit von

$$\begin{aligned} t_{tot} &= a + b \cdot S \\ mit \quad a &= (79{,}04 \pm 0.11)\mu s \\ und \quad b &= (3.605 \pm 0.034)\ \frac{\text{ns}}{\text{Wörter}} \end{aligned} \tag{6.1}$$

Wobei S die Anzahl der Wörter pro Ereignis ist. Die Rate der produzierten Ereignisse bestimmt sich dann zu:

$$R_{prod} = \frac{1}{t_{tot}} \tag{6.2}$$

Die benutzten Buffer sind nun

$$N_{buf} = t \cdot R_{prod} - \frac{t}{S} \cdot R_{GB} \tag{6.3}$$

mit $R_{GB} = 29.421.321 \frac{\text{Wörter}}{\text{s}}$ der Transportkapazität vom Gigabit Netzwerk. Wenn man

nun 6.1 und 6.2 in 6.3 einsetzt und nach S umformt erhält man:

$$N_{buf} = t \cdot \frac{1}{a + b \cdot S} - t \cdot \frac{R_{GB}}{S}$$

Dies lässt sich umformen zu:

$$\begin{aligned}
&\Leftrightarrow \frac{N_{buf}}{t} \cdot (a + b \cdot S) \cdot S = S - (a + b \cdot S) \cdot R_{GB} \\
&\Leftrightarrow \frac{N_{buf}}{t} \cdot b \cdot S^2 + \left(\frac{N_{buf}}{t} \cdot a - 1 + b \cdot R_{GB} \right) \cdot S + a \cdot R_{GB} = 0 \\
&\Leftrightarrow S^2 + \left(\frac{a}{b} + \left(R_{GB} - \frac{1}{b} \right) \cdot \frac{t}{N_{buf}} \right) \cdot S + \frac{t}{N_{buf}} \cdot \frac{a}{b} \cdot R_{GB} = 0 \\
&\Rightarrow S = -\frac{1}{2} \cdot \left(\frac{a}{b} + \left(R_{GB} - \frac{1}{b} \right) \cdot \frac{t}{N_{buf}} \right) \\
&\quad \pm \sqrt{\left(\frac{1}{2} \cdot \left(\frac{a}{b} + \left(R_{GB} - \frac{1}{b} \right) \cdot \frac{t}{N_{buf}} \right) \right)^2 - \frac{t}{N_{buf}} \cdot \frac{a}{b} \cdot R_{GB}}
\end{aligned}$$

Nach Einsetzen der Spillzeit $t = 4\,\mathrm{s}$ und der Bufferzahl $N_{buf} = 20.000$ erhält man $S = 5936{,}33$ Wörter sowie $S = 21\,732{,}8$ Wörter. Bei 5937 Wörter pro Ereignis würden die 20.000 Buffer also nicht mehr ausreichen. Um nun zu betrachten, ab welcher Größe die Ereignisse während der Spillpause nicht mehr weggeschrieben werden können muss zunächst die benutze Zahl der Puffer N_{Buf} abhängig von der Größe der Ereignisse bestimmt werden. Dazu wird die Zahl der abtransportierten Ereignisse während der Extraktion von den Zahl der anfallenden Ereignissen subtrahiert:

$$N_{Buf} = 4 \cdot R_{prod} - \frac{4 \cdot R_{GB}}{S} \tag{6.4}$$

S bezeichnet dabei die Größe der Ereignisse in Wörtern. Diese Formel gilt solange die Zahl der Puffer nicht ausgenutzt wird also nur für S < 5937 Wörter. Jetzt muss bestimmt werden wie lange es dauert diese Puffer zu übertragen:

$$t_{trans} = \frac{N_{Buf} \cdot S}{R_{GB}} \tag{6.5}$$

Im Grenzfall ist die Zeit t_{trans} die zur Übertragung nötig ist gleich den 1,8 Sekunden der Pause. Wenn man diesen Wert für t_{trans} einsetzt und Formel 6.4 in 6.5 einsetzt erhält man:

$$1{,}8 = \frac{S}{R_{GB}} \cdot \left(4 \cdot R_{prod} - \frac{4 \cdot R_{GB}}{S} \right) \tag{6.6}$$

Dies lässt sich umformen zu

$$\begin{aligned} 1{,}8 &= \frac{4 \cdot R_{prod}}{R_{GB}} \cdot S - 4 \\ \Leftrightarrow S &= (1{,}8 + 4) \cdot \frac{R_{GB}}{4 \cdot R_{prod}} \end{aligned}$$

Nun wird noch Formel 6.1 und 6.2 eingesetzt und man erhält:

$$\begin{aligned} 1{,}8 &= \frac{4}{(a + b \cdot S) \cdot R_{GB}} \cdot S - 4 \\ 1{,}8 \cdot (a + b \cdot S) &= \frac{4}{R_{GB}} \cdot S - 4 \cdot (a + b \cdot S) \\ 1{,}8 \cdot b \cdot S - \frac{4}{R_{GB}} \cdot S + 4 \cdot b \cdot S &= -1{,}8 \cdot a - 4 \cdot a \\ \left(5{,}8 \cdot b - \frac{4}{R_{GB}}\right) \cdot S &= -5{,}8 \cdot a \\ S &= \frac{5{,}8 \cdot a}{\frac{4}{R_{GB}} - 5{,}8 \cdot b} \\ \Leftrightarrow S &= 3984{,}74 \,\text{Wörter} \end{aligned} \tag{6.7}$$

Es folgt also, dass ab 3985 Wörtern die Daten während der Spillpause nicht mehr weg transportiert werden können. Dies ist auch in Abbildung 6.12 zu sehen. Abschließend sind noch einmal die Datenpunkte für 32 und 20.000 Puffer zusammen in einem Diagramm aufgetragen (Abbildung 6.13). Aufgetragen sind dabei mit schwarzen Punkten die Datenpunkte bei 32 Puffern zum Zwischenspeichern der Ereignisse. Diese Punkte sind identisch zu denen in Abbildung 6.6. Die neuen Datenpunkte bei 20.000 Puffern sind mit violetten Punkten hinzugefügt. Die schwarze und blaue Linie sind wieder die identischen Anpassungen an die Datenpunkte aus den vorherigen Abbildungen. In Rot ist der berechnete Grenzwert aus Formel 6.4 aufgetragen. Dieser stimmt mit dem Abknickpunkt der Daten überein. Wenn man nun eine durch den Datentransport bedingte zusätzliche Totzeit von 10 µs akzeptiert (orange Linie) erhöht sich, durch die Verwendung des großen Puffers, die mögliche Eventgröße von etwa 2500 auf etwa 4000 Wörter. Bei größeren Ereignissen nimmt die mögliche Ereignisrate langsam ab, sie ist aber aufgrund der niedrigeren Totzeit immer noch größer als mit kleinen Puffern.

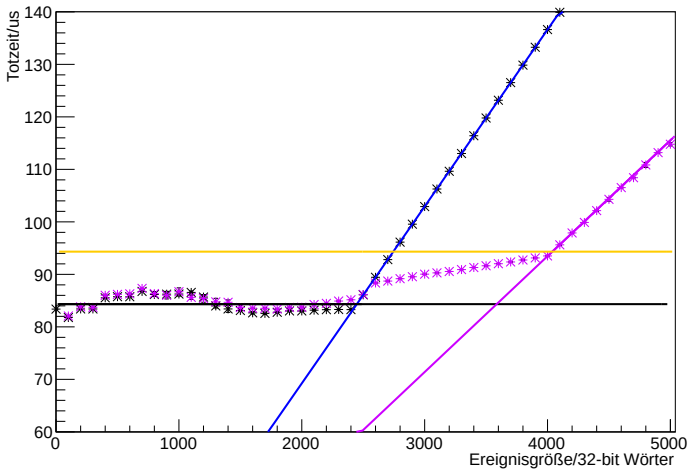


Abbildung 6.13.: **Auswirkung der Puffergröße auf die Abhängigkeit der Totzeit von der Eventgröße mit Transport über das Netzwerk.** Aufgetragen sind in schwarz die Datenpunkte bei einer normalen Pufferzahl von 32 Events, sowie in violett die Datenpunkte bei 20.000 Puffern. Die blaue und violette Linie ist eine lineare Anpassung an die jeweiligen Datenpunkte. Die orangene Linie liegt 10 µs über der schwarzen Linie.

6.5. Eignung der Datenakquisition für die Ereignisgrößen im Crystal-Barrel/TAPS-Experiment

Um zu entscheiden, ob die entwickelte Datenakquisition für das Crystal-Barrel/TAPS-Experiment geeignet ist, muss zunächst bestimmt werden, wie viele 32bit-Wörter im Mittel in jedem Event gespeichert werden. Dazu wurde zusätzlich zu den in Kapitel 6.2 benutzten Datenblöcken aus den Strahlzeiten ‘‘August 2013’’(158291) und ‘‘November 2013’’(163211) noch ein Datenblock aus der ‘‘Oktober 2010’’-Strahlzeit betrachtet. In dieser Strahlzeit wurde ein transversal polarisiertes Butanol Target benutzt, bei dem durch das Magnetfeld des Targets Positronen und Elektronen in die Detektoren in Vorwärtsrichtung abgelenkt werden. Der untersuchte Datenblock war 144993 mit einem extrahierten mittleren Strahlstrom von 230 pA. In der ‘‘August 2013’’-Strahlzeit wurde mit 640 pA ein deutlich größerer Strahlstrom extrahiert, was zu vielen Treffern in den Detektoren sorgte, vor allem im Tagger-Detektor. In der ‘‘November 2013’’-Strahlzeit wurde ein Niob-Target benutzt, welches auf Grund seiner hohen Dichte und Massenzahl, hohen Wirkungsquerschnitte hat und damit auch viele Treffer in den Detektoren erzeugt.

In allen drei Datenblocks kommt es daher aus unterschiedlichen Gründen zu sehr großen Ereignissen. In der folgenden Tabelle sind die mittlere Ereignisgröße und die Standardabweichung mit der die Ereignisgröße um den Mittelwert schwankt aufgeführt:

Datenblock	Target	Mittlere Größe	Standardabweichung
144993	Butanol	1378 Wörter	50,9 Wörter
158291	LH	1449 Wörter	79,62 Wörter
163211	Niob	1358 Wörter	54,87 Wörter

Damit sind in der Flüssigwasserstoff-Strahlzeit aus dem August 2013 mit 1449 32bit-Wörtern die größten Ereignisse aufgetreten. In Kapitel 6.2 wurde ermittelt, dass um Spielraum für zukünftige Verbesserungen der Ausleseelektronik zu haben, es möglich sein sollte, eine Ereignisrate von 12 kHz zu erreichen. Es wurde in Kapitel 6.2 ebenfalls bestimmt, dass mit der in dieser Arbeit vorgestellten Datenakquisition bei der Rate von 12 kHz im Mittel 2450 32-bit Wörter gespeichert werden können. Dies liegt deutlich über der maximal anfallenden mittleren Ereignisgröße von 1449 32-bit Wörtern, welche in der Flüssigwasserstoff-Strahlzeit im August 2013 ermittelt wurde. Es besteht hier sogar noch die Möglichkeit durch zukünftige Erweiterungen des Experiments die mittlere Ereignisgröße um fast einen Faktor von 1,7 zu vergrößern. Falls die Ereignisrate die 12 kHz nicht erreichen sollte, wird dieser Faktor entsprechend größer. In Kapitel 6.3

wurde ermittelt dass mit dem Crystal-Barrel/TAPS-Experiment, durch die Auslese der Daten aus den Auslesemodulen, ohne Umbau eine maximale Ereignisrate von etwa 2600 Hz zu erreichen ist. Selbst mit Austausch eines großen Teils der Komponenten des Experiments ist nur eine Rate von 6500 Hz zu erreichen. Bei der aktuell möglichen Ereignisrate ergibt sich damit ein zusätzlicher Faktor von mindestens 4,6. Daher besteht auch hier noch ausreichend Spielraum für Erweiterungen. Falls die Ereignisgröße von 2450 32-Bit Wörtern bei 12 kHz doch nicht ausreichen sollten, gibt es noch zwei einfache Möglichkeiten die Leistungsfähigkeit der Datenakquisition anzupassen. Zum Einen wurde in Kapitel 6.4 gezeigt, dass durch ein Ausnutzen der Extraktionspause der ELSA die mittlere Ereignisgröße auf etwa 4000 32-bit Wörter möglich ist. Zum Anderen kann auch das verwendete Netzwerk auf ein 10-Gigabit Netzwerk umgebaut werden. Dadurch würde sich die mögliche Übertragungsrate deutlich erhöhen und damit auch die mögliche mittlere Ereignisgröße. Es ist jedoch nicht zu erwarten, dass sich dafür in absehbarer Zeit eine Notwendigkeit ergeben wird.

Abschließend ist zu sagen, dass die vorgestellte Datenakquisition die Anforderungen des Crystal-Barrel/TAPS-Experiments vollständig erfüllt und sogar beim aktuellen Aufbau noch mindestens einen Faktor 7,8 Reserve bietet, welcher für Verbesserungen im Experiment genutzt werden kann.

7. Physikalische Ergebnisse

Eine erste Version der im Rahmen dieser Doktorarbeit entwickelten Datenakquisition wurde ab dem Jahr 2007 im Crystal-Barrel/TAPS-Experiment eingesetzt. Seitdem wurde in einer Vielzahl von Strahlzeiten Daten für verschiedene Messprogramme genommen:

Strahlzeit	Target	Polarisation Target	Polarisation Strahl
März 2007	Kohlenstoff	-	-
September 2007	Butanol	Longitudinal	Zirkular
November 2007	Butanol	Longitudinal	Zirkular
April 2008	Butanol	Longitudinal	Linear
August 2008	Butanol	Longitudinal	Linear
Oktober 2008	LH2	-	-
November 2008	LH2	-	Zirkular
Dezember 2008	Deuterium	-	Zirkular
Januar 2009	Kohlenstoff	-	-
Januar 2009	Kohlenstoff	-	Linear
Mai 2009	Butanol	Longitudinal	Zirkular
August 2009	Butanol	Longitudinal	Zirkular
August 2009	Butanol	Longitudinal	Linear
September 2009	Butanol	Longitudinal	Zirkular
November 2009	Butanol	Longitudinal	Zirkular
Juli 2010	Butanol	Transversal	Linear
Oktober 2010	Butanol	Transversal	Linear
Januar 2011	D-Butanol	Longitudinal	Zirkular
Juni 2011	D-Butanol	Longitudinal	Zirkular
November 2011	Kohlenstoff	-	Linear
November 2011	Kohlenstoff	-	Zirkular
Juli 2013	LH2	-	Linear
August 2013	LH2	-	Linear
September 2013	LH2	-	Linear
Oktober 2013	LH2	-	Linear
November 2013	Niob	-	-
Dezember 2013	Niob	-	-
Januar 2014	Niob	-	-

Die Leistungsfähigkeit der Datenakquisition wurde dabei über die Zeit kontinuierlich gesteigert. Dazu wurde zunächst die pro Ereignis auftretende Totzeit reduziert (siehe Kapitel 6.1). Die dadurch resultierende Verbesserung der gespeicherten Ereignisrate ist in Abbildung 7.1 gezeigt. Es ist in der Strahlzeit von 2010 eine deutliche Erhöhung der

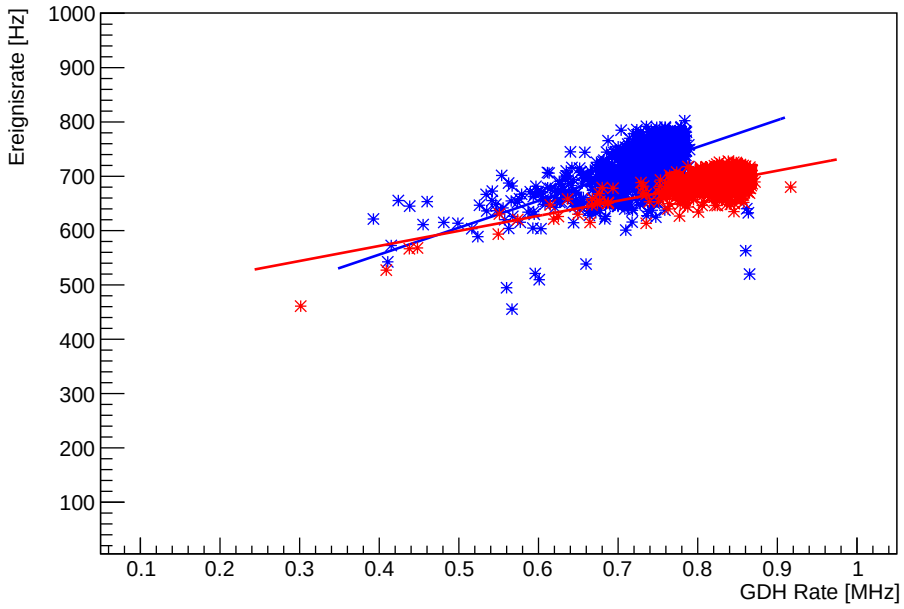
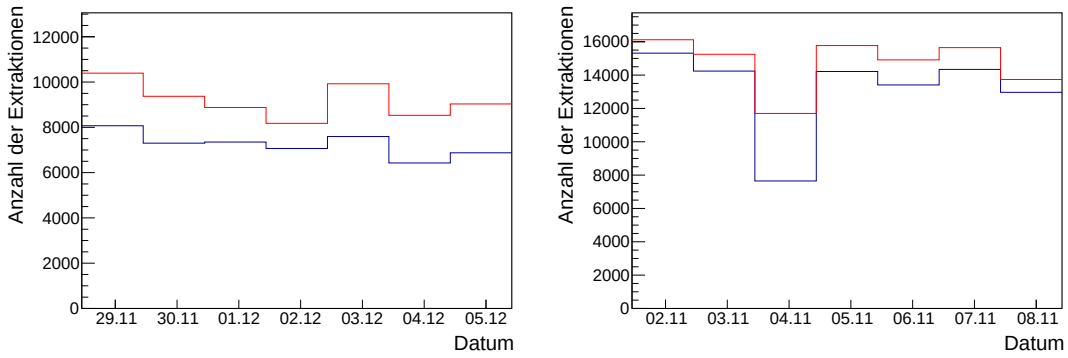


Abbildung 7.1.: **Ereignisrate gegen den Strahlstrom aufgetragen für die Strahlzeiten "Juli 2010" (Blau) und "November 2007" (rot).** Dabei wurde die GDH Rate, welche die Rate eines Teiles des Tagger-Detektors (Kapitel 2.3)ist, als Maß für den Strahlstrom genommen. Es ist in der Strahlzeit "Juli 2020" eine deutliche Erhöhung der Ereignisrate bei gleichem Strahlstrom zu sehen.

Ereignisrate, bei gleichem extrahierten Strahlstrom vom Teilchenbeschleuniger ELSA, zu erkennen.

Des Weiteren wurden die Ausleseroutinen der lokalen Eventbuilder verbessert, um auftretende Fehler in den TDCs und ADCs besser zu kompensieren. Dadurch wurde die Zeit reduziert, in welcher Probleme diagnostiziert wurden und die Firmware von den Auslesekarten neu geladen wurde. In dieser Zeit kann die Datennahme nicht laufen und von daher konnten auch keine Ereignisse gespeichert werden. Als weitere Verbesserung wurden die vom Schichtpersonal während der Datennahme zu erledigenden Aufgaben weitgehend automatisiert (Kapitel 5.4.3). So wurde zum Beispiel das Verstellen des Goniometers in Strahlzeiten mit Linearpolarisation und das starten von neuen Daten-



(a) Strahlzeit im November 2007

(b) Strahlzeit im Oktober 2013

Abbildung 7.2.: **Anteil der gespeicherten Extraktionen pro Tag für zwei Strahlzeiten.** Aufgetragen ist jeweils die Zahl der gespeicherten Extraktionen für jeden Tag. Zum Vergleich ist in Rot die Anzahl der vom Beschleuniger gelieferten Extraktionen gezeigt. Am 04.11.2013 ist der Wert der gespeicherten Extraktionen kleiner, da an diesem Tag der Beschleuniger für 6,5 Stunden ausgeschaltet war.

blöcken automatisiert. Dadurch konnten diese Aufgaben zum Einen in einer kürzeren Zeit erledigt werden, zum Anderen waren diese Schritte dadurch auch weniger fehleranfällig. Durch die Reduzierung der Fehleranfälligkeit und die Automatisierung der Abläufe bei der Datennahme konnte der Anteil der Strahlzeit, bei dem Ereignisse gespeichert werden, verbessert werden. Abbildung 7.2 zeigt die Auswirkung dieser Verbesserungen. Dort werden für zwei Strahlzeiten die gespeicherten Extraktionen mit den vom Beschleuniger gelieferten Extraktionen verglichen. Dabei ist jeweils eine repräsentative Wochen gezeigt, in der keine großen Unterbrechungen des Strahlbetriebs stattgefunden haben. Der Prozentsatz der gespeicherten Extraktionen hat sich dabei von 79% auf 92,4% vergrößert, wenn in der ‘‘Oktober 2013’’-Strahlzeit der Tag mit dem Teilchenbeschleunigerausfall vernachlässigt wird.

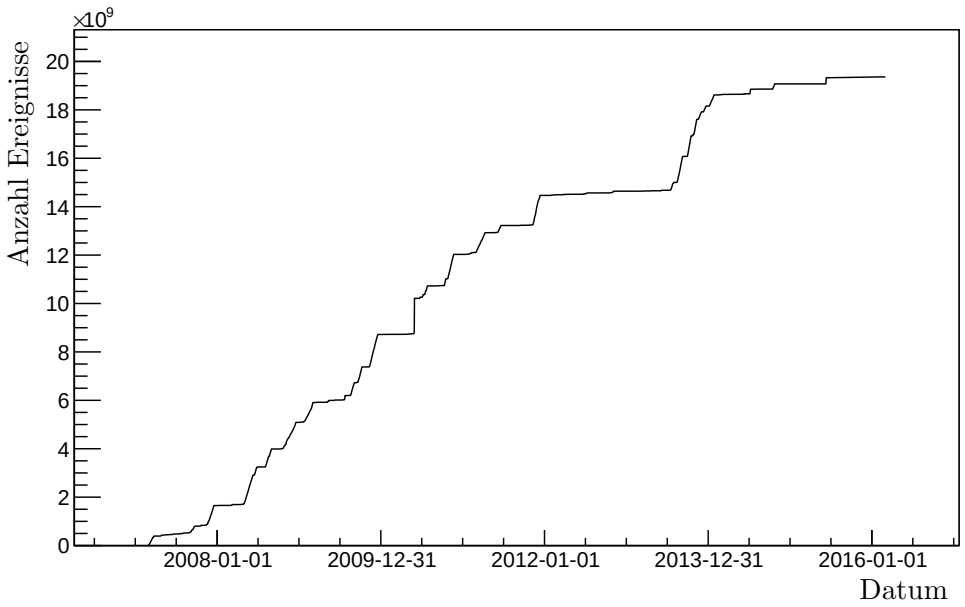


Abbildung 7.3.: **Summierte Anzahl der mit der Datenakquisition des Crystal-Barrel/TAPS-Experiments gespeicherter Ereignisse, aufgetragen gegen die Zeit.**

Insgesamt wurden in den Strahlzeiten bis zum Jahr 2016 mehr als 19 Milliarden Ereignisse gespeichert (siehe Abbildung 7.3). Im Folgenden werden einige der in diesen Strahlzeiten mit der hier vorgestellten Datenakquisition erzielten Ergebnisse vorgestellt.

7.1. Reaktionskanal $\gamma p \to p\pi^0$

Der Zerfallskanal mit dem höchsten Wirkungsquerschnitt ist der Kanal $\gamma p \to p\pi^0 \to p\gamma\gamma$ (siehe Kapitel 1). In diesem Kanal zerfällt das angeregte Teilchen (N^* oder Δ^*) zunächst in ein Proton und ein neutrales Pion. Das π^0 ist das leichteste Meson mit einer Ruhemasse von $(134{,}9766 \pm 0{,}0006)\,\mathrm{MeV}$ [Pat+16] und es findet hier ein neutraler Zerfall ohne Ladungsübertrag statt. Das π^0 zerfällt in diesem Kanal weiter in zwei γ-Quanten. Dieser Zerfall hat mit über 98% die höchste Übergangswahrscheinlichkeit [Pat+16]. Dadurch werden insgesamt zwei ungeladene (γ) und ein geladenes Teilchen (p) im Endzustand erzeugt. Das Crystal-Barrel/TAPS-Experiment ist optimal geeignet zum Nachweis von γ-Teilchen. Durch den hohen Wirkungsquerschnitt ist es mit diesem Kanal am Crystal-Barrel/TAPS-Experiment am einfachsten, eine aussagekräftige Statistik von Ereignissen anzusammeln. Daher wurde in den meisten Strahlzeiten auch ein Datentrigger benutzt, der auf die Detektion dieser drei Teilchen im Endzustand ausgelegt ist. Aus den mit der hier vorgestellten Datenakquisition gespeicherten Daten wurden im Rahmen mehrerer Doktorarbeiten [Thi12; Got13; Har17] in diesem Kanal die Polarisationsobservablen G, E, T, P und H bestimmt, deren Veröffentlichungen in Tabelle 7.1 gelistet sind. Zusätzlich befindet sich eine Dissertation in Vorbereitung, welche die Strahlasymmetrie Σ analysiert [Afz18b].

Autor	Titel	Referenz
A. Thiel et al.	Well-established nucleon resonances revisited by double-polarization measurements	[Thi+12]
J. Hartmann et al.	The $N(1520)3/2^-$ helicity amplitudes from an energy-independent multipole analysis based on new polarization data on photoproduction of neutral pions	[H+14b]
M. Gottschall et al.	First measurement of the helicity asymmetry for $\gamma p \to p\pi^0$ in the resonance region	[Got+14]
J. Hartmann et al.	The polarization observables T , P , and H and their impact on $\gamma p \to p\pi^0$ multipoles	[Har+15]
A. Thiel et al.	Double-polarization observable G in neutral-pion photoproduction off the proton	[Thi+17]

Tabelle 7.1.: Publizierte Veröffentlichungen für den Kanal $\gamma p \to p\pi^0$.

In Abbildung 7.4 sind beispielhaft die Winkelverteilungen in einigen gemessenen Energiebereichen für die Polarisationsobservable G, sowie Vorhersagen einiger Partialwellenanalysen (PWAs) gezeigt. Die aufgetragenen Partialwellenanalysen (siehe Kapitel 1.4) sind Bonn-Gatchina [A+], Jülich-Bonn [Rön+], SAID [Bri+] und MAID [Tia+].

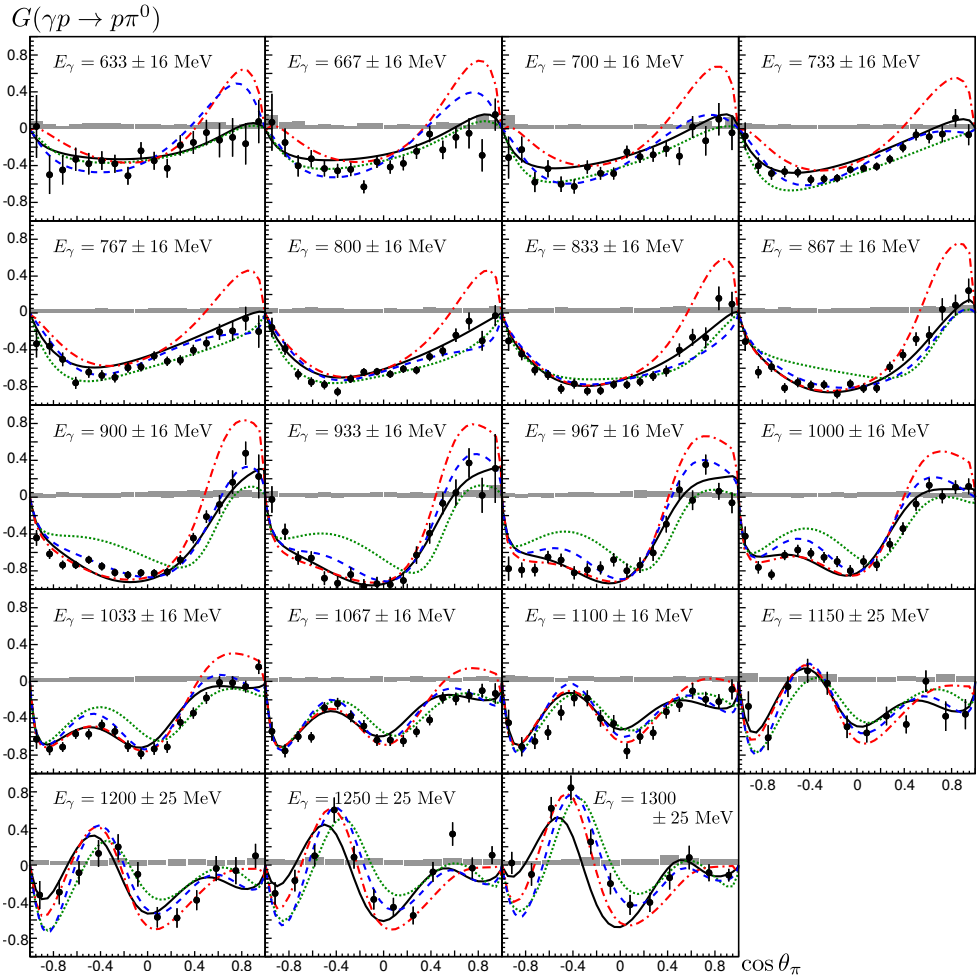
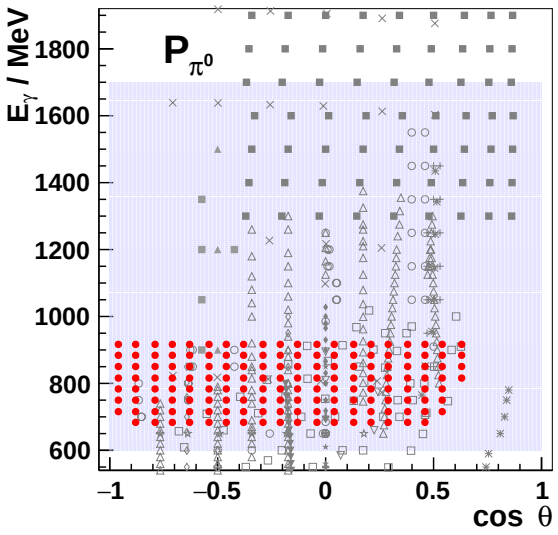


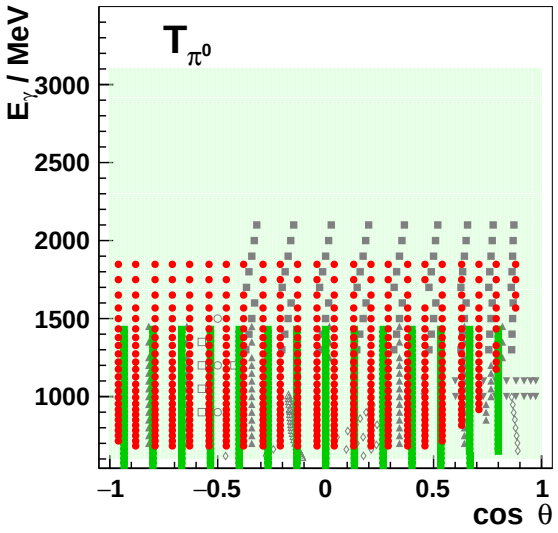
Abbildung 7.4.: **Doppelpolarisationsobservable G in $\gamma p \to p\pi^0$.** Gezeigt sind die mit dem Crystal-Barrel/TAPS-Experiment gemessenen Werte (Punkte) und die Vorhersagen verschiedener Partialwellenanalysen. Die Linien sind dabei BnGa 2011-02 [Ani+12](Schwarz), SAID CM12 [Wor+12](rot), MAID 2007 [Dre+99; Hil+13](grün) und JüBo 2013-01 [Rön+14](Blau). Bild aus [Thi+17].

Vom Crystal-Barrel/TAPS-Experiment wurde die Doppelpolarisationsobservablen G dabei erstmals über einen großen Energie- und Winkelbereich gemessen. Es sind dabei deutliche Abweichungen der Vorhersagen zu den gemessenen Daten zu erkennen. Dies unterstreicht noch einmal die Bedeutung der Messungen für ein besseres Verständnis der zu Grunde liegenden Multipole.

Insgesamt konnte durch die genommenen Daten die Datenbasis der Polarisationsobservablen deutlich verbessert werden. In den Abbildungen 7.5 und 7.6 ist die Datenbasis nach den mit dieser Datenakquisition durchgeführten Strahlzeiten gezeigt. Zum Vergleich ist die Datenbasis vor den Strahlzeiten (wie auch in den Abbildungen 1.9 und 1.10 in Kapitel 1.5 gezeigt) hier ebenfalls noch einmal eingetragen.

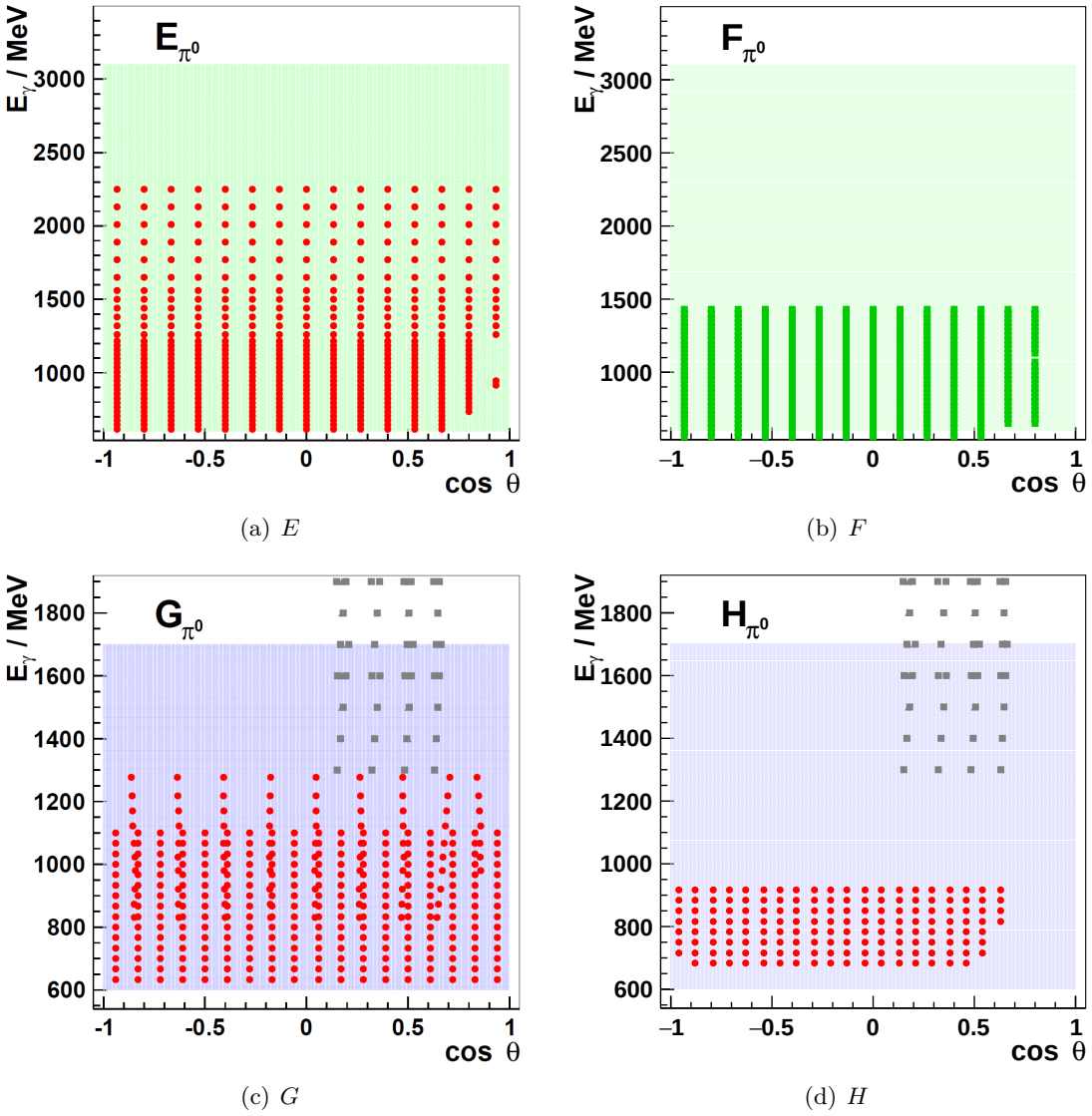


(a) P



(b) T

Abbildung 7.5.: **Polarisationsobservablen der Rückstoßpolarisation P und Targetasymmetrie T im Kanal $\gamma p \to p\pi^0$.** Gezeigt ist die Datenbasis vor den neuen Messungen mit dem Crystal-Barrel/TAPS-Experiment in grau und die neu dazugekommenen Messpunkte in rot. Bild aus [Afz18a].



(a) E (b) F

(c) G (d) H

Abbildung 7.6.: **Doppelpolarisationsobservablen bei polarisiertem Strahl und polarisiertem Target E, F, G und H im Kanal $\gamma p \to p\pi^0$.** Gezeigt ist die Datenbasis vor den neuen Messungen des Crystal-Barrel/TAPS-Experiments in grau und die neuen Messpunkte in rot. Bild aus [Afz18a].

In allen gezeigten Observablen bis auf F konnte die Datenbasis des Kanals $\gamma p \to p\pi^0$ durch die Ergebnisse der Analysen der neu gemessenen Daten des Crystal-Barrel/TAPS-Experiments wesentlich erweitert werden. Besonders in den Doppelpolarisationsobservablen E, G und H konnten die neuen Datenpunkte den gemessenen Winkel und Energiebereich deutlich vergrößern. Für die Observable Σ sind die Daten des Crystal-Barrel/TAPS-Experiments sogar die einzigen bisher in diesem Kanal gemessenen Daten.

7.2. Weitere Reaktionskanäle

Neben dem Reaktionskanal $\gamma p \to p\pi^0$ wurden noch weitere Kanäle ausgewertet. Der prominenteste dieser Kanäle ist $\gamma p \to p\eta$. Dies ist ebenfalls ein neutraler Zerfall mit einem η-Meson im Endzustand. Das η ist ein Meson mit einer Ruhemasse von $(547{,}862 \pm 0{,}017)\,\mathrm{MeV}$ [Pat+16] und ist damit mehr als 4 mal schwerer als das π^0. Die beiden Zerfallskanäle des η mit der höchsten Übergangswahrscheinlichkeit sind $\eta \to \gamma\gamma$ mit $(39{,}41 \pm 0{,}20)\,\%$ und $\eta \to 3\pi^0$ mit $(32{,}68 \pm 0{,}23)\,\%$ [Pat+16]. Im ersten Fall entstehen analog zum Zerfall des π^0 drei Teilchen im Endzustand. Im zweiten Fall entstehen 7 Teilchen, da jedes π^0 wieder hauptsächlich in zwei γ zerfällt. Bei einer Triggerbedingung, welche auf 3 Teilchen im Endzustand schaut, wird auf Grund des höheren Wirkungsquerschnitts hauptsächlich die Reaktion $\gamma p \to p\pi^0 \to p\gamma\gamma$ gespeichert. Es befinden sich aber dennoch beide Zerfallskanäle mit $p\eta$ im Zwischenzustand in den Daten. Es ist auch möglich einen Trigger zu benutzen, welcher 4 Teilchen im Endzustand fordert. Mit diesem werden jedoch keine Ereignisse aus der Reaktion $\gamma p \to p\eta \to p\gamma\gamma$ gespeichert und man kann die Daten dieser Strahlzeiten nicht für die Analyse des Zerfallskanals $\gamma p \to p\pi^0$ benutzen. Deshalb wurde in den meisten Messperioden ein Dreiteilchen-Trigger benutzt. Dadurch konnten diese Strahlzeiten für alle Analysen benutzt werden. Nur Strahlzeiten, durch welche die noch selteneren Zerfallskanäle $\gamma p \to p\eta'$ und $\gamma p \to p\omega$ analysiert werden sollten, benutzen einen Vierteilchen-Trigger. Dies waren die Strahlzeiten ‘‘März 2007’’, ‘‘Januar 2009’’, ‘‘Januar 2011’’, ‘‘Juni 2011’’ und ein Teil von ‘‘November 2011’’.

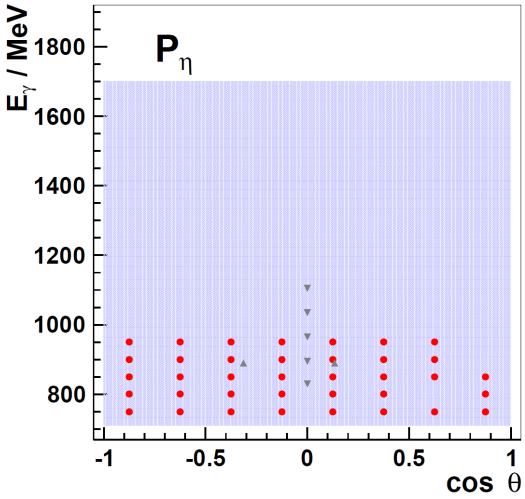
Mit der in dieser Arbeit vorgestellten Datenakquisition wurden in einer Vielzahl von Strahlzeiten Ereignisse der Reaktionen $\gamma p \to p\eta$ gespeichert. Diese Daten wurden ebenfalls im Rahmen von Doktorarbeiten ausgewertet und so die Polarisationsobservablen E, G, H, T und P gemessen [Mül18; Grü16; Har17]. Zusätzlich wurde eine Veröffentlichung zur Publikation eingereicht [M+17] und eine weiter Dissertation befindet sich in Vorbereitung [Afz18b].

Auch mit den Daten, die in den Vierteilchen-Strahlzeiten genommen wurden, konnten Veröffentlichungen zu den Analysen der Kanäle $\gamma p \to p\eta'$ und $\gamma p \to p\omega$ publiziert werden.

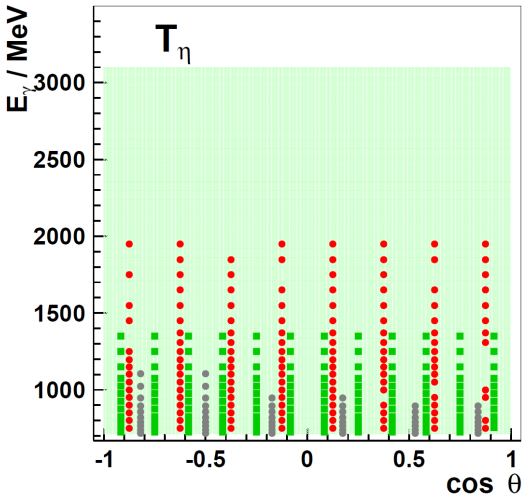
Es gibt ebenfalls Analysen von Kanälen mit zwei Mesonen im Endzustand. Für den Kanal $\gamma p \to p\pi^0\pi^0$ werden eine Veröffentlichung [Sei+18], und zwei Doktorarbeiten [Sei18; Mah18] vorbereitet. Außerdem wurde dieser Kanal in einer Diplomarbeit [Spi13] analysiert. Für den Kanal $\gamma p \to p\pi^0\eta$ ist eine Doktorarbeit in Vorbereitung [Urf18].

Die vollständige Liste der Veröffentlichungen, welche auf Daten basieren, die mit der in dieser Arbeit beschriebenen Datenakquisition gespeichert wurden, findet sich im Anhang A

Durch die Daten, welche mit der im Rahmen dieser Arbeit entwickelten Datenakquisition gewonnen wurden, konnte die Datenbasis der Polarisationsobservablen deutlich vergrößert werden. Die neue Datenbasis nach den Messungen des Crystal-Barrel/TAPS-Experiments ist in den Abbildungen 7.7 und 7.8 gezeigt, zum Vergleich ist auch die frühere Datenbasis aus Abbildung 1.11 mit dargestellt.

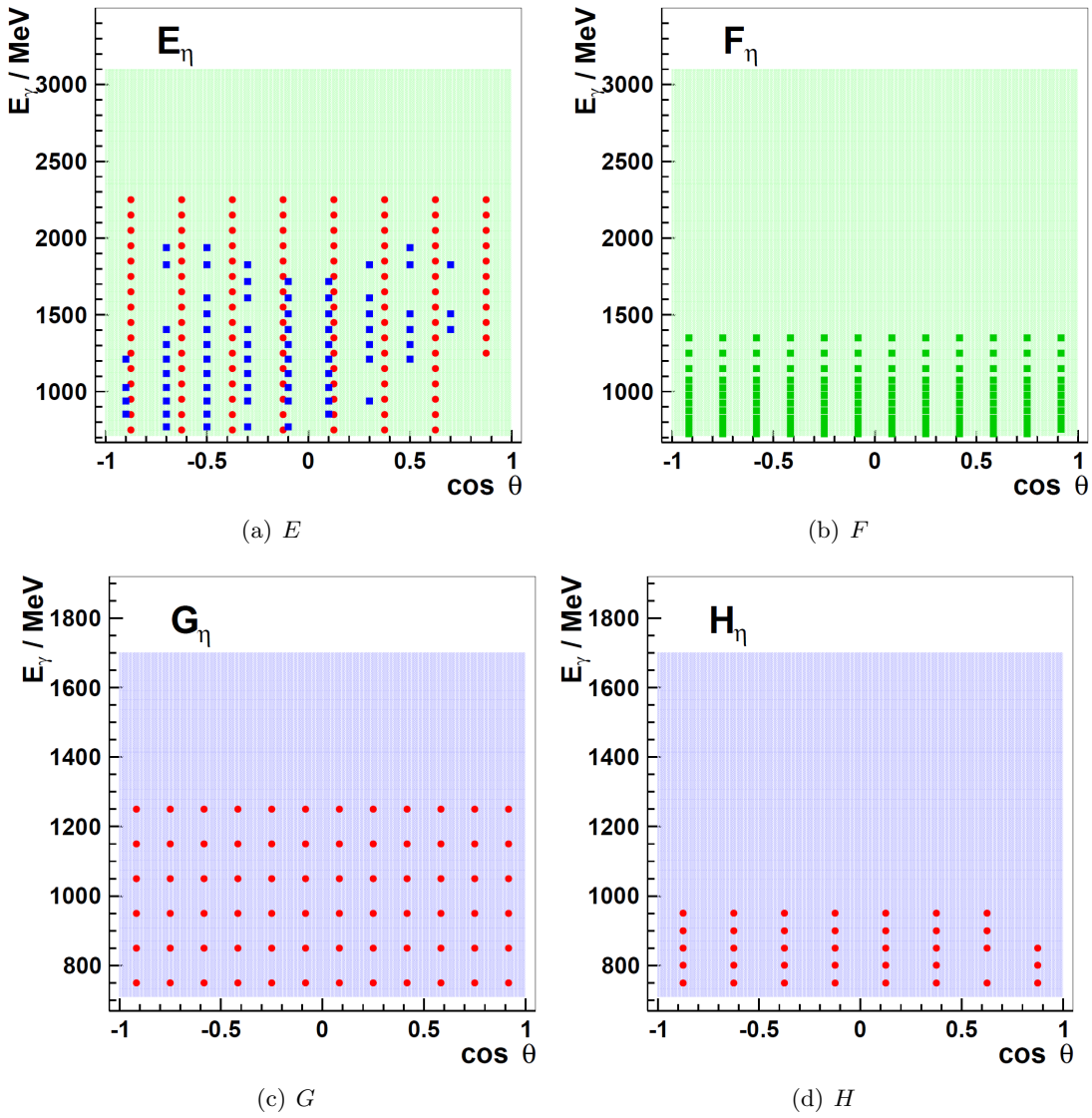


(a) P



(b) T

Abbildung 7.7.: **Observablen der Rückstoßpolarisation P und Targetasymmetrie T im Kanal $\gamma p \to p\eta$.** Gezeigt ist die Datenbasis vor den neuen Messungen mit dem Crystal-Barrel/TAPS-Experiment (grau). Die neuen Messpunkte des Crystal-Barrel/TAPS-Experiments (rot), sowie die während dieser Zeit am MAMI (grün) gemessenen Werte. Bild aus [Afz18a].



(a) E (b) F

(c) G (d) H

Abbildung 7.8.: **Doppelpolarisationsobservablen bei polarisiertem Strahl und polarisiertem Target E, F, G und H im Kanal $\gamma p \to p\eta$.** Gezeigt sind die neuen Messpunkte des Crystal-Barrel/TAPS-Experiments (rot), sowie die während dieser Zeit am MAMI (grün) und am JLAB Beschleuniger [Mec+03] (blau) gemessenen Werte. Bild aus [Afz18a].

In allen gezeigten Polarisationsobservablen für den Kanal $\gamma p \to p\eta$ bis auf F hat sich die Datenbasis durch die mit dem Crystal-Barrel/TAPS-Experiment gemessenen Daten teilweise deutlich vergrößert. Für die Observablen G und H sind die veröffentlichten Daten sogar die einzigen in diesem Kanal.

7.3. Einfluss der Daten auf die Partialwellenanalysen

In den Kapiteln 7.1 und 7.2 wurden die Veröffentlichungen vorgestellt, welche auf den Daten basieren, die mit der in dieser Arbeit vorgestellten Datenakquisition gespeichert wurden. Die Ergebnisse in diesen Veröffentlichungen wurden an mehrere Partialwellenanalysegruppen weitergegeben. Diese Gruppen konnten damit dann ihre Analysen anpassen und dadurch ihre Vorhersagen verbessern. In Abbildung 7.9 ist die dadurch entstehende Verbesserung gezeigt.

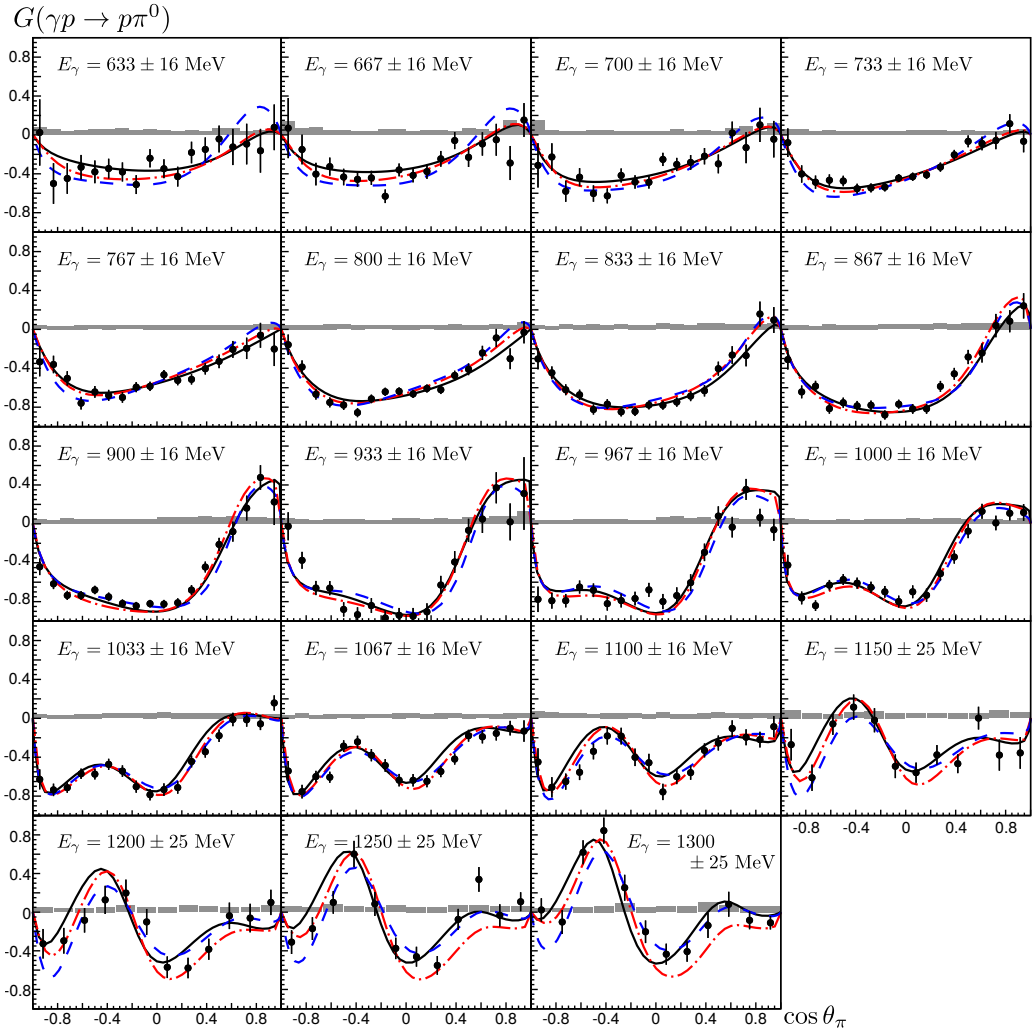
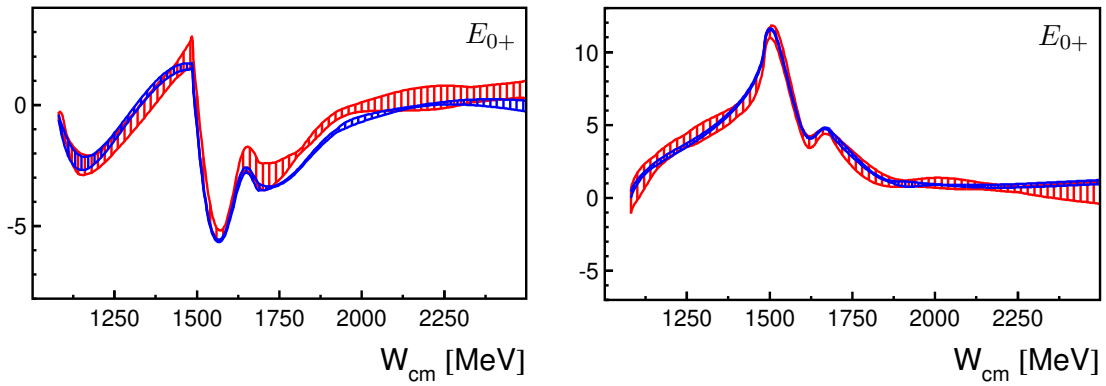


Abbildung 7.9.: **Doppelpolarisationsobservable G in $\gamma p \to p\pi^0$.** Gezeigt sind die mit dem Crystal-Barrel/TAPS-Experiment gemessenen Werte (Punkte) und die Fits an diese Werte. Die Linien sind dabei BnGa 2014-02 (Schwarz), SAID (rot) und JüBo (Blau). Bild aus [Thi+17]

Im Vergleich zur Abbildung 7.4, in der nur die Vorhersagen gezeigt waren, wurden die Partialwellenanalysen hier an die neuen Daten angepasst. Die Messwerte können dadurch deutlich besser beschrieben werden. Die Messungen des Crystal-Barrel/TAPS-Experiments führten somit zu einer Verbesserung der Partialwellenanalysen und damit zu einer Verbesserung des Verständnisses der zu Grunde liegenden Multipole. In [Ani+16] sind die Auswirkungen der neu genommenen Daten auf die Partialwellenanalysen aufgeführt. Dabei zeigt sich sowohl eine Verbesserung der Varianz der Fits innerhalb der einzelnen Partialwellenanalysen, als auch eine Annäherung der Beschreibungen verschiedener Partialwellenanalysegruppen aneinander. Die Gruppe der Bonn-Gatchina Partialwellenanalyse arbeitet eng mit den Gruppen, welche in Bonn die Daten des Crystal-Barrel/TAPS-Experiments analysieren, zusammen. Daher wurde zunächst die Verbesserungen der Bonn-Gatchina PWA durch die neu gemessenen Daten untersucht. Dies ist in Abbildung 7.10 beispielhaft für den Multipol E_{0+} gezeigt.



(a) Realteil (b) Imaginärteil

Abbildung 7.10.: **Verbesserung der BnGa-Partialwellenanalyse durch neue Daten.** Gezeigt ist der Multipol E_{0+} im Kanal $\gamma p \to p\pi^0$. In Rot gezeigt ist der Bereich, der von verschiedenen Fits der BnGa2011-01 and BnGa2011-02 Partialwellenanalysen abgedeckt wird [Ani+12]. In Blau gezeigt ist der Bereich, den diese Fits abdecken, wenn die neu mit dem Crystal-Barrel/TAPS-Experiment genommenen Daten einbezogen werden. Bild modifiziert aus [Har+15].

Dabei wurde von den Fits der Bonn-Gatchina PWA aus der Veröffentlichung [Ani+12] ausgegangen. Die Fehlerbänder wurden aus der (1σ) Ausbreitung von 12 Anpassungen mit unterschiedlichen Annahmen generiert. Die Annahmen sind in [Har+15] beschrieben. Dann wurden zusätzlich für den Kanal $\gamma p \to p\pi^0$ die Ergebnisse für E [Got13; Got+14], G [Thi12; Thi+12] sowie T, P und H [Har17; Har+15] benutzt um die Fits zu verbessern. Aus anderen Kanälen wurden die Daten, aus den als ‘‘Referenz [25]’’ in [Har+15]

aufgeführten Veröffentlichungen, benutzt. Nachdem diese Daten in den Fits der Bonn-Gatchina PWA mit einbezogen wurden, war ein deutlich niedrigerer Unterschied zwischen Anpassungen in der Bonn-Gatchina PWA zu erkennen.

Doch nicht nur die einzelnen Partialwellenanalysen sind durch die neuen Daten verbessert worden. Die Ergebnisse zu denen die verschiedenen PWA-Gruppen kommen, sind durch die neuen Daten auch untereinander vergleichbarer geworden. Für die Partialwellenanalysen Bonn-Gatchina, Jülich-Bonn und SAID ist dies am Beispiel vom Multipol E_{0+} im Kanal $\gamma p \to p\pi^0$ in Abbildung 7.11 gezeigt.

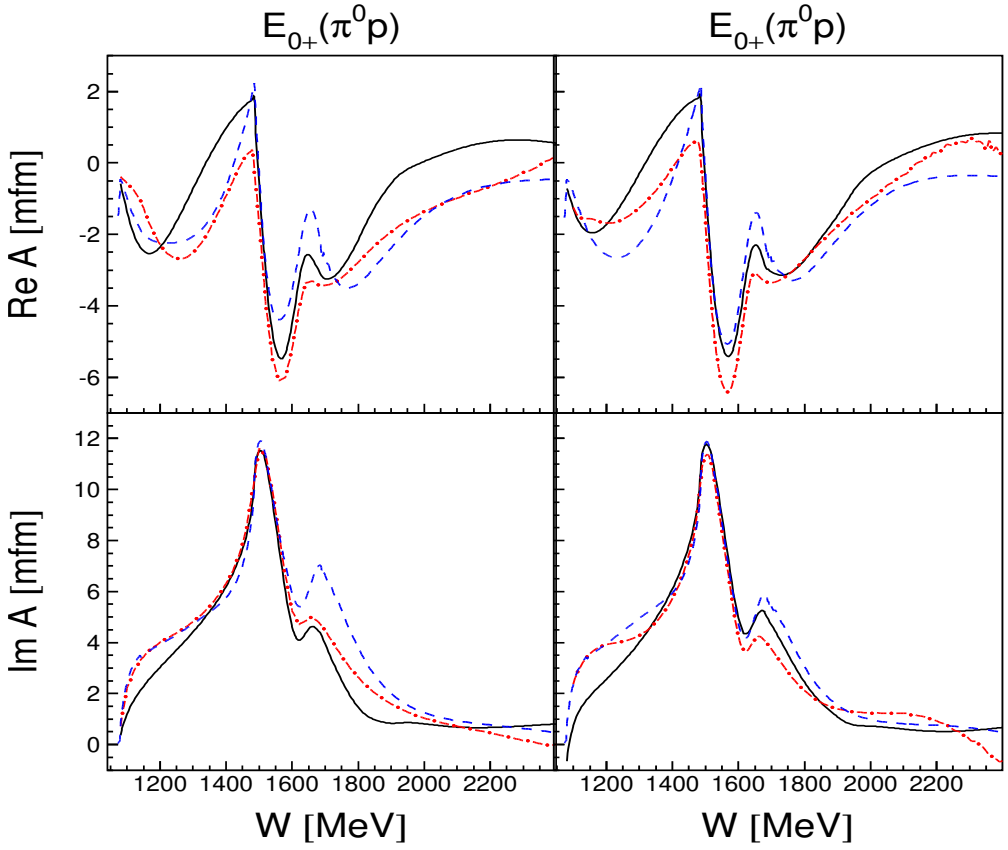


Abbildung 7.11.: Gezeigt ist der Real- und Imaginärteil des Multipols E_{0+} im Kanal $\gamma p \to p\pi^0$ in verschiedenen Partialwellenanalysen: auf der linken Seite ohne die neuen Daten, auf der rechten Seite mit Einbezug der neuen Daten. Gezeigt sind BnGa (Schwarz), JüBo (Blau) und SAID (rot). Bild modifiziert aus [Ani+16]

Die weiteren Multipole bis $\ell = 4$ sind in [Ani+16] aufgeführt. Die Kurven der Partialwellenanalysen für die Multipole liegen dabei nach Einbezug der neuen Daten näher aneinander. Die für die neuen Anpassungen benutzen Datensätze sind in Tabelle 7.2

aufgeführt.

Reaktion	Observable	Ref	BnGa	JüBo	SAID
$\gamma p \to p\pi^0$	$d\sigma/d\Omega$,Σ	[Hor+13]	-	$\star$	$\star$
	T,P,H	[Har+15]	√	√	√
	E	[Got+14; Got+]	√	√	√
	P,C_x,C_z	[Luo+12]	-	-	√
	Σ	[Dug+13]	√	$\star$	$\star$
	G	[Thi+12; Thi+17]	√	√	√
	C_x	[Sik+14]	√	-	√
$\gamma p \to n\pi^+$	Σ	[Dug+13]	√	$\star$	$\star$
$\gamma p \to K^+\Lambda$	$d\sigma/d\Omega$	[Jud+14]	√	-	-
$\gamma p \to K^+\Sigma^0$	$d\sigma/d\Omega$	[Jud+14]	√	-	-

Tabelle 7.2.: Veröffentlichte Datensätze, welche benutzt wurden um die neuen Fits an die Daten zu erstellen sind mit √ markiert. Die mit $\star$ markierten Datensätze waren schon in den älteren Vorhersagen benutzt.

Um die Annäherung der PWAs aneinander zu Beschreiben, wurde die Varianz der einzelnen Analysen zueinander über alle Multipole des Kanals $\gamma p \to p\pi^0$ aufsummiert und dann Energieabhängig aufgetragen.

In Abbildung 7.12 ist dies gezeigt, sowohl für die Anpassungen ohne die neuen Datensätze als auch für die Anpassungen mit den neuen Datensätzen. Dabei wurden wieder die in Tabelle 7.2 aufgeführten Datensätze hinzugenommen. Man erkennt eine deutliche Annäherung bei den meisten Energien. Nur bei sehr niedrigen beziehungsweise hohen Energien kam es zu einer Verschlechterung. Dies zeigt, dass es noch notwendig ist weitere Datensätze zu messen, besonders mit Experimenten die auch diese Energien gut messen können. Da der ELSA-Beschleuniger Elektronen mit einer Energie von bis zu 3,5 GeV an das Experiment extrahieren kann, ist das Crystal-Barrel/TAPS-Experiment dafür gut geeignet.

Nachdem durch die neu gespeicherten Daten die Multipole von den Partialwellenanalysen relativ ähnlich beschrieben werden, müssen nun die Resonanzen extrahiert werden. Hier ist noch zu vergleichen, ob sich die Resonanzen in den Partialwellenanalysen ebenfalls angenähert haben.

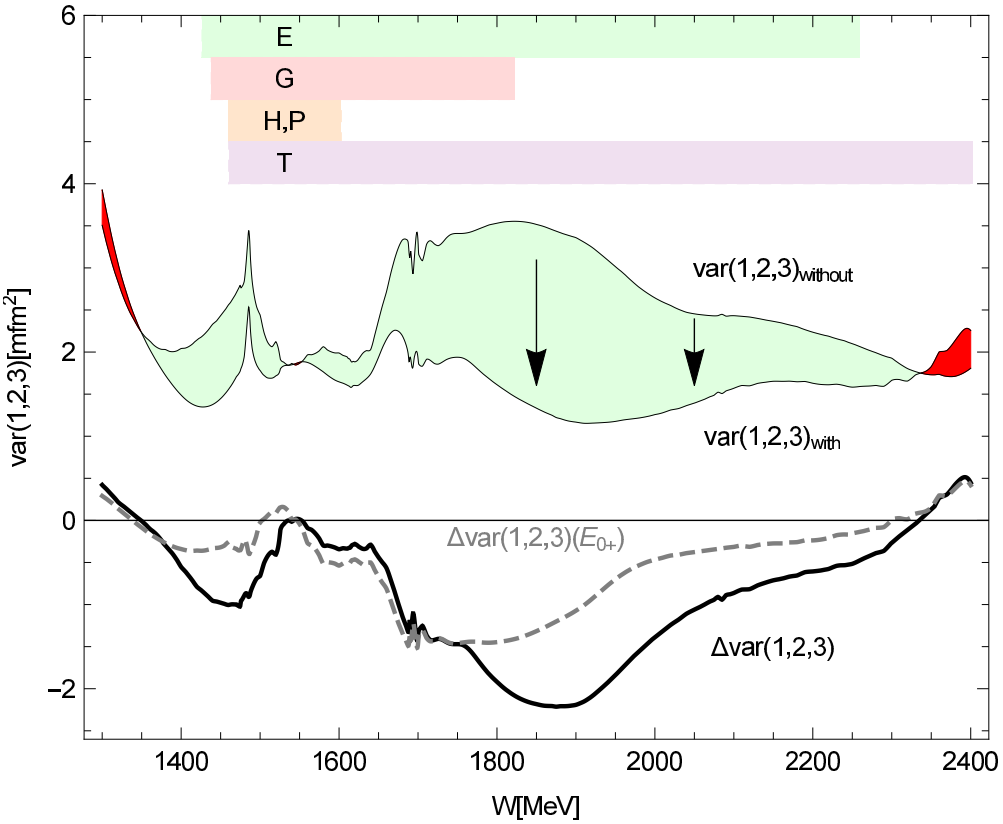


Abbildung 7.12.: **Konvergenz der verschiedenen Partialwellenanalysen, hervorgerufen durch die neuen Daten.** Gezeigt ist die Varianz der Partialwellenanalysen Bonn-Gatchina, Jülich-Bonn und SAID vor und nach dem Einbeziehen der neuen Daten. Die Varianz ist dabei über alle Multipole bis $\ell = 4$ summiert. Der grün schraffierte Bereich stellt eine Verbesserug der Varianz dar, der rot schraffierte eine Verschlechterung. Die Balken am oberen Ende zeigen den Energiebereich, der von den genommenen Daten abgedeckt wird. Bild aus [Ani+16]

8. Zusammenfassung und Ausblick

Analog zur Untersuchung der Anregungszustände der Atome mit der Atomspektroskopie werden mit der Hadronenspektroskopie die Anregungszustände der Hadronen untersucht. Dadurch können zum Beispiel Schlüsse über die Kräfte und die Wechselwirkung zwischen den Quarks gezogen werden. Durch die kurze Lebensdauer überlagern sich im Gegensatz zur Atomspektroskopie die Anregungszustände in der Hadronenspektroskopie, wodurch schwach koppelnde Resonanzen teilweise komplett überdeckt werden. Dies erfordert es, neben dem totalen Wirkungsquerschnitt weitere Observablen zu messen. Besonders wichtig ist es dabei Polarisationsobservablen zu messen, da die zugrundeliegenden Multipole dort nicht nur über ihr Betragsquadrat einfließen, sondern auch in Interferenztermen zwischen Multipolen verschiedener Ordnungen vorkommen. Dadurch kann leichter auf schwach koppelnde Resonanzen zugegriffen werden. Um die Multipole jedoch eindeutig bestimmen zu können, ist es notwendig eine vollständige Datenbasis aus Wirkungsquerschnitten, Polarisationsobservablen und Doppelpolarisationsobservablen aufzubauen. Dabei ist es erforderlich, mindestens 8 geschickt gewählte Observablen zu messen. Die Observablen werden mit Partialwellenanalysen angepasst, um Schlüsse über die zu Grunde liegenden Resonanzen zu ziehen. Diese Resonanzen können dann mit phänomenologischen Modellen und Rechnungen der Gitter-QCD verglichen werden.

Das Crystal-Barrel/TAPS-Experiment am ELSA-Teilchenbeschleuniger ist gut geeignet um diese Messungen durchzuführen, da es photoinduzierte Reaktionen untersucht. Dabei ist es im Gegensatz zu pioninduzierten Reaktion möglich, beide Teilchen im Anfangszustand zu polarisieren und dadurch ist der Zugang zu einer großen Zahl von 16 Observablen gegeben. Um jedoch Messungen mit zirkular-polarisierten Photonen und polarisiertem Target zu ermöglichen, war es notwendig den Experimentplatz des Crystal-Barrel/TAPS-Experiments zu ändern und weitere Detektorkomponenten zum Aufbau hinzuzufügen. Damit die im umgebauten Experiment entstehenden Daten gespeichert werden konnten, wurde im Rahmen dieser Arbeit eine neue Datenakquisition entwickelt, aufgebaut und erfolgreich eingesetzt. Diese basiert auf einem modularen Design mit mehreren, in der Programmiersprache C++ programmierten Komponenten, welche untereinander über das TCP/IP Protokoll kommunizieren. Hauptziel des Konzeptes war es, die durch die Auslese und das Abspeichern der Daten entstehende Totzeit zu minimieren. Zusätzlich zur eigentlichen Datenerfassung wurde im Rahmen dieser Arbeit noch ein

Synchronisationssystem und ein neuer Trigger entwickelt. Das Synchronisationssystem stellt die Konsistenz der Daten der einzelnen Subkomponenten untereinander sicher und der Trigger ermöglicht eine einfache und flexible Konfiguration der Bedingungen, bei denen die Auslese der Daten des Experiments gestartet wird.

Die Leistungsfähigkeit und Zukunftssicherheit des entwickelten Systems wurde ebenfalls im Rahmen dieser Arbeit überprüft. Es wurde dabei untersucht, welche Ereignisraten und welche Ereignisgrößen in den Strahlzeiten des Crystal-Barrel/TAPS-Experiments auftreten können. Zusätzlich wurde untersucht, wie sich die Ereignisraten durch Verbesserungen der benutzen Elektronik im Experiment steigern lassen. Abschließend wurde untersucht, ob die hier vorgestellte Datenakquisition in der Lage ist, diese Ereignisraten und Ereignisgrößen zu speichern. Dabei ergab sich, dass dies möglich ist und auch noch Reserven für andere zukünftige Erweiterungen vorhanden sind.

Dass die Datenakquisition hervorragend für den Einsatz im Crystal-Barrel Experiment geeignet ist, wurde auch in zahlreichen Strahlzeiten gezeigt. Die in diesen Strahlzeiten mit der hier vorgestellten Datenakquisition gespeicherten Daten konnten benutzt werden, um die Kanäle $\gamma p \to p\pi^0$ und $\gamma p \to p\eta$ zu analysieren. Dabei wurden neben dem differentiellen Wirkungsquerschnitt $d\sigma$, der Strahlasymmetrie Σ, der Rückstoßpolarisation P und der Targetasymmetrie T auch die Doppelpolarisationsobservablen E, G und H gemessen. Aus diesen Analysen sind zahlreiche Dissertationen und Veröffentlichungen entstanden.

Zusätzlich sind auch Veröffentlichungen aus Analysen der Kanäle $\gamma p \to p\eta'$ und $\gamma p \to p\omega$ entstanden. Die Kanäle $\gamma p \to p\pi^0\pi^0$ und $\gamma p \to p\pi^0\eta$ werden ebenfalls zur Zeit analysiert.

Mit den Informationen, welche in diesen Analysen gewonnen wurden, konnten die Vorhersagen von Partialwellenanalysen überprüft und verbessert werden.

Die in dieser Arbeit vorgestellte Datenakquisition ist flexibel ausgelegt zum Speichern von Daten in unterschiedlichen Experimenten. Dies wurde im Olympus-Experiment am DESY demonstriert, in dem die hier vorgestellte Datenakquisition benutzt wurde, um den Effekt des Zwei-Photon Austauschs mit elastischer Elektron-Proton und Positron-Proton Streuung zu untersuchen. Auch hier wurde mit Erfolg eine große Zahl an Daten aufgenommen, woraus ebenfalls mehrere Veröffentlichungen entstanden sind.

Damit konnte durch die Daten, welche mit der in dieser Arbeit vorgestellten Datenakquisition gewonnen wurden, das Verständnis der Hadronen- und Kernphysik in unterschiedlichen Feldern verbessert werden.

A. Publizierte Veröffentlichungen

Author	Titel	Referenz
A. Thiel et al.	Well-established nucleon resonances revisited by double-polarization measurements	[Thi+12]
M. Nanova et al.	Determination of the η'-nucleus optical potential	[Nan+13]
R. Milner et al.	The OLYMPUS Experiment	[Mil+14]
J. Hartmann et al.	The $N(1520)3/2^-$ helicity amplitudes from an energy-independent multipole analysis based on new polarization data on photoproduction of neutral pions	[H+14b]
M. Gottschall et al.	First measurement of the helicity asymmetry for $\gamma p \to p\pi^0$ in the resonance region	[Got+14]
S. Friedrich et. al.	Experimental constraints on the ω-nucleus real potential	[Fri+14]
A. Thiel et al.	Three-body nature of N^* and Δ^* resonances from sequential decay chains	[Thi+15]
J. Hartmann et al.	The polarization observables T, P, and H and their impact on $\gamma p \to p\pi^0$ multipoles	[Har+15]
H. Eberhardt et al.	Measurement of double polarisation asymmetries in ω-photoproduction	[Ebe+15]
A. Wilson et al.	Photoproduction of ω mesons off the proton	[Wil+15]
S. Friedrich et al.	Momentum dependence of the imaginary part of the ω- and η'-nucleus optical potential	[Fri+16]
M. Nanova et al.	Determination of the real part of the η'-Nb optical potential	[Nan+16]
A. Thiel et al.	Double-polarization observable G in neutral-pion photoproduction off the proton	[Thi+17]
B. S. Henderson et al.	Hard Two-Photon Contribution to Elastic Lepton-Proton Scattering: Determined by the OLYMPUS Experiment	[Hen+17]
L. Witthauer et al.	Photoproduction of η mesons from the neutron: Cross sections and double polarization observable E	[W+17]
M. Nanova et al.	The η'-carbon potential at low meson momenta	[Nan+18]
J. Müller et al.	New data on $\vec{\gamma}\vec{p} \to \eta p$ with polarized photons and protons and their implications for $N^* \to N\eta$ decays (zur Veröffentlichung eingereicht)	[M+17]

Tabelle A.1.: Publizierte Veröffentlichungen von Analysen von Daten, welche, mit der im Rahmen dieser Arbeit entwickelten Datenakquisition gespeichert wurden und Experimenten, welche die Datenakquisition benutzt haben.

B. Trigger

B.1. Belegung der Eingangskanäle der Triggermodule im Crystal-Barrel/TAPS-Experiment

Eingang	Triggersignal	Beschreibung	Bitmuster
1	Tagger_or1	Tagger Koinzidenz	0x1
2	Tagger_or2	Tagger Oder	0x2
3	Møller		0x4
4	GIM	Oder von allen GIM Kristallen	0x8
5	Innen1	Mindestens 1 Lage gefeuert	0x10
6	Innen2	Mindestens 2 Lagen gefeuert	0x20
7	FPC1	Forward Plug #cluster = 1	0x40
8	FPC2	Forward Plug #cluster > 1	0x80
9	FPV	Forward Plug Veto	0x100
10	Cherenkov		0x200
11	Taps1	Mindestens ein Treffer	0x400
12	Taps2	Taps pulser	0x800
13	Taps3	Mindestens zwei Treffer	0x1000
14	delayed_tagger		0x2000
15	Testpulser	20 MHz clock	0x4000
16	Lightpulser		0x8000

Tabelle B.1.: **Belegung der Eingänge der Triggermodule im Crystal-Barrel/TAPS-Experiment.** Aufgeführt sind die Eingänge der Triggermodule mit ihrer Kurzbezeichnung, einer Beschreibung und dem für eine Aktivierung notwendigen Bitmuster.

B.2. Zeitlicher Ablauf im Trigger

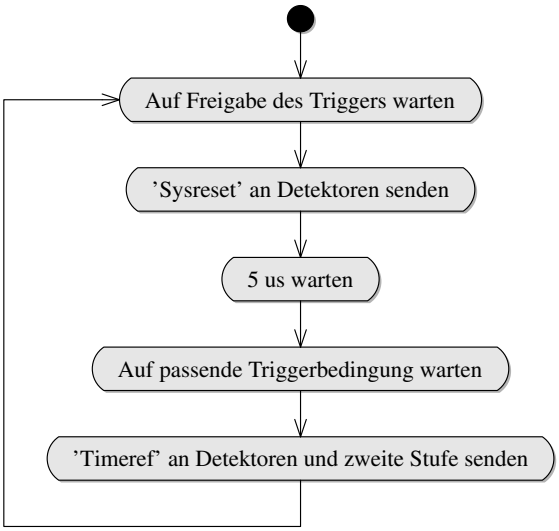


Abbildung B.1.: **Schematischer Ablauf in der ersten Triggerstufe.**

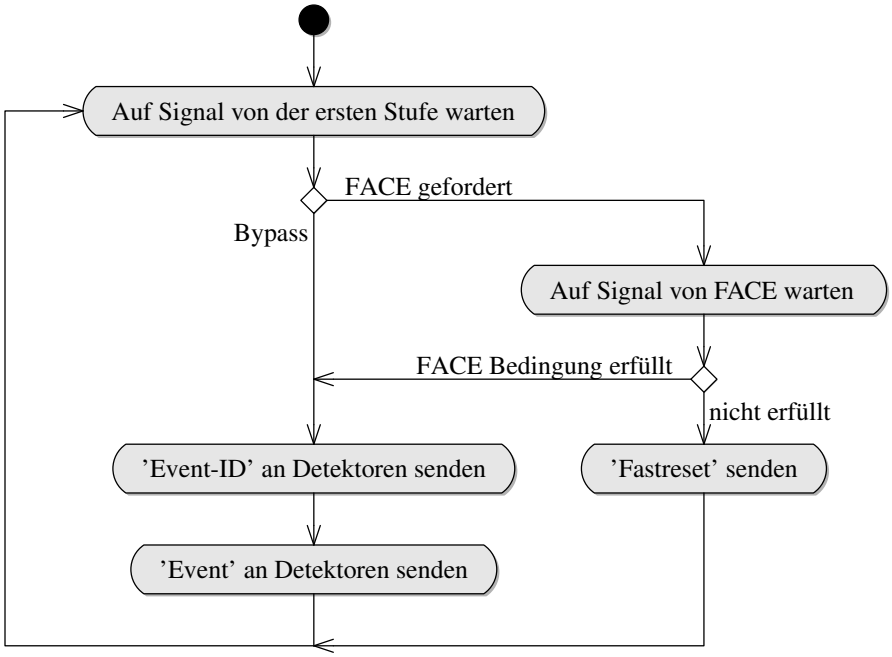


Abbildung B.2.: **Schematischer Ablauf in der zweiten Triggerstufe.**

C. Der VME-Trigger

C.1. Konfiguration der Triggerlogikblöcke

C.1.1. VME-Addressen der Konfigurationswörter

Adresse	31	30	29-24	23 - 16	15-1	0
0x14	not used					consider_spill
0x20,0x28,0x30,...,0x98	triggerpattern				enablepattern	
0x24,0x2C,0x34,...,0x9C	active	clocktrigger	2ndlevel	prescaler		
0x580,0x584,...,0x5BC	spill		not used		eventid	

Tabelle C.1.: **Konfigurationswörter der Triggerlogikblöcke des VME-Triggers.** Die Bedeutung der einzelnen Konfigurationswörter ist in Abschnitt C.1.2 zu finden.

C.1.2. Beschreibung der Konfigurationswörter

- **active:** Dieses Bit aktiviert diesen Logikblock. Wenn dieses Bit nicht gesetzt wird gibt dieser Logikblock kein Ausgangssignal.
- **enablepattern:** Auswahl welcher der Triggereingänge für die Triggerentscheidung betrachtet werden sollen.
- **triggerpattern:** Das Muster, welches die Triggereingänge zeigen sollen, wenn eine Triggerentscheidung gefällt werden soll. Eine 1 gibt dabei vor, das der Eingang ein Signal liefern muß. Eine 0 gibt vor, dass der Eingang kein Signal geben darf (veto).
- **prescaler:** Mit diesem Konfigurationswort wird vorgegeben das nicht jedesmal, wenn das gewünschte Muster erreicht wird, ein Signal ausgegeben wird. Stattdessen wird nur alle X mal ein Signal ausgegeben, wobei X von diesem Konfigurationswort vorgegeben wird. Dies kann ausgenutzt werden um Muster mit hoher Rate zu unterdrücken, so dass nicht nur bei diesem Muster ausgelesen wird.
- **2ndlevel:** Mit diesem Konfigurationswort werden die Anforderungen an die zweite Triggerstufe gesetzt.

- **eventid:** Mit diesem Konfigurationswort werden die vom Logikblock ausgegebene Event-ID gesetzt.
- **consider_spill:** Mit diesem Konfigurationswort wird gesteuert ob das Triggermodul das Spillsignal, welches nur während der Extraktion des Beschleunigers gesetzt ist betrachten soll. Dieses Konfigurationswort wird nur einmal gesetzt und gilt für alle Triggerlogikblöcke.
- **spill:** Hier wird gesteuert ob, wenn consider_spill gesetzt ist, Ausgaben nur erfolgen, wenn das Spill Signal an oder aus ist.
- **clocktrigger:** Wenn dieses Bit gesetzt ist, benutzt dieser Logikblock nicht die Triggereingänge sondern benutzt ein Signal fester Frequenz um das Triggersignal auszulösen. Die Frequenz kann dabei mit dem *prescaler* Konfigurationswort gesteuert werden.
- **scaler_event_prescaler:** Dieses Wort steuert wie oft ein ausgehendes Triggersignal als Scalerevent markiert wird. Wenn dieses Wort auf 0 gesetzt wird werden keine Triggersignale als Scalerevents markiert.

D. Das Sync System

D.1. Adressen des Sync-Clients

Adresse	Name
0x0014	Event
0x0018	0x1 = busy(nur lesen), 0x2 = okay(lesen und schreiben)
0x001C	Lesen der Daten auf dem Sync-Bus
0x0020	0x1 = Trigger, 0x4 = Active
0x0028	Zurücksetzen des Moduls
0x002C	Lösche Zähler
0x0030 - 0x0040	Zähler
0x0050	Setzte Sync-id
0x0054	Event
0x0058	Setze Busy
0x005C	Lösche Busy

Tabelle D.1.: **VME-Adressen des Sync-Client Moduls.**

Adresse	Name	Eingang	Gate
0x0030	Event	Event	Offen
0x0034	Sysreset	Sysreset	Offen
0x0038	Fastreset	Fastreset	Offen
0x003C	Lifetime	Clock	Okay
0x0040	Deadtime	Clock	Busy
0x0044	FreeClock	Clock	Offen

Tabelle D.2.: **Tabelle mit den zur Verfügung stehenden Zählern im Sync-Client Modul.** Aufgeführt sind die VME Adresse unter der die Zähler zu erreichen sind, die benutzen Signale, sowie die Bedingung unter der gezählt wird (Gate).

D.2. Adressen des Sync-Masters

Adresse	Name
0x0014	Okay
0x0018	Busy
0x001C	ECL-Ausgang

Tabelle D.3.: **VME-Adressen des Sync-Master Moduls.**

D.3. Zeitlicher Ablauf im Synchronisationssystem

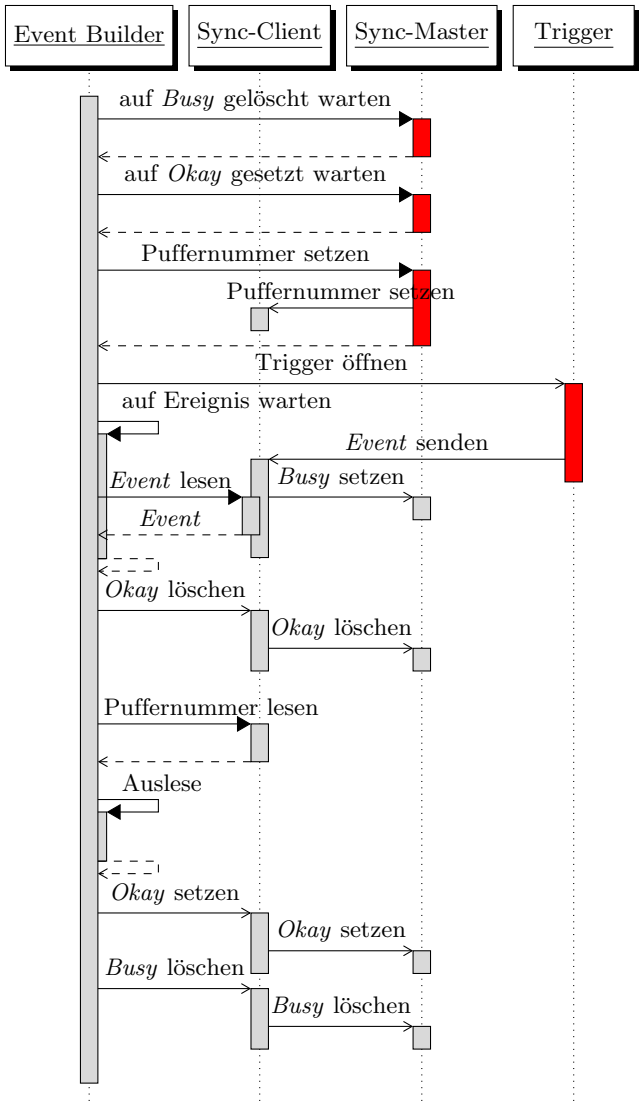


Abbildung D.1.: **Schematische Darstellung des zeitlichen Ablaufs des Synchronisationssystems im Event Builder am Trigger.** Zunächst wartet der für den Trigger zuständige Event Builder darauf, dass alle Subdetektoren das *Busy*-Signal gelöscht und das *Okay*-Signal gesetzt haben. Anschließend gibt er die Puffernummer an den Sync-Master, welcher diese über den Sync-Bus an die Sync-Client Module weitergibt. Nun öffnet der Eventbuilder den Trigger und wartet auf ein Ereignis. Nach einer positiven Triggerentscheidung wird ein Event Signal an den Sync-Client gesendet, wobei sofort von diesem ein *Busy* an den Sync Master signalisiert wird. Nachdem der Eventbuilder das positive Eventsignal gelesen hat, löscht er das *Okay*-Signal und liest die Puffernummer. Nach erfolgter Auslese setzt er das *Okay* Signal und löscht das *Busy*-Signal. Die Unterschiede zu den anderen Event Buildern sind rot markiert.

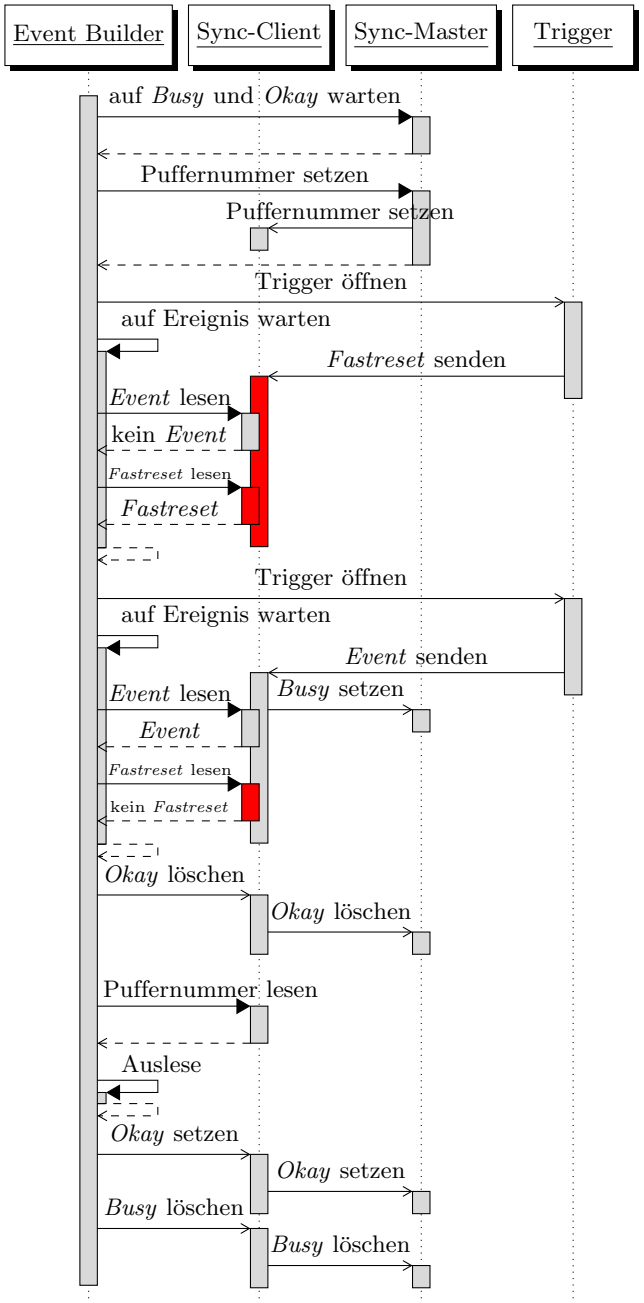


Abbildung D.2.: **Schematische Darstellung des zeitlichen Ablaufs des Synchronisationssystems im Event Builder am Trigger beim zweistufigen Trigger** Bis zum Öffnen des Triggers ist das Verhalten identisch zur einstufigen Version. Beim zweistufigen Trigger kann vom Trigger ein *Fastreset*-Signal gesendet werden. Zusätzlich zum *Event*-Signal liest der Eventbuilder noch das *Fastreset*-Signal. Wenn dieses gesetzt ist wird der Trigger erneut geöffnet. Falls das *Fastreset*-Signal nicht gesetzt ist, erfolgt der Rest identisch zur einstufigen Variante. Die Unterschiede zur einstufigen Variante sind rot markiert.

D.4. Das Kommando BC1 des COMPASS TCS-Systems

0001	TCS-reset	EOS	BOS	*	Catch-reset	Spill-Nr [10:0]
19-16	15	14	13	12	11	10-0

Tabelle D.4.: **TCS-Controller Broadcast Command BC 1.** Schematische Beschreibung des Kommandos BC 1 welches zum Zurücksetzen der COMPASS-TDCs benutzt wird. In diesem Fall ist das Kommando 0x10800.

E. Datenakquisition

E.1. Wichtige Funktionen der Klasse CRawDataBuffer

getDataPtr

Mit dieser Funktion kann vom DataThread oder vom ProcessThread die Position angefordert werden an welche die Daten der nächsten Bank gespeichert werden können. Dabei muss sichergestellt werden, dass nicht über die Grenzen des reservierten Speicherbereichs geschrieben wird (siehe getFreeBufferSize).

getFreeBufferSize

Um sicherzustellen, dass nicht mehr Daten in den Speicher geschrieben werden, als Platz im Speicherbereich reserviert wurde, kann man mit dieser Funktion die Größe des noch freien Speicherbereichs, an der mit getDataPtr angefragten Stelle, abgefragt werden. Die Größe wird dabei in Bytes angegeben.

addData

Diese Funktion wird benutzt um der Klasse CRawDataBuffer mitzuteilen, dass Daten an die mit getDataPtr angefragte Stelle geschrieben wurden, und wichtige zusätzliche Daten wie Bankname, Bankversion und Bankgröße zu speichern.

listBanks

In dieser Funktion die Liste der Namen aller gespeicherten Bänke ausgegeben.

dumpBanks

Diese Funktion gibt die Inhalte aller gespeicherten Bänke aus. Zusätzlich wird für jede Bank der Name und die Größe ausgegeben.

getBankInfo

Mit dieser Funktioon werden die im Buffermanager gespeicherten Daten (Position vom Header1, die Position der Daten, der Name der Bank und ob die Bank übers Netzwerk

gesendet werden soll) abgefragt. Dazu wird der Funktion der Name der Bank übergeben und die Funktion liefert dann die Daten in Form der in Tabelle E.1 gezeigten Struktur zurück.

tBufferManager	
char *	Header
char *	Data
unsigned int	Size
string	Name
bool	send

Tabelle E.1.: **Struktur des tBufferManagers.**

sendData

Diese Funktion wird vom DataThread benutzt um die in der Klasse gespeicherten Daten über das Netzwerk zu übertragen. Dazu muß dieser Funktion das Netzwerkziel, an dass die Daten geschickt werden sollen, übergeben werden.

reset

Mit dieser Funktion wird der Zwischenspeicher geleert. Dabei wird der BufferManager geleert, die Datengröße zurückgesetzt und der Zeiger der mit der Funktion getDataPtr angefordert wird wieder in die Ausgangsposition gesetzt.

Funktionen zum Zugriff auf die Header Daten

Für den Zugriff auf die am Anfang des Speicherbereichs des CRawDatabuffers gespeicherten Daten (Tabelle 5.1) gibt es jeweils eine Funktion welche den gespeicherten Wert zurückliefert. Die Funktionen sind getCPUid(), getEventStatus(), getglobalBufferNumber(), getlocalBufferNumber(), getsoftInterruptCounter(), gethardInterruptCounter(), getLifetime(), getDeadtimePrevEvent(), getEventTime() und getReadoutTime(). Da in dem mit Eventstatus bezeichneten Speicherbereich zusätzlich noch den EventTyp gespeichert ist, gibt es dafür noch eine eigene Funktion getInterruptqualifier().

E.2. Variablen zur Kontrolle des Datenerfassungssystems

Sämtliche Puffergrößen und -zahlen der Datenakquisition können über Variablen angepasst werden. Diese Variablen befinden sich in drei Dateien und sind in den folgenden Tabellen aufgeführt:

Name	Beschreibung	gesetzter Wert
MAXTABLEBANKSIZE	Maximale Größe der TableBank in Bytes (LEVB)	64758
LEVB_BUFFERCOUNT	Anzahl der Puffer in den LEVBs	32
LEVB_BUFFERSIZE	Größe der Puffer in Bytes in den LEVBs	32768

Tabelle E.2.: **Variablen in buffersizes.h**

Name	Beschreibung	Wert
EVS_MAXLEVBSIZE	Max. Größe von einem LEVB empfangener Daten	32768
EVS_MAXTABLEDATASIZE	Max. Größe der TableBank in Bytes (EVS)	8192
EVS_BUFFERNUMBER	Anzahl Puffer für LEVBConnectionHandler	60
EVS_EVENTBUFFERNUMBER	Anzahl Event-Puffer für EventProcessThread	30

Tabelle E.3.: **Variablen in evstypedefs.h**

Name	Beschreibung	gesetzter Wert
MAXDETECTORS	Maximale Anzahl der LEVBs	32

Tabelle E.4.: **Variablen in baselevbids.h**

E.3. Struktur der Zebra Bänke

E.3.1. VELBAE2

size	name	type
unsigned int	trgfired	bitpattern
unsigned int	trgpattern	bitpattern
unsigned int	trgfiredlow	bitpattern
unsigned int	trgpatternlow	bitpattern
unsigned int	trgfiredhigh1	bitpattern
unsigned int	trgfiredhigh2	bitpattern
unsigned int	trgpatternhigh1	bitpattern
unsigned int	trgpatternhigh2	bitpattern
unsigned int	trgstatuswordlevel1	bitpattern
unsigned int	trgstatuswordlevel2	bitpattern
unsigned int	lifetime	counter
unsigned int	deadtime	counter
unsigned int	event	counter
unsigned int	spilltime	counter
unsigned int	fastreset	counter

Tabelle E.5.: **Bankstruktur TRC0**

size	name	type
32 x unsigned int	spill_scaler	counter
32 x unsigned int	lifetime_scaler	counter

Tabelle E.6.: **TRS1**

Kanal	Name
0-13	Nicht Belegt
14	fastreset
15	clock
16	Tagger
17	Tagger-OR
18	Moeller
19	GIM
20	Innen1
21	Innen2
22	FP CF1
23	FP CF2
24	FP Veto
25	Cherenkov
26	Taps1
27	Taps2
28	Taps3
29	Delay Tagger
30	Testpulser
31	Lightpulser

Tabelle E.7.: **Kanalzuordnung für Bank TRS1**

size	name	type
96 x unsigned int	tagger_bar_scaler	Zähler

Tabelle E.8.: **TNS0**

size	name	type
6 x unsigned int	goni_position	Position Tisch 1-6

Tabelle E.9.: **GOC0**

E.3.2. VELBATN

Größe	Name	Typ
variabel x unsigned int	CATCH_TAGGER_BAR	CATCH TDC Wörter

Tabelle E.10.: **Bankstruktur TNT0**

Größe	Name	Typ
1 x unsigned int	Helicity pattern	bitpattern

Tabelle E.11.: **Bankstruktur TNC0**

Größe	Name	Typ
variable x 32bit	CATCH_TAGGER_FIBER_SCALER	CATCH-Zähler

Tabelle E.12.: **Bankstruktur TNS1**

Größe	Name	Typ
12 x unsigned int	Moeller counter 1	Zähler
12 x unsigned int	Moeller counter 2	Zähler

Tabelle E.13.: **Bankstruktur TNS2**

E.3.3. VELBAX1

size	name	type
unsigned int	Lowrange pedestal 1	Bitpattern Pedestal
unsigned int	Lowrange pedestal 2	Bitpattern Pedestal
	...	
unsigned int	Lowrange pedestal 96	Bitpattern Pedestal
unsigned int	Highrange pedestal 1	Bitpattern Pedestal
unsigned int	Highrange pedestal 2	Bitpattern Pedestal
	...	
unsigned int	Highrange pedestal 96	Bitpattern Pedestal

Tabelle E.14.: **Bankstruktur X1TB**

size	name	type
unsigned int	Datenwort 1	Bitpattern ADC
unsigned int	Datenwort 2	Bitpattern ADC
	...	

Tabelle E.15.: **Bankstruktur X1A0**

size	name	type
unsigned short	cluster counter	Zähler
unsigned short	cell counter	Zähler
variable x unsigned short	fifo	FACE Einträge

Tabelle E.16.: **Bankstruktur FAC0**

Bitpattern Pedestal

Index (bit 32-21) + Pedestal (bit 20-11) + Sigma (bit 10-1)

Bitpattern ADC

'01' (Bit 32-31) + Range Bit (Bit 30) + Offen (Bit 29-28)
+ Index (Bit 27-17) + Offen (Bit 16-13) + ADC-Wert (Bit 12-1)

E.3.4. VELBAX2

size	name	type
unsigned int	Lowrange pedestal 1	Bitpattern Pedestal
unsigned int	Lowrange pedestal 2	Bitpattern Pedestal
	...	
unsigned int	Lowrange pedestal 96	Bitpattern Pedestal
unsigned int	Highrange pedestal 1	Bitpattern Pedestal
unsigned int	Highrange pedestal 2	Bitpattern Pedestal
	...	
unsigned int	Highrange pedestal 96	Bitpattern Pedestal

Tabelle E.17.: **Bankstruktur X2TB**

size	name	type
unsigned int	Datenwort 1	Bitpattern ADC
unsigned int	Datenwort 2	Bitpattern ADC
	...	

Tabelle E.18.: **Bankstruktur X2A0**

Bitpattern Pedestal

Index (bit 32-21) + Pedestal (bit 20-11) + Sigma (bit 10-1)

Bitpattern ADC

'01' (Bit 32-31) + Range Bit (Bit 30) + Offen (Bit 29-28)
+ Index (Bit 27-17) + Offen (Bit 16-13) + ADC-Wert (Bit 12-1)

E.3.5. VELBAIN

Größe	Name	Typ
variabel x unsigned int	CATCH_CHAPI_FIBER	CATCH TDC Wörter

Tabelle E.19.: **Bankstruktur INT0**

Kanal	Name
1-513	Kanäle Innendetektor
1500	Paddle
1501	Fast-or Layer 3
1502	Fast-or Layer 2
1503	Fast-or Layer 1
1504	Fast-trig 1/3
1505	Fast-trig min2/3
1506	SUM Layer3
1507	SUM Layer2
1508	SUM Layer1
1509	ChaPI 1/3
1510	ChaPI min2/3
1511	Nicht Belegt
1512	refchannel
1513	Or-PM13
1514-1531	Nicht Belegt
65261	Nicht Belegt

Tabelle E.20.: **Kanalzuordnung für Bank INT0**

E.3.6. VELBAFP

Größe	Name	Typ
variabel x unsigned int	CATCH_FW_VETO	CATCH TDC Wörter
variabel x unsigned int	CATCH_FW_CRY	CATCH TDC Wörter

Tabelle E.21.: **Bankstruktur FPT0**

Größe	Name	Typ
1 x unsigned int	lp_up_control	Kontrollwort Pulserbox-Up
1 x unsigned int	lp_down_control	Kontrollwort Pulserbox-Down
16 x unsigned short	fera_reference_values	ADC Werte Lichtpulser Referenz

Tabelle E.22.: **Bankstruktur LPC0**

Größe	Name	Typ
32 x unsigned int	fw_scaler	Zähler

Tabelle E.23.: **Bankstruktur FPS0**

E.3.7. VELBAGIM

Größe	Name	Typ
32 x unsigned int	GIM_PEDESTAL	Pedestal Werte

Tabelle E.24.: **Bankstruktur GITB**

Größe	Name	Typ
1 x int	event_number	24bit counter
1 x unsigned int	multiplicity	bitpattern
1 x unsigned int	crate	bitpattern
1 x unsigned int	geo	bitpattern
16 x unsigned int	hdata	ADC Werte Highrange
16 x unsigned int	ldata	ADC Werte Lowrange

Tabelle E.25.: **Bankstruktur GIA0**

Größe	Name	Typ
variable x unsigned int	GIM_TDC	CAEN TDC Struktur

Tabelle E.26.: **Bankstruktur GIT0**

Größe	Name	Typ
32 x unsigned int	fw_scaler	Zähler

Tabelle E.27.: **Bankstruktur GIS0**

E.3.8. VELBAMC1-VELBAMC6

Größe	Name	Typ
variable x unsigned int	Minitaps	CAEN NTEC Struktur

Tabelle E.28.: **Bankstruktur M1C0 bis M6C0**

Größe	Name	Typ
336 x unsigned int	mt_scaler	Zähler

Tabelle E.29.: **Bankstruktur M1S0**

E.4. Telnet Kommandos

Kommando	Bedeutung
"INIT"	Start der LEVB-Prozesse
"START"	Starten der Ausleseschleife im Readoutthread
"ABORT"	Abbrechen des Starts
"STOP"	Stop der LEVB-Prozesse
"USER"	USER Kommando (Tab. E.33 und E.34)
"SYNC"	SYNC Kommando (Tab. E.32)
"SCAL"	SCAL Kommando (Tab. E.31)
"COUNTER"	Auslese der Zähler des Sync-Clients
"DATARATE"	Auslese der Datenrate
"DEADTIME"	Auslese der Totzeit zwischen den letzten Ereignissen
"LOGOUT"	Beendet die Telnet Verbindung
"KILLPROG"	Beendet den LEVB
"STATUS"	Gibt den aktuellen Status aus
"RESET"	Führt die Reset Funktion des LEVBs aus
"CONFIG"	Setzt die Konfigurationsdatei
"CMDLOGON"	Telnet Kommandos werden gespeichert
"CMDLOGOFF"	Telnet Kommandos werden nicht gespeichert
"RUNNR"	Setzt die aktuelle Runnr

Tabelle E.30.: Allgemeine Telnet-Kommandos der Lokalen Eventbuilder.

Kommando	Bedeutung
"BUF"	Gibt die aktuelle Software-Buffernummer aus.
"TRIG"	Gibt die aktuelle Software-Triggernummer aus.
"RATE"	Gibt die aktuelle Triggerrate aus.

Tabelle E.31.: Telnet-Kommandos die mit "SCAL " beginnen. Mit diesem Kommando werden verschiedene Zählerstände gelesen.

Kommando	Bedeutung
"SB"	Setzt das *Busy*-Signal
"CB"	Löscht das *Busy*-Signal
"SO"	Setzt das *Okay*
"CO"	Löscht das *Okay*
"CS"	Setzt die Zählerstände zurück
"AC"	Frat ab ob der LEVB aktiviert ist
"STATUS"	Liest das Statusregister
"WRITESTATUS"	Schreibt das Statusregister
"LATCH"	Liest das Latchregister
"WRITELATCH"	Schreibt das Latchregister
"CLEARLATCH"	Setzt das Latchregister zurück
"GETBUS"	Gibt die gerade anliegenden Signale des Sync-Bus aus
"GETLATCHEDBUS"	Gibt die Signale des Sync-Busses beim Latchen aus
"BUFNUM"	Gibt die aktuelle Buffernummer aus

Tabelle E.32.: Telnet-Kommandos die mit "SYNC " beginnen. Mit diesen Kommandos wird das Sync-System gesteuert und überwacht.

Kommando	Bedeutung
"SETCPU "	Aktiviert eine mit Komma getrennte Liste von LEVBs
"OK"	Gibt das bitpattern der *Okay*-Signale aller LEVBs aus
"BUSY"	Gibt das bitpattern der *Busy*-Signale aller LEVBs aus
"TEST"	Öffnet den Trigger für ein Ereignis
"CLEARACTIVE"	Deaktiviert alle LEVBS
"BUFNUM "	Setzt die nächste Buffernummer
"TRIGGER "	Setzt die Konfigurationsdatei des Triggers
"BEAMDIAG"	Gibt einige Zähler zur Strahldiagnose aus
"PAUSE"	Pausiert die Datennahme durch blockieren des Triggers
"UNPAUSE"	Nimmt die Datennahme wieder auf
"SCALER "	Trigger-LEVB Zählerkommandos (Tab. E.35)
"FLASHER "	Lichtpulser-Kommandos (Kap. E.4.1)
"GONI "	Goniometer-Kommandos (Tab. E.36)

Tabelle E.33.: Telnet-Kommandos die mit "USER " beginnen, für den Trigger-LEVB.

Kommando	Bedeutung
"FLASHER "	Lichtpulserkommandos (Kap. E.4.1)

Tabelle E.34.: Telnet-Kommandos die mit "USER " beginnen, für den Forwardplug-LEVB.

Kommando	Bedeutung
‘‘READVIDEOSCAL’’	Gibt die aktuellen Zählständen des Videoscalers aus.
‘‘READNR ’’	Gibt einen der Zählerstände des Triggers aus
‘‘READALL’’	Gibt alle Zählerstände des Triggers aus
‘‘READALLSIS’’	Gibt alle Zählerstände des externen SIS-Zählers aus

Tabelle E.35.: Telnet-Kommandos die mit ‘‘USER SCALER ’’ beginnen, für den Trigger-LEVB.

Kommando	Bedeutung
‘‘ON’’	Legt fest das der nächste Block ein Goniometerrun ist
‘‘OFF’’	Legt fest das der nächste Block kein Goniometerrun ist
‘‘NEXTPOS’’	Fährt die nächste Goniometerposition an
‘‘FILE’’	Legt die Konfigurationsdatei des Goniometers fest
‘‘POSITION’’	Gibt die aktuelle Goniometerposition aus
‘‘PERCENTAGE’’	Gibt den Fortschritt des Goniometerruns aus
‘‘STATUS’’	Gibt den Status des Goniometerruns aus
‘‘WAITFINISHED’’	Wartet bis der Goniometerrun beendet ist
‘‘DIAMOND’’	Legt die nächste Diamantposition fest

Tabelle E.36.: Telnet-Kommandos die mit ‘‘USER GONI ’’ beginnen, für den Trigger-LEVB.

E.4.1. Lichtpulsersteuerung

Die Kommandos des Lichtpulsers befinden sich in zwei lokalen Eventbuildern. Die beiden folgenden Tabellen listen jeweils die Kommandos.

Lichtpulsersteuerbefehle im Trigger-Eventbuilder:

FLASHER ON

Legt fest, dass als nächstes ein Lichtpulserblock gespeichert werden soll.

FLASHER ON

Legt fest, dass als nächstes kein Lichtpulserblock gespeichert werden soll.

FLASHER FILE <Dateiname>

Legt die Konfigurationsdatei für den nächsten Lichtpulserblock fest.

FLASHER PERCENTAGE

Fragt der Fortschritt des gerade laufenden Lichtpulserblocks ab.

FLASHER STATUS

Fragt den Status des laufenden Lichtpuserblocks ab (siehe Tabelle E.37).

FLASHER WAITFINISHED

Wartet bis der Lichtpulserblock beendet ist.

Status	Bedeutung
LP_NOTUSED	Lichtpulser nicht konfiguriert. Lichtpulsersteuerung ausgeschaltet.
LP_READY	Lichtpulserkonfiguration eingelesen. Bereit zum Start der Datenname.
LP_RUNNING	Lichtpulserblock wird gespeichert.
LP_FINISHED	Letzte Filterposition gespeichert. Lichtpulserblock abgeschlossen.

Tabelle E.37.: **Mögliche Stati in der Lichtpulsersteuerung.**

Lichtpulsersteuerbefehle im Forwardplug-Eventbuilder:

FLASHER START <DOWN/UP>

Startet das Blitzen einer der beiden Crystal-Barrel-Hälften.

FLASHER STOP

Stoppt das Blitzen beider Lichtpulserhälften.

FLASHER FILTERUP <Abschwächung/Filterwert>

Selektiert eine Abschwächung oder eine Filterkombination für die Strahlaufwärts-Hälfte.

FLASHER FILTERDOWN <Abschwächung/Filterwert>

Selektiert eine Abschwächung oder eine Filterkombination für die Strahlabwärts-Hälfte.

FLASHER DIVIDER <IN/OUT>

Selektiert ob der Abgriff für den Forwardplug ein oder ausgefahren ist.

FLASHER RESET

Setzt den Lichtpulser zurück.

FLASHER ON

Legt fest dass der nächste Datenblock ein Lichtpulserblock ist.

FLASHER OFF

Legt fest dass der nächste Datenblock kein Lichtpulserblock ist.

FLASHER STATUS

Gibt den Status des Lichtpulsers zurück (Format siehe unten).

Beispiel für die Ausgabe des Lichtpulserstatus vom Forwardplug-Eventbuilder:

''Current Downstream Filterword: 0x1 Current transmission: 0.703''

''Current Upstream Filterword: 0x2 Current transmission: 0.477''

''Current Divider position is: IN''

''Flashlampstatus is: ON''

Abbildungsverzeichnis

Literatur

[A+] A. V. Anisovich, E. Klempt, V. A. Nikonov u. a., *Bonn-Gatchina Partial Wave Analysis*, [Abruf am 08-Oktober-2018], URL: https://pwa.hiskp.uni-bonn.de/.

[Afz18a] F. Afzal, Persönliche Kommunikation, 2018.

[Afz18b] F. Afzal, *Measurement of the beam and helicity asymmetry in the reactions $\gamma p \to p\pi^0$ and $\gamma p \to p\eta$ with the CBELSA/TAPS experiment at ELSA and with the A2 experiment at MAMI*, (in Vorbereitung), Dissertation: Helmholtz-Institut für Strahlen- und Kernphysik, 2018.

[Ani+12] A. V. Anisovich u. a., *Properties of baryon resonances from a multichannel partial wave analysis*, Eur. Phys. J. **A48** (2012) 15, arXiv: 1112.4937 [hep-ph].

[Ani+16] A. V. Anisovich u. a., *The impact of new polarization data from Bonn, Mainz and Jefferson Laboratory on $\gamma p \to \pi N$ multipoles*, Eur. Phys. J. **A52**.9 (2016) 284, arXiv: 1604.05704 [nucl-th].

[ANSI97] *American National Standard for The Mezzanine Concept M-Module Specification*, ANSI/VITA 12-1996, Mai 1997.

[B+99] Ch. Bradtke, H. Dutz, H. Peschel u. a., *A New Frozen-Spin Target for 4π Particle Detection*, Nuclear Instruments and Methods **A**.436 (1999) 430–442.

[Bar04] O. Bartholomy, *Photoproduktion einzelner Mesonen am Proton bei CB-ELSA: Untersuchung der Reaktionen $\gamma p \to p\eta$, $\gamma p \to p\pi^0$ und $\gamma p \to p\eta'$ bei Photonenenergien zwischen 0.3 und 3 GeV*, Dissertation: Helmholtz-Institut für Strahlen- und Kernphysik, 2004.

[BDS75] I. S. Barker, A. Donnachie und J. K. Storrow, *Complete Experiments in Pseudoscalar Photoproduction*, Nucl. Phys. **B95** (1975) 347–356.

[Ber+12] J. Beringer u. a., *Review of Particle Physics (RPP)*, Phys. Rev. **D86** (2012) 010001.

[Blo72] W. R. Blood, *MECL System Design Handbook*, 2. Aufl., Motorola Semiconductor Products Inc., 1972.

[Blu+86] E. Blucher u. a., *Tests of caesium iodide crystals for an electromagnetic calorimeter*, Nuclear Intruments and Methods **A**.249 (1986) 201–227.

[Bös06] S. Böse, *Modifikation und Test des Lichtpulsersystems für den Crystal Barrel Aufbau an ELSA*, Diplomarbeit: Helmholtz-Institut für Strahlen- und Kernphysik, 2006.

[Bri+] W. J. Briscoe u. a., *George Washington University Partial Wave Analysis*, [Abruf am 08-Oktober-2018], URL: http://gwdac.phys.gwu.edu/.

[CAE04] CAEN, *V874B Technical Information Manual*, http://www.caen.it, CAEN, Nov. 2004.

[Che+57] G. F. Chew u. a., *Relativistic dispersion relation approach to photomeson production*, Phys. Rev. **106** (1957) 1345–1355.

[CM97] D. G. Crabb und W. Meyer, *Solid polarized targets for nuclear and particle physics experiments*, Ann. Rev. Nucl. Part. Sci. **47** (1997) 67–109.

[COM+07] P. A. COMPASS Collaboration u. a., *The COMPASS Experiment at CERN*, Nuclear Instruments and Methods **A**.577 (2007) 455–518.

[CT97] W.-T. Chiang und F. Tabakin, *Completeness rules for spin observables in pseudoscalar meson photoproduction*, Phys. Rev. **C55** (1997) 2054–2066, arXiv: nucl-th/9611053 [nucl-th].

[Deu05] A. Deur, ‘‘The Strong coupling constant at low Q**2’’, *Quark-hadron duality and the transition to pQCD. Proceedings, 1st Workshop, Frascati, Italy, June 6-8, 2005*, 2005 51–56, arXiv: hep-ph/0509188 [hep-ph], URL: http://www1.jlab.org/Ul/publications/view_pub.cfm?pub_id=6363.

[Die08] J. Dielmann, *Entwicklung, Aufbau und Test eines Detektors zur Bestimmung des Photonenflusses an der Bonner Photonenmarkierungsanlage*, Diplomarbeit: Helmholtz-Institut für Strahlen- und Kernphysik, 2008.

[Dre+99] D. Drechsel u. a., *A Unitary isobar model for pion photoproduction and electroproduction on the proton up to 1-GeV*, Nucl. Phys. **A645** (1999) 145–174, arXiv: nucl-th/9807001 [nucl-th].

[Dug+13] M. Dugger u. a., *Beam asymmetry Σ for π^+ and π^0 photoproduction on the proton for photon energies from 1.102 to 1.862 GeV*, Phys. Rev. C **88** (6 Dez. 2013) 065203.

[Ebe+15] H. Eberhardt u. a., *Measurement of double polarisation asymmetries in ω-photoproduction*, Phys. Lett. B **750** (2015) 453–458, arXiv: 1504.02221 [nucl-ex].

[Ebe06] H. Eberhardt, *Messung der Targetpolarisation und Detektorstudie für das Møllerpolarimeter des Crystal-Barrel-Aufbaus an ELSA*, Diplomarbeit: Physikalisches Institut, 2006.

[Edw+11] R. G. Edwards u. a., *Excited state baryon spectroscopy from lattice QCD*, Phys. Rev. **D84** (2011) 074508, arXiv: 1104.5152 [hep-ph].

[Fle01] H. Flemming, *Entwurf und Aufbau eines Zellularlogik-Triggers für das Crystal-Barrel-Experiment an der Elektronenbeschleunigeranlage ELSA*, Dissertation: Institut für Experimentalphysik I, 2001.

[For09] K. Fornet-Ponse, *Tagging-System, Absolutnormierung des Photonenflusses und Bestimmung des xy-WQ für das CBTAPS-Experiment an ELSA*, Diss.: Physikalisches Institut, 2009.

[Fri+14] S. Friedrich u. a., *Experimental constraints on the ω-nucleus real potential*, Phys. Lett. B **736** (2014) 26–32, arXiv: 1407.0899 [nucl-ex].

[Fri+16] S. Friedrich u. a., *Momentum dependence of the imaginary part of the ω- and η'-nucleus optical potential*, Eur. Phys. J. A **52**.9 (2016) 297, arXiv: 1608.06074 [nucl-ex].

[Fro16] F. Frommberger, *ELSA Internetseite*, www-elsa.physik.uni-bonn.de, 2016.

[Fun08] Ch. Funke, *Analyse der Triggerfähigkeiten zur Selektion hadronischer Ereignisse und Entwicklung eines Hochgeschwindigkeits-Triggers für den Vorwärtskonus des Crystal-Barrel-Detektors*, Dissertation: Helmholtz-Institut für Strahlen- und Kernphysik, 2008.

[Fun13] Ch. Funke, *CBDF linear file format*, CB Internal Note (2013).

[Got+] M. Gottschall u. a., "Double-polarization observable E in neutral-pion photoproduction", Publikation in Vorbereitung.

[Got+14] M. Gottschall u. a., *First measurement of the helicity asymmetry for $\gamma p \to p\pi^0$ in the resonance region*, Phys. Rev. Lett. **112** (2014) 012003, arXiv: 1312.2187 [nucl-ex].

[Got13] M. Gottschall, *Bestimmung der Doppelpolarisationsobservablen E für die Reaktion $\gamma p \to p\pi^0$ am CBELSA/TAPS-Experiment*, Dissertation: Helmholtz-Institut für Strahlen- und Kernphysik, 2013.

[Gru06] B. Grube, *A Trigger Control System for COMPASS and A Measurement of the Transverse Polarization of Δ and Σ Hyperons from Quasi-Real Photo-Production*, Dissertation: Physik-Department, 2006.

[Grü06] M. Grüner, *Modifikation und Test des Innendetektors für das Crystal Barrel Experiment*, Diplomarbeit: Helmholtz-Institut für Strahlen- und Kernphysik, 2006.

[Grü16] M. Grüner, *Messung der Doppelpolarisationsobservable G in der Reaktion $\gamma p \to p\eta$ mit dem Crystal-Barrel/TAPS-Experiment an ELSA*, Dissertation: Helmholtz-Institut für Strahlen- und Kernphysik, 2016.

[Grü18] M. Grüner, Persönliche Kommunikation, 2018.

[H+14a] J. Hartmann, H. Dutz, A. V. Anisovich u. a., *The $N(1520)3/2^-$ helicity amplitudes from an energy-independent multipole analysis based on new polarization data on photoproduction of neutral pions*, Phys. Rev. Lett. **113** (2014) 062001, arXiv: 1407.2163 [nucl-ex].

[H+14b] J. Hartmann, H. Dutz, A. V. Anisovich u. a., *The $N(1520)3/2^-$ helicity amplitudes from an energy-independent multipole analysis based on new polarization data on photoproduction of neutral pions*, Phys. Rev. Lett. **113** (2014) 062001, arXiv: 1407.2163 [nucl-ex].

[Har+15] J. Hartmann u. a., *The polarization observables T , P , and H and their impact on $\gamma p \to p\pi^0$ multipoles*, Phys. Lett. **B748** (2015) 212–220, arXiv: 1506.06226 [nucl-ex].

[Har17] J. Hartmann, *Measurement of Double Polarization Observables in the Reactions $\gamma p \to p\pi^0$ and $\gamma p \to p\eta$ with the Crystal Barrel/TAPS Experiment at ELSA*, Dissertation: Helmholtz-Institut für Strahlen- und Kernphysik, 2017.

[Hei+01] F. H. Heinsius u. a., *Implementation of the dead-time free F1 TDC in the COMPASS detector readout*, Nuclear Instruments and Methods **A**.461 (2001) 507–510.

[Hen+17] B. S. Henderson u. a., *Hard Two-Photon Contribution to Elastic Lepton-Proton Scattering: Determined by the OLYMPUS Experiment*, Phys. Rev. Lett. **118**.9 (2017) 092501, arXiv: 1611.04685 [nucl-ex].

[Hil+13] M. Hilt u. a., *Pion photo- and electroproduction in relativistic baryon chiral perturbation theory and the chiral MAID interface*, Phys. Rev. **C88** (2013) 055207, arXiv: 1309.3385 [nucl-th].

[Hil06] W. Hillert, *The Bonn Electron Stretcher Accelerator ELSA: Past and future*, European Physical Journal **A**.28 (2006) 139–148, ISSN: 1434-6001.

[Hor+13] D. Hornidge u. a., *Accurate Test of Chiral Dynamics in the* $\vec{\gamma}\, p \to \pi^0 p$ *Reaction*, Phys. Rev. Lett. **111** (6 Aug. 2013) 062004.

[IK78] N. Isgur und G. Karl, *P Wave Baryons in the Quark Model*, Phys. Rev. **D18** (1978) 4187.

[Jud+14] T. C. Jude u. a., $K^+\Lambda$ *and* $K^+\Sigma^0$ *photoproduction with fine center-of-mass energy resolution*, Phys. Lett. **B735** (2014) 112–118, arXiv: 1308.5659 [nucl-ex].

[Kai07] D. Kaiser, *Aufbau und Test des Gas-Çerenkov-Detektors für den Crystal-Barrel-Aufbau an ELSA*, Diplomarbeit: Helmholtz-Institut für Strahlen- und Kernphysik, 2007.

[Kam10] S. Kammer, *Zirkular und linear polarisierte Photonen am CBELSA-Taps Experiment*, Dissertation: Physikalisches Institut, 2010.

[Kla12] P. Klassen, *Integration des MiniTAPS-Detektors in die Crystal Barrel Datenerfassung*, Diplomarbeit: Helmholtz-Institut für Strahlen- und Kernphysik, 2012.

[Kru+95] B. Krusche u. a., *New threshold photoproduction of* η *mesons off the proton*, Physical Review Letters 74 (1995) 3736.

[Leu01] R. Leukel, *Photoproduktion neutraler Pionen am Proton mit linear polarisierten Photonen im Bereich der* $\Delta(1232)$*-Resonanz*, Dissertation, 2001.

[Lör+01] U. Löring u. a., *Relativistic quark models of baryons with instantaneous forces: Theoretical background*, Eur. Phys. J. **A10** (2001) 309–346, arXiv: hep-ph/0103287 [hep-ph].

[Luo+12] W. Luo u. a., *Polarization Components in π^0 Photoproduction at Photon Energies up to 5.6 GeV*, Phys. Rev. Lett. **108** (22 Mai 2012) 222004.

[M+17] J. Müller, J. Hartmann, M. Grüner u. a., *New data on $\vec{\gamma}\vec{p} \to \eta p$ with polarized photons and protons and their implications for $N^* \to N\eta$ decays*, submitted to Phys. Lett. B, 2017.

[Mah18] Ph. Mahlberg, "Measurement of beam asymmetries in the reaction $\gamma p \to p\pi^0\pi^0$ with the Crystal Barrel/TAPS experiment at ELSA", (Dissertation in Vorbereitung), Universität Bonn, 2018.

[Mak11] K. Makonyi, *Setup of the MINITAPS Detector for the CBELSA/TAPS Experiment*, Dissertation: 2. Physikalisches Institut, 2011.

[Mec+03] B. A. Mecking u. a., *The CEBAF Large Acceptance Spectrometer (CLAS)*, Nucl. Instrum. Meth. **A503** (2003) 513–553.

[Mil+14] R. Milner u. a., *The OLYMPUS Experiment*, Nucl. Instrum. Meth. **A741** (2014) 1–17, arXiv: 1312.1730 [physics.ins-det].

[Mül18] J. Müller, *Bestimmung der Doppelpolarisationsobservablen E in der Reaktion $\vec{\gamma}\vec{p} \to p\eta$ am CBELSA/TAPS-Experiment in Bonn*, Dissertation: Helmholtz-Institut für Strahlen- und Kernphysik, 2018.

[Nak+10] K. Nakamura u. a., *Review of Particle Physics*, Journal of Physics **G 37** (2010) 075021.

[Nan+13] M. Nanova u. a., *Determination of the η'-nucleus optical potential*, Phys. Lett. B **727** (2013) 417–423, arXiv: 1311.0122 [nucl-ex].

[Nan+16] M. Nanova u. a., *Determination of the real part of the η'-Nb optical potential*, Phys. Rev. C **94**.2 (2016) 025205, arXiv: 1607.07228 [nucl-ex].

[Nan+18] M. Nanova u. a., *The η'-carbon potential at low meson momenta* (2018), arXiv: 1810.01288 [nucl-ex].

[Oli+14] K. A. Olive u. a., *Review of Particle Physics*, Chin. Phys. **C38** (2014) 090001.

[Ome81] A. S. Omelaenko, *AMBIGUITIES OF MULTIPOLE ANALYSIS OF THE PHOTOPRODUCTION OF NEUTRAL PIONS ON A NUCLEON. (IN RUSSIAN)*, Yad. Fiz. **34** (1981) 730–740.

[Pat+16] C. Patrignani u. a., *Review of Particle Physics*, Chin. Phys. **C40**.10 (2016) 100001.

[Pee+07] H. van Pee u. a., *Photoproduction of π^0-mesons off protons from the $\Delta(1232)$ region to $E_\gamma = 3GeV$*, European Physical Journal **A**.31 (2007) 61.

[Pov+06] B. Povh u. a., *Teilchen und Kerne. Eine Einführung in die physikalischen Konzepte*, 7. Auflage, Springer-Verlag GmbH, 2006, ISBN: 3540366857.

[Rön+] D. Rönchen u. a., *The Jülich-Bonn dynamical coupled-channel analysis of pion- and photon-induced hadronic reactions*, [Abruf am 08-Oktober-2018], URL: http://collaborations.fz-juelich.de/ikp/meson-baryon/juelich_amplitudes.html.

[Rön+14] D. Rönchen u. a., *Photocouplings at the Pole from Pion Photoproduction*, Eur. Phys. J. **A50**.6 (2014) 101, [Erratum: Eur. Phys. J.A51,no.5,63(2015)], arXiv: 1401.0634 [nucl-th].

[San+09] A. M. Sandorfi u. a., *Calculations of Polarization Observables in Pseudoscalar Meson Photo-production Reactions* (2009), arXiv: 0912.3505 [nucl-th].

[San+11] A. M. Sandorfi u. a., *Determining pseudoscalar meson photo-production amplitudes from complete experiments*, J. Phys. **G38** (2011) 053001, arXiv: 1010.4555 [nucl-th].

[Sch+94] W. J. Schwille u. a., *Design and construction of the SAPHIR detector*, Nuclear Instruments and Methods **A**.344 (1994) 470–486.

[Sch04] Ch. Schmidt, *Entwicklung eines neuen Datenakquisitionssystems für das CB-ELSA-Experiment*, Dissertation: Helmholtz-Institut für Strahlen- und Kernphysik, 2004.

[Sei+18] T. Seifen u. a., "Polarisation observables in double neutral pion phootproduction", Publikation in Vorbereitung, 2018.

[Sei18] T. Seifen, ''Messung von Polarisationsobservablen in der $2\pi^0$ Photoproduktion mit dem Crystal-Barrel/TAPS-Experiment'', (Dissertation in Vorbereitung), Universität Bonn, 2018.

[She11] Sheevar, *Hg-Niederdrucklampe mit Spektrum --- Wikipedia, The Free Encyclopedia*, [Abruf am 08-Oktober-2018; Teile des Bildes entfernt und skaliert], 2011, URL: https://de.wikipedia.org/wiki/Datei:Hg-Spektrum.jpg.

[Sik+14] M. H. Sikora u. a., *Measurement of the $^1H(\vec{\gamma}, \vec{p})\pi^0$ reaction using a novel nucleon spin polarimeter*, Phys. Rev. Lett. **112**.2 (2014) 022501, arXiv: 1309.7897 [nucl-ex].

[Spi13] K. Spieker, *Determination of the polarization observables Σ and G in the reaction $\gamma p \to p\pi^0\pi^0$*, Diplomarbeit: Helmholtz-Institut für Strahlen- und Kernphysik, 2013.

[Thi+12] A. Thiel u. a., *Well-established nucleon resonances revisited by double-polarization measurements*, Phys. Rev. Lett. **109** (2012) 102001, arXiv: 1207.2686 [nucl-ex].

[Thi+15] A. Thiel u. a., *Three-body nature of N^* and Δ^* resonances from sequential decay chains*, Phys. Rev. Lett. **114**.9 (2015) 091803, arXiv: 1501.02094 [nucl-ex].

[Thi+17] A. Thiel u. a., *Double-polarization observable G in neutral-pion photoproduction off the proton*, Eur. Phys. J. **A53**.1 (2017) 8, arXiv: 1604.02922 [nucl-ex].

[Thi12] A. Thiel, *Bestimmung der Doppelpolarisationsobservablen G in π^0-Photoproduktion*, Dissertation: Helmholtz-Institut für Strahlen- und Kernphysik, 2012.

[Tia+] L. Tiator u. a., *MAID Partial Wave Analysis*, [Abruf am 08-Oktober-2018], URL: https://maid.kph.uni-mainz.de/.

[Urf18] G. Urff, ''Measurement of beam asymmetries in the reaction $\gamma p \to p\pi^0\eta$ with the Crystal Barrel/TAPS experiment at ELSA'', (Dissertation in Vorbereitung), Universität Bonn, 2018.

[W+17] L. Witthauer, M. Dieterle u. a., *Photoproduction of η mesons from the neutron: Cross sections and double polarization observable E*, Eur. Phys. J. A **53**.1 (2017) 58.

[Wal18] D. Walther, Persönliche Kommunikation, 2018.

[WBT14] Y. Wunderlich, R. Beck und L. Tiator,
The complete-experiment problem of photoproduction of pseudoscalar mesons in a truncated partial-wave analysis,
Phys. Rev. **C89**.5 (2014) 055203.

[Wil+15] A. Wilson u. a., *Photoproduction of ω mesons off the proton*,
Phys. Lett. **B749** (2015) 407–413, arXiv: 1508.01483 [nucl-ex].

[Wil74] K. G. Wilson, *Confinement of Quarks*,
Phys. Rev. **D10** (1974) 2445–2459, [,319(1974)].

[Win06] A. Winnebeck,
Entwicklung und Implementierung eines universellen, FPGA basierten Triggermoduls für das Crystal-Barrel-Experiment an ELSA,
Diplomarbeit: Helmholtz-Institut für Strahlen- und Kernphysik, 2006.

[Wor+12] R. L. Workman u. a.,
Unified Chew-Mandelstam SAID analysis of pion photoproduction data,
Phys. Rev. **C86** (2012) 015202, arXiv: 1202.0845 [hep-ph].

[Wun+17] Y. Wunderlich u. a., *Determining the dominant partial wave contributions from angular distributions of single- and double-polarization observables in pseudoscalar meson photoproduction*, Eur. Phys. J. **A53**.5 (2017) 86,
arXiv: 1611.01031 [physics.data-an].

[Wun12] Y. Wunderlich,
Studies on a complete experiment for pseudoscalar meson photoproduction,
Diplomarbeit: Helmholtz-Institut für Strahlen- und Kernphysik, 2012.

Danksagung

Zunächst einmal möchte ich ganz herzlich meinem Doktorvater Prof. Dr. Reinhard Beck danken. Nur durch ihn war es mir möglich mich mit dem interessanten Thema der Datenakquision zu beschäftigen. Er hat es mir ermöglicht, an zwei Experimenten viele Erfahrungen damit zu sammeln. Auch möchte ich Prof. Dr. Kai Brinkmann für die Übernahme des Zweitgutachtens danken.

Prof. Dr. Ulf-G. Meißner und Prof. Dr. Rainer Manthey danke ich für die Bereitschaft zur Teilnahme an meiner Promotionskommission.

Meiner Familie möchte ich auch meinen Dank aussprechen, da sie es mir überhaupt erst ermöglicht haben ein Studium zu beginnen und mich auch schon in meiner Kindheit für die Physik begeistern konnten.

Meinen besonderen Dank möchte ich meiner Verlobten Nadja aussprechen, die mir immer emotional beigestanden hat und mich in allen Lagen unterstützte. Ohne sie wäre diese Arbeit so nicht möglich gewesen.

Ich danke auch meinen Arbeitskollegen aus den Gruppen Beck, Thoma und Thiel, mit denen ich immer harmonisch zusammenarbeiten konnte. Besonders danken möchte ich hier Christian Funke und Alexander Winnebeck, mit denen ich besonders eng und harmonisch zusammengearbeitet haben, unter anderem auch im Olympus Experiment. Ebenfalls danken möchte ich Jun. Prof. Annika Thiel, Christian Honisch und Yannick Wunderlich für die Unterstützung bei wissenschaftlichen Fragen.

Großer Dank gebührt auch Martin Urban, der mich immer sowohl bei der Arbeit, als auch neben der Arbeit unterstützt hat.

Ich danke auch noch Christoph Schmidt, dessen Konzepte die Anfangsgrundlage dieser Arbeit gebildet haben und Michael Lang und Hartmut Kalinowski, die immer für hilfreiche Ratschläge und technische Unterstützung gesorgt haben.

Außerdem danke ich meinen Bürokollegen, die immer für ein gutes Arbeitsklima gesorgt haben und so den Spaß an der Arbeit immer erhalten haben. Zusätzlich zu den bereits genannten, möchte ich hier Peter Klassen, Manuela Gottschall und Christoph Wendel erwähnen.

Ich danke auch Marcus Grüner für die gute Zusammenarbeit, besonders im Rahmen des COMPASS-Experiments.