Programmierung des Microcontrollers für die AMIGA Ausleseelektronik

Bachelorarbeit
zur Erlangung des akademischen Grades
Bachelor of Science
(B.Sc.)

dem Fachbereich Physik
der Universität Siegen

vorgelegt von
Martin Tigges

April 2009

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Vor fast vier Jahrhunderten richtete Galileo Galilei als einer der Ersten sein Fernrohr gegen den Himmel. Seither bestaunt der Mensch die unendlichen Weiten des Universums, die bis heute eine große Faszination auf die gesamte Menschheit ausüben. Dies liegt vorallem an den beeindruckenden Himmelsaufnahmen moderner Teleskope, mit denen jeder Spektralbereich von Radiowellen bis hin zur Röntgenstrahlung untersucht werden kann. In der Öffentlichkeit vergleichsweise unbemerkt geblieben sind jedoch die hochenergetischen Teilchen, mit denen die Erde unablässig aus dem Universum bombardiert wird. Auch wenn diese kosmische Strahlung zunächst unsichtbar erscheint, so hat sie doch großen Einfluss auf die Menschen und deren Verständnis des Universums, denn alle diese Teilchen führen einzigartige Informationen über die Vorgänge im Kosmos mit sich.
Die Astroteilchenphysik widmet sich diesen Informationen und versucht mehr über die Struktur der Materie im Kleinen und den Strukturen des Universums im Großen herauszufinden. Betrachtet man den Aspekt, dass die höchsten jemals auf der Erde beobachteten Teilchenenergien aus der kosmischen Strahlung stammen, so stellt sich die Frage nach der Herkunft dieser Teilchen.
Um eine Antwort auf diese Frage zu erhalten, wurde das internationale Pierre-Auger Observatorium ins Leben gerufen. Es hat sich zum Ziel gesetzt die höchstenergetischen Teilchen der kosmischen Strahlung zu untersuchen und so dem fundamentalen Rätsel der modernen Physik auf den Grund zu gehen. Das Observatorium wird aus einer 3000 km^2 großen Detektoranlage auf der Süd- und einer 10000 km^2 großen Anlage auf der Nordhalbkugel bestehen. Die erste Anlage in der argentinischen Pampa wurde im Frühjahr 2008 fertiggestellt. Gleichzeitig wurden Erweiterungen geplant, die das Experiment an die neuesten wissenschaftlichen Anforderungen anpassen.
Eine dieser Erweiterungen ist das AMIGA Projekt, welches aus zusätzlichen Komponenten zur Vermessung eines breiteren Energie- und Teilchenspektrums besteht. Die dazugehörige Ausleseelektronik wird von der Arbeitsgruppe Hochenergiephysik der Universität Siegen in Zusammenarbeit mit Universitäten in Argentinien entwickelt. Als Schnittstelle zwischen Ausleseelektronik und PC dient dabei ein Microcontroller, dessen Programmierung wesentlicher Bestandteil dieser Arbeit ist.

2 Kosmische Strahlung

Wie bereits erwähnt wird die Erde fortwährend von hochenergetischen Teilchen aus den Tiefen des Universums getroffen. Diesen Teilchenstrom bezeichnet man als *kosmische Strahlung*, früher auch *Höhenstrahlung* genannt.

2.1 Die Entdeckung

Entdeckt wurde die kosmische Strahlung 1912 von dem österreichischen Physiker Viktor Franz Hess. Er beschäftigte sich mit der Frage, woher die ionisierende Strahlung stammt, die auf der Erdoberfläche gemessen wurde. Vor seiner Entdeckung vermutete man radioaktive Nukleide in der Erdkruste als mögliche Ursache. Allerdings konnte Prof. Hess dies mit Heißluftballonexperimenten (bis 5000 m Höhe) widerlegen. Anhand der mitgeführten Magnetspektrometer stellte er fest, dass die Intensität der Strahlung zunächst wie erwartet mit zunehmender Höhe abnahm, ab einer bestimmten Höhe (ca. 1000 m) allerdings wieder merklich anstieg. Aus diesen Beobachtungen schloss er, dass uns Strahlung aus den Tiefen des Universums erreicht.

Der Entdeckung der Höhenstrahlung folgten weitere. Mit einer Nebelkammer konnte Dimitry Skobelzyn 1927 zum ersten mal Sekundärteilchen[1], die durch Wechselwirkung der kosmischen Strahlung mit Teilchen der Erdatmosphäre erzeugt werden, sichtbar machen.
Im Jahre 1932 wurde das Positron von Carl Anderson entdeckt, wofür er zusammen mit Viktor Franz Hess 1936 den Nobelpreis erhielt und den Grundstein der modernen Teilchenphysik legte.

2.2 Zusammensetzung der primären kosmischen Strahlung

Unter kosmischer Strahlung versteht man heute geladene Teilchen, die die Erde aus dem Weltall treffen.
Fand noch keine Wechselwirkung mit der Erdatmosphäre statt, bezeichnet man jene als *primäre* kosmische Strahlung. Sie besteht zu etwa 98% aus Atomkernen und zu 2% aus Elektronen, wobei sich die Atomkerne wiederum in 87% Wasserstoffkerne bzw. Protonen, 12% Heliumkerne bzw. α-Teilchen und etwa 1% schwerere Kerne

[1] Siehe Kapitel 2.6 auf Seite 11.

aufteilen.
Zum Vergleich ist in Abbildung 2.1 die Elementhäufigkeit in der primären kosmischen Strahlung und im Sonnensystem dargestellt.

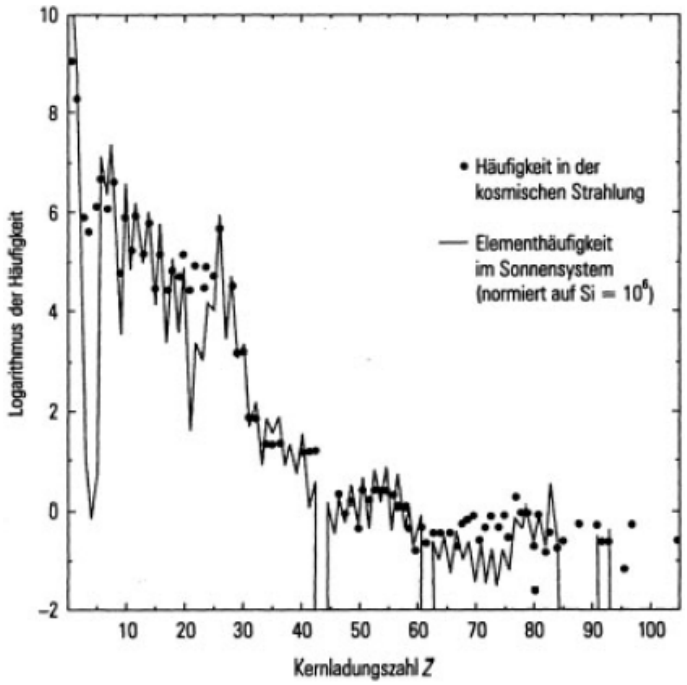

Abbildung 2.1: Elementhäufigkeit in der primären kosmischen Strahlung [Gru00]

Man erkennt viele Übereinstimmungen in der chemischen Zusammensetzung des solaren Systems und der kosmischen Strahlung. Beobachtete Abweichungen lassen sich durch Fragmentation[2] von schwereren Kernen in der galaktischen Materie auf dem Weg zur Erde erklären.

2.3 Energiespektrum der primären kosmischen Strahlung

Die kosmische Strahlung überdeckt einen sehr großen Energiebereich von 10^6 eV bis zu ultrahohen Energien von über 10^{20} eV. Dargestellt ist dies in Abbildung 2.2 für Energien oberhalb von 100 MeV. Hier ist der differentielle Fluss der kosmischen Strahlung in Abhängigkeit von der Energie aufgetragen. Die Daten stammen aus einer großen Anzahl von Messungen, und die Zahlen an der Kurve repräsentieren die integralen Teilchenflüsse oberhalb der Markierungen.

[2] Aufbrechung schwerer Kerne in leichtere Kerne durch Kollision.

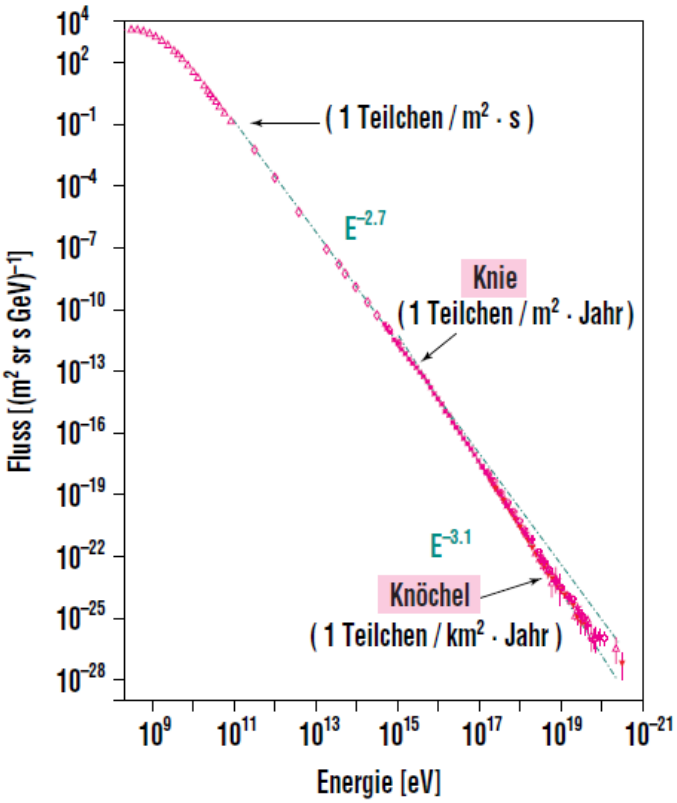

Abbildung 2.2: Energiespektrum der primären kosmischen Strahlung [Blü01]

Zu erkennen ist, dass der differentielle Teilchenfluss ab einer Energie von ungefähr 1 GeV (Ruheenergie eines Wasserstoffkerns bzw. Protons) steil abfällt. Er folgt dabei folgendem Potenzgesetz:

$$I(E) \propto E^{-\gamma} \quad . \tag{2.1}$$

Dabei bezeichnet γ den *spektralen Index*, der bis zu einer Energie von ungefähr $4 \cdot 10^{15}$ eV einen Wert von $\gamma = 2,7$ annimmt. Oberhalb dieser Energieschwelle steigt der spektrale Index auf $\gamma = 3,1$ an bis zu einer Energie von ungefähr $4 \cdot 10^{18}$ eV, wo er wieder auf $\gamma = 2,7$ abfällt. In der Struktur des Energiespektrums sind also zwei markante Knicke zu erkennen. Der Knick bei ca. $4 \cdot 10^{15}$ eV wird *erstes Knie* genannt und der zweite Knick bei etwa $4 \cdot 10^{18}$ eV wird als *Knöchel* bezeichnet. Neben den markanten Knicken, gibt es einen weniger offensichtlichen Knick im Bereich $10^{17} - 10^{18}$ eV, der als *zweites Knie* Einzug in die Literatur erhalten hat und einen spektralen Index von $\gamma = 3,3$ aufweist.
Die Entstehung des ersten und zweiten Knies sowie vorallem die Gründe für das Abknicken des Energiespektrums am Knöchel sind bislang noch nicht vollständig verstanden und stehen im Mittelpunkt der aktuellen Forschung.

Betrachtet man den Teilchenfluss am ersten Knie von rund einem Teilchen pro Quadratmeter und Jahr sowie am Knöchel von etwa einem Teilchen pro Quadratkilome-

ter und Jahr, so wird deutlich, dass man solche Teilchenenergien (ab $E = 10^{14}$ eV) nicht direkt beispielsweise durch Ballon- oder Satellittenexperimente nachweisen kann. Es sind vielmehr Detektoren notwendig, die eine große Detektorfläche bieten. Diese lassen sich allerdings nur in der Nähe des Erdbodens realisieren, was eine Messung der primären kosmischen Strahlung unmöglich macht. Daher muss man auf die indirekte Messung, also der Messung der sekundären kosmischen Strahlung[3] zurückgreifen. So wird zum Beispiel derzeit das Verhalten am ersten und zweiten Knie von dem Bodendetektorexperiment KASCADE-Grande[4] untersucht.
Für die Messung von den höchsten Teilchenenergien, deren Fluss sogar weniger als ein Teilchen pro Quadratkilometer und Jahrhundert beträgt, sind jedoch gigantische Anlagen wie das internationale Pierre-Auger-Observatorium[5] in Argentinien notwendig.

2.4 Ursprung der primären kosmischen Strahlung

Betrachtet man das breite Energiespektrum der primären kosmischen Strahlung, so stellt sich die Frage nach der Herkunft solch hochenergetischer Teilchen.
Da die geladenen Teilchen der kosmischen Strahlung in interstellaren Magnetfeldern vielfach abgelenkt werden und daher isotrop auf die Erde treffen, kann man mit Ausnahme der allerhöchsten Energien – oberhalb von ca. $5 \cdot 10^{19}$ eV – nicht über die Einfallsrichtung der Teilchen Rückschlüsse auf ihren Ursprung schließen. Es sind nur Rückschlüsse über die Zusammensetzung und das Energiespektrum möglich.
Im Folgenden sollen mögliche Quellen bzw. Beschleunigungsmechanismen mit den zu erreichenden Teilchenenergien vorgestellt werden.

2.4.1 Zyklotron-Mechanismus

Auf der Sonnenoberfläche existieren Bereiche, deren Temperatur geringer ist als ihre Umgebung und folglich dunkler erscheinen; die sogenannten *Sonnenflecken*. Diese Sonnenflecken erzeugen durch Bewegung des Plasmas, welches hauptsächlich aus Protonen und Elektronen besteht, zeitlich veränderliche Magnetfelder $\frac{dB}{dt}$, also einen zeitlich veränderlichen magnetischen Fluss $\frac{d\phi}{dt}$. Nach dem faradayschen Induktionsgesetz

$$-\frac{d\phi}{dt} = U \tag{2.2}$$

[3] Die Entstehung der sekundären kosmischen Strahlung wird im Detail in Kapitel 2.6 auf Seite 11 erläutert.
[4] KASCADE-Grande (KArlsruhe Shower Core and Array DEtector-Grande) ist ein Experiment des Forschungszentrums Karlsruhe. Es besteht aus einem Netz von 252 Bodendetektorstationen, mit Hilfe derer ausgedehnte Luftschauer vermessen werden. KASCADE-Grande ist die Erweiterung des 1996 beendeten, ursprünglichen KASCADE-Experiments.
[5] Siehe Kapitel 3 auf Seite 15.

wird durch die zeitliche Änderung des magnetischen Flusses eine Ringspannung U induziert. Mit dem magnetischen Fluss durch eine Kreisfläche[6]

$$\phi = \int \vec{B} \cdot d\vec{A} = B\pi R^2 \qquad (\vec{A} \| \vec{B}) \tag{2.3}$$

ergibt sich für die Energieänderung ΔE des Teilchens bei einem Umlauf um das zeitlich veränderliche Magnetfeld:

$$\Delta E = eU = e\pi R^2 \frac{dB}{dt} \qquad . \tag{2.4}$$

Setzt man nun entsprechende Werte für Magnetfelder pro Zeit $\frac{dB}{dt}$ und den Radius R des Sonnenflecks ein, kann man den Energiegewinn pro Umlauf eines Teilchens der Ladung e berechnen.

Durch den Zyklotron-Mechanismus können geladene Teilchen auf eine Energie von 10^{11} eV beschleunigt werden.

2.4.2 Sonnenfleckenpaare

Sonnenflecke treten häufig in Paaren mit entgegengesetzter magnetische Orientierung auf und bewegen sich aufeinander zu, bis sie zu einem späteren Zeitpunkt miteinander verschmelzen. Zur Vereinfachung betrachtet man, wie in Abbildung 2.3 dargestellt, einen Sonnenfleck als ruhend und den anderen als mit relativer Geschwindigkeit $\vec{v}$ bewegt.

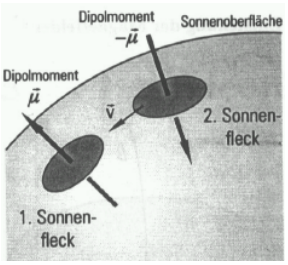

Abbildung 2.3: Schematische Darstellung eines Sonnenfleckenpaares [Gru00]

Beide Sonnenflecken wirken nun als bewegter Dipol, welcher ein elektrisches Feld erzeugt und so geladene Teilchen beschleunigen kann.

Durch die Sonnenfleckenpaare lassen sich Teilchenenergien von 10^9 eV erzeugen.

[6] Die Sonnenflecken werden als kreisförmig angenommen und das Magnetfeld steht senkrecht auf der Kreisfläche.

2.4.3 Schockwellenbeschleunigung

Bei einer Supernova wird durch die Explosion Materie mit sehr hoher Geschwindigkeit ins All geschleudert. Diese Materie stellt im Vergleich zum interstellaren Medium eine Schockfront dar, die sich mit bestimmter Geschwindigkeit im Universum ausbreitet. In Abbildung 2.4 ist schematisch dargestellt wie sich die erzeugte Schockfront mit Geschwindigkeit $\vec{u_1}$ im interstellaren Raum bewegt.

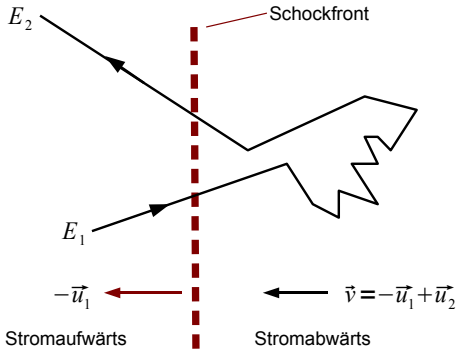

Abbildung 2.4: Schematische Darstellung der Schockwellenbeschleunigung [Def04]

Hinter der Schockfront strömt Gas mit der Geschwindigkeit $\vec{u_2}$ (System der Schockwelle) weg. Folglich beträgt die Geschwindigkeit des wegströmenden Gases im Laborsystem $\vec{u_1} - \vec{u_2}$. Trifft nun ein Teilchen mit der Energie E_1 auf die Schockfront, so gelangt es in den geschockten Bereich (rechts), gewinnt dort kinetische Energie und verlässt die Schockwelle mit größerer Energie E_2. Bei diesem Zyklus gewinnt das Teilchen Energie, die proportional zu $u_1 - u_2$ ist. Berücksichtigt man die variablen Streuwinkel und rechnet relativistisch, so erhält man eine allgemeine Formel für den Energiegewinn ΔE eines sich mit der Geschwindigkeit $v \approx c$ bewegten Teilchens:

$$\frac{\Delta E}{E} = \frac{4}{3}\frac{u_1 - u_2}{c} \quad . \tag{2.5}$$

Wird ein Teilchen zwischen zwei Schockfronten, die sich mit unterschiedlichen Geschwindigkeiten ausbreiten, mehrfach reflektiert, kann es auf sehr hohe Energien beschleunigt werden. Für den Energiegewinn eines Teilchens der Geschwindigkeit v und der Masse m ergibt sich folgender Zusammenhang:

$$\Delta E \approx mv\Delta u \quad . \tag{2.6}$$

Wobei Δu für die Geschwindigkeitsdifferenz der beiden Schockfronten steht.

Mit der Schockwellenbeschleunigung, die wegen ihrer Ähnlichkeit zum Fermi-Mechanismus[7] und der Linearität der Geschwindigkeit auch *Fermi-Mechanismus 1. Ordnung* genannt wird, lassen sich Maximalenergien von 10^{14} eV erklären.

[7] Benannt nach dem Entdecker Enrico Fermi.

2.4.4 Fermi-Mechanismus

Kosmische Teilchen können auch Energie gewinnen, indem sie mit Magnetwolken bzw. bewegten Plasmawolken wechselwirken und so beschleunigt werden. Man betrachtet dazu zwei Fälle. Zum einen wenn die Geschwindigkeit des einfallenden Teilchens $\vec{v}$ antiparallel zur Geschwindigkeit $\vec{u}$ der Gaswolke ist und zum anderen, dass $\vec{v}$ parallel zu $\vec{u}$ ist. Im ersten Fall gewinnt das Teilchen Energie und im zweiten Fall verliert es Energie. Somit ergibt sich im Mittel folgender Zusammenhang für den Energiegewinn ΔE eines sich kosmischen Teilchens der Masse m:

$$\Delta E = \Delta E_1 + \Delta E_2 = mu^2 \quad . \tag{2.7}$$

Der beschriebene Beschleunigungsmechanismus wird häufig wegen der Geschwindigkeit der Magnetwolke, welche quadratisch in die Berechnung eingeht, als *Fermi-Mechanismus 2. Ordnung* bezeichnet. Allerdings können kosmische Teilchen nicht mehrfach an der bewegten Plasmawolke reflektiert werden, wodurch die resultierenden Teilchenenergien als zu gering angesehen werden, um als mögliche Kandidaten für den Ursprung der ultrahochenergetischen Teilchen in Frage zu kommen.

2.4.5 Pulsare

Nach Supernovaeexplosionen bleiben als Überreste der Sonnen schnell rotierende Neutronensterne zurück, die *Pulsare* genannt werden. Dabei handelt es sich um sehr kompakte Objekte, deren Dichten vergleichbar mit den Dichten von Atomkernen sind, und die typischen Radien von $R_{Pulsar} = 20$ km aufweisen. Im Vergleich dazu beträgt der typische Radius eines Sterns $R_{Stern} = 10^6$ km. Betrachtet man nun den magnetischen Fluss vor und nach dem Gravitationskollaps

$$\int_{Stern} \vec{B}_{Stern} \cdot d\vec{A}_{Stern} = \int_{Pulsar} \vec{B}_{Pulsar} \cdot d\vec{A}_{Pulsar} \quad , \tag{2.8}$$

so ergibt sich für das Magnetfeld des Pulsars B_{Pulsar}:

$$B_{Pulsar} = B_{Stern} \cdot \frac{R^2_{Stern}}{R^2_{Pulsar}} \quad . \tag{2.9}$$

Es entstehen also außergewöhnlich hohe Magnetfelder, welche experimentell bestätigt wurden. Da die Rotationsachse der Pulsare selten mit der Richtung des magnetischen Feldes übereinstimmt, erfahren die Magnetfelder zeitliche Änderungen. Folglich wird eine Spannung induziert und der Energiegewinn kosmischer Teilchen lässt sich ähnlich zur Zyklotron-Beschleunigung[8] bestimmen.

Durch die großen Magnetfelder der Pulsare lassen sich die Teilchen auf Energien von 10^{19} eV beschleunigen.

[8] Siehe Kapitel 2.4.1 auf Seite 6

2.4.6 Binärsystem

Ein System aus einem normalen Stern und einem Neutronenstern bezeichnet man als *Binärsystem*. Dabei saugt der erheblich massivere Neutronenstern Materie vom normalen Stern auf und es entsteht ein Plasmastrom. Aufgrund der Drehimpulserhaltung bildet der Plasmastrom dabei eine Akkretionsscheibe um den Neutronenstern. Aufgrund der bewegten Teilchen des enormen Plasmastroms, bilden sich große elektromagnetische Felder aus. Das magnetische Feld des Neutronenstern steht dabei senkrecht auf der Akkretionsscheibe und somit wirkt die Lorentzkraft auf die geladenen Teilchen, sodass für den Energiegewinn ΔE gilt:

$$\Delta E = \int \vec{F} \cdot d\vec{s} = evB\Delta s \qquad (\vec{v} \perp \vec{B}) \qquad , \tag{2.10}$$

wobei

$$\vec{F} = e(\vec{v} \times \vec{B}) \tag{2.11}$$

die Lorentzkraft darstellt.

Auf diese Weise ergeben sich Teilchenenergien von 10^{19} eV.

2.4.7 Aktive galaktische Kerne

Unter den sogenannten *aktiven galaktischen Kernen* versteht man supermassive schwarze Löcher in der Größenordnung von $10^5 - 10^{10}$ Sonnenmassen, welche Materie in Bereichen von einigen Kiloparsec[9] akkreditieren. Senkrecht zur Akkretionsscheibe bilden sich dabei Plasmaströme von einigen Megaparsec Länge aus, welche als *Jets* bezeichnet werden. Da sich die Jets nicht gleichförmig sondern stoßwellenartig ausbreiten, werden geladene Teilchen beschleunigt.

Man vermutet diese Objekte als mögliche Ursache von höchstenergetischen Teilchen mit Energien über 10^{20}eV.

2.5 GZK-Cutoff

Der GZK-Cutoff ist nach den Physikern *Kenneth Greisen*, *Vadim Kuzmin* und *Georgiy Zatsepin* benannt und beschreibt eine obere Grenze für kosmische Teilchenenergien weit entfernter Quellen. Protonen p mit einer Energie oberhalb von $6 \cdot 10^{19}$ eV wechselwirken mit den Photonen γ der sogenannten *kosmischen Hintergrundstrahlung*[10] und erzeugen Pionen $\pi_{\pm/0}$ gemäß folgender Reaktionen:

[9] 1 Parsec ist die Entfernung, von der aus betrachtet eine astronomische Einheit (mittlerer Abstand der Erde von der Sonne) unter einem Winkel (der Parallaxe) von einer Bogensekunde $(1/3600)°$, erscheint. Umrechnung: 1 pc entspricht $3,085 \cdot 10^{16}$ m

[10] Kosmische Schwarzkörper-Strahlung mit einer derzeitigen Temperatur von $2,7$ K als Relikt des Urknalls.

$$p + \gamma_{CMB} \longrightarrow p + \pi^0 \quad , \tag{2.12}$$

$$p + \gamma_{CMB} \longrightarrow n + \pi^+ \quad . \tag{2.13}$$

Die Pionerzeugung geht allerdings mit einem Energieverlust der Protonen einher, sodass sie nach einer mittleren freien Weglänge von 100 Mpc die GZK-Schwelle unterschreiten. Detektiert man nun Protonen mit einer Energie von $E > 6 \cdot 10^{19}$ eV, können diese demnach nur von Quellen innerhalb eines Umgebungsradius von 100 Mpc stammen.

2.6 Sekundäre kosmische Strahlung

Sekundäre kosmische Strahlung entsteht durch Wechselwirkung der primären kosmischen Teilchen mit Nukleonen der Erdatmosphäre. Die daraus entstehenden Reaktionsprodukte wechselwirken wiederum mit weiteren Atomkernen der Atmosphäre, sodass ein sogenannter *ausgedehnter Teilchenschauer* entsteht. Ein solcher Teilchenschauer ist in Abbildung 2.5 für ein Proton als Primärteilchen dargestellt.

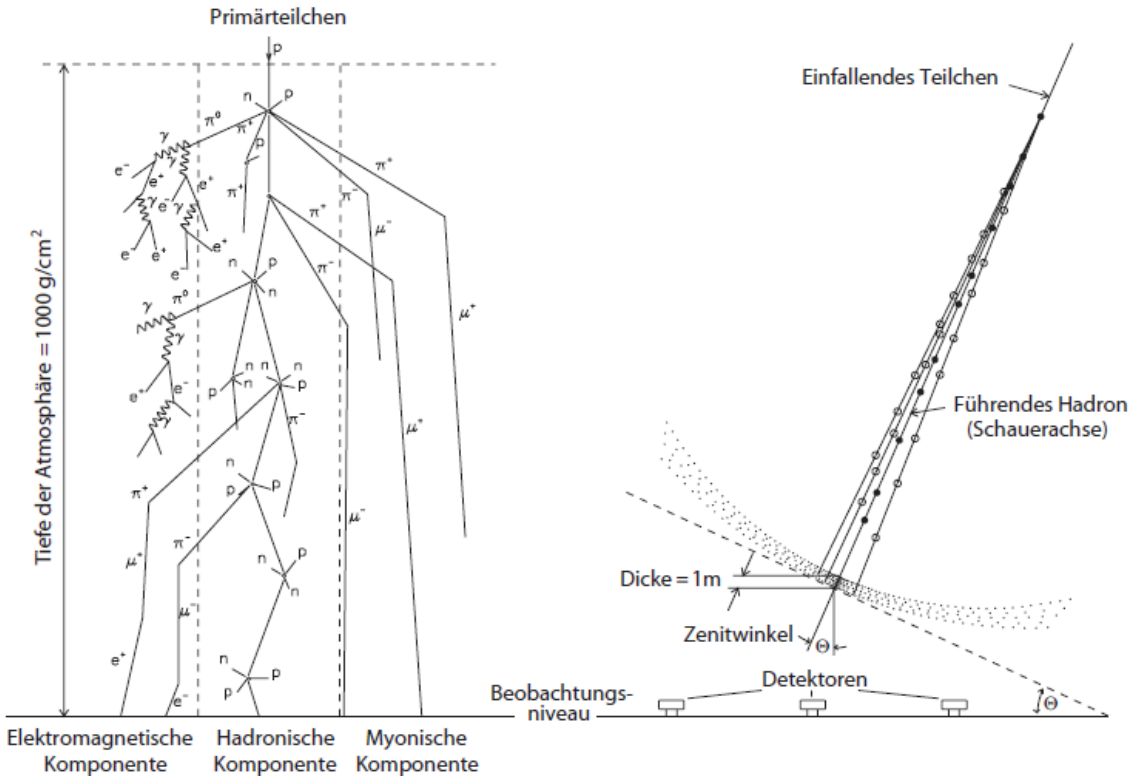

Abbildung 2.5: Links: Entwicklung eines Teilchenschauers; Rechts: Seitenansicht eines Teilchenschauers [Alk75]

Man unterteilt die Kaskade an Sekundärteilchen im Wesentlichen in drei Komponenten: Die hadronische, myonische und elektromagnetische Komponente.

Betrachtet man ein Proton als Primärteilchen, so gilt für die erste Wechselwirkung mit einem Atmosphärenteilchenkern N:

$$p + N \longrightarrow p' + N' + a\pi^+ + a\pi^- + a\pi^0 \quad . \tag{2.14}$$

Es entstehen also hauptsächlich[11] weitere Protonen p', Neutronen bzw. Nukleonen N' und durchschnittlich zu gleicher Anzahl a positiv geladene Pionen π^+, negativ geladene Pionen π^- und neutrale Pionen π^0. Diese Teilchen gehören alle zur Familie der Hadronen und beschreiben somit die **hadronische Komponente**. Im weiteren Verlauf zerfallen die neutralen Pionen aufgrund ihrer kurzen mittleren Lebensdauer ($\tau_{\pi^0} = 8,4 \cdot 10^{-17}$ s) praktisch sofort in zwei Photonen

$$\pi^0 \longrightarrow \gamma + \gamma \tag{2.15}$$

und initialisieren dadurch **elektromagnetische Kaskaden**. Die so entstandenen Photonen erzeugen durch Paarbildung[12]

$$\gamma + N \longrightarrow N' + e^- + e^+ \tag{2.16}$$

Elektron-Positron-Paare. Anschließend senden sowohl die Elektronen e^- als auch die Positronen e^+ durch Bremsstrahlung[13] Photonen aus, welche in der nächsten Wechselwirkung wiederum Elektronen und Positronen erzeugen und somit einen elektromagnetischen Schauer ausbilden. Das Maximum der elektromagnetischen Kaskade ist dann erreicht, wenn der Energieverlust durch Ionisation gleich dem Energieverlust durch Bremsstrahlung ist.

Die geladenen Pionen der hadronischen Komponente haben größere mittlere Lebensdauern ($\tau_{\pi^\pm} = 2,6 \cdot 10^{-8}$ s). Daher können sie auf diesem Wege weiter mit der Atmosphäre wechselwirken bis sie leptonisch in Myonen μ und Neutrinos ν zerfallen

$$\pi^+ \longrightarrow \mu^+ + \nu_\mu \tag{2.17}$$

$$\pi^- \longrightarrow \mu^- + \overline{\nu}_\mu \tag{2.18}$$

und die **myonische Komponente** bilden[14]. Hochenergetische Myonen können aufgrund der Verlängerung der mittleren Lebensdauer ($\tau_\mu = 2,2 \cdot 10^{-6}$ s) durch den relativistischen Effekt der Zeitdilatation und ihrer geringen Wechselwirkungsrate

[11] Neben den Pionen, Protonen und Neutronen entstehen auch Kaonen. Diese können aber wegen ihrer geringen Anzahl im Vergleich zu den Pionen vernachlässigt werden.

[12] Photonen können im elektrischen Feld eines Atomkerns ein Elektron-Positron-Paar erzeugen.

[13] Energieverlust geladener Teilchen bei Beschleunigung durch Aussendung von Photonen

[14] Auf die Neutrinos wird hier nicht weiter eingegangen, da sie so gut wie nicht wechselwirken und sich aus diesem Grund auch nur sehr schwer nachweisen lassen.

mit großer Wahrscheinlichkeit die Erde erreichen. Da sich die Anzahl der Elektronen nach Erreichen des Schauermaximums der elektromagnetischen Kaskade wesentlich stärker verringert als die myonische Komponente, kann anhand des Verhältnisses von Elektronen zu Myonen Rückschluss auf die Höhe des ersten Wechselwirkungspunkts geschlossen werden.

Durch die beschriebene Entwicklung eines Teilchenschauers erzeugt beispielsweise ein primäres Proton mit einer Energie von $E = 10^{15}$ eV etwa 10^6 Sekundärteilchen auf Meereshöhe, wovon 80% Myonen sind.

3 Das Pierre-Auger-Observatorium

Das internationale *Pierre-Auger-Observatorium*[1] ist derzeit das größte Experiment zur indirekten Messung der kosmischen Strahlung.
Nachdem es Experimenten wie AGASA[2] oder FLy's Eye[3] gelang Teilchenenergien oberhalb von $E = 10^{20}$ eV nachzuweisen, besteht das Ziel des Pierre-Auger-Observatoriums darin genügend Statistik in diesem Bereich der höchsten Energien von $10^{17} - 10^{21}$ eV zu sammeln, um so die Frage nach der Herkunft ultrahochenergetischer Teilchen beantworten zu können. Ein weiteres Ziel ist die Erforschung des Knöchels im Energiespektrum[4] bzw. die Erforschung des GZK-Cutoffs[5] als mögliche Ursache dafür.
Stationiert ist das südliche Auger-Observatorium in der Provinz Mendoza in Argentinien, wo es im Jahre 2004 mit der Datenaufnahme begonnen hat.
Bei dieser Anlage kommt erstmals eine Hybrid-Technik zur Messung der ausgedehnten Teilchenschauer zum Einsatz. Diese besteht aus einer Kombination von einem Bodendetektorfeld und optischen Teleskopen, wodurch die Messunsicherheiten erheblich reduziert und die Daten gegenseitig ergänzt werden. Insgesamt besteht das Experiment aus 1600 hexagonal im Abstand von $1,5$ km angeordneten Oberflächendetektoren, die eine Fläche von 3000 km^2 einschließen. Umgeben werden diese von vier Beobachtungsstationen mit jeweils sechs Fluorenszenzteleskopen.
Man erwartet einen Teilchenfluss für dieses Observatorium von ca. 5000 Ereignissen oberhalb von $E = 10^{19}$ eV und ca. 40 oberhalb von $E = 10^{20}$ eV pro Jahr.

In Abbildung 3.1 ist das Anfang des Jahres 2008 fertig gestellte südliche Pierre-Auger-Experiment dargestellt.

[1] Benannt nach dem französischen Physiker Pierre Victor Auger, dem es schon 1938 gelang den kosmischen Ursprung der Teilchenschauer anhand von Bodendetektorexperimenten nachzuweisen.

[2] AGASA (Akeno Giant Air Shower Array) befand sich in Akeno (Japan) und hat mit Bodendetektoren 11 Ereignisse oberhalb von $E = 10^{20}$ eV gemessen, obwohl weniger als ein Ereignis erwartet worden war. Daten wurden von 1990 - 2002 genommen.

[3] FLy's Eye (Fliegenaugenprinzip) war das erste Experiment, das Teilchenschauer anhand von Fluoreszenzstrahlung nachgewiesen hat. Von 1981 - 1993 wurden in Utah Daten aufgenommen und es konnte ein Ereignis mit einer Energie von $E = 3 \cdot 10^{20}$ eV gemessen werden. Das Nachfolgeexperiment war HiRes (bis 2004). AGASA und HiRes wurden in Utah zu dem Experiment Telescope Array zusammengelegt und haben 2007 mit der Datennahme begonnen.

[4] Siehe Kapitel 2.3 auf Seite 4.

[5] Siehe Kapitel 2.5 auf Seite 10.

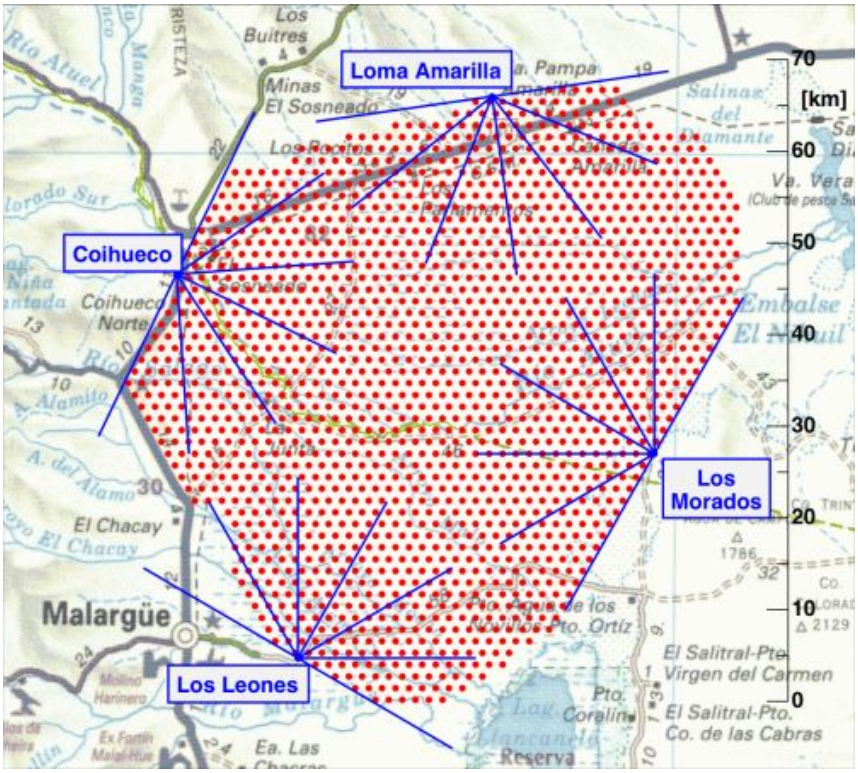

Abbildung 3.1: Das südliche Pierre-Auger-Observatorium in Mendoza (Argentinien); dargestellt sind die Bodendetektoren (rot) und die Beobachtungsstationen mit ihren Fluorenszenzteleskopen bzw. deren Sichtfelder (blau) [augde]

3.1 Bodendetektor

Alle 1600 Bodendetektoren sind gleich aufgebaut. Sie bestehen aus einem zylindrischen Tank von 10 m^2 Grundfläche und 120 cm Höhe, der mit 12 Tonnen hochreinem[6] Wasser gefüllt ist. Das vollkommen autarke System wird über Solarzellen mit Strom versorgt und sendet die gemessenen Daten via Antennen an eine Empfängerstation. Da die Lichtgeschwindigkeit im Wasser etwa um den Faktor $1,3$ geringer ist als im Vakuum, können sich hochenergetische Teilchen im Wasser schneller bewegen als das Licht und strahlen dadurch das sogenannte *Cherenkov-Licht*[7] ab. Das bläuliche Cherenkov-Licht lässt sich dann mittels drei Photomultipliern detektieren. Dabei werden schwache Lichtsignale (bis hin zu einigen Photonen) registriert, verstärkt und so ein ausreichend großes elektrisches Signal erzeugt, das anschließend über einen DAC (Digital-Analog-Converter) in einen entsprechenden Datenstrom umge-

[6] Hochreines Wasser wird deswegen verwendet, um zum einen die Entstehung von Bakterien zu vermeiden und zum anderen Absorptionseffekte der Photonen im Wasser zu minimieren.

[7] Cherenkov-Licht ist benannt nach seinem Entdecker Pawel Alexejewitsch Tscherenkow und entsteht ähnlich zu einem Überschallkegel, wenn sich geladene Teilchen schneller als die Lichtgeschwindigkeit durch ein Medium c/n (n: Brechungsindex) bewegen.

wandelt wird. Die exakte Zeit wird über GPS (Global Positioning System) bestimmt, wodurch Informationen über die Ankunftszeit und den zeitlichen Verlauf gewonnen werden können. So erhält man letztendlich Hinweise auf die Art des primären kosmischen Teilchens und dessen Energie. Die Richtung des primären Teilchens kann aus der zeitlichen Differenz angesprochener Detektortanks bestimmt werden.
Ein schematischer Aufbau einer Bodendetektorstation ist in Abbildung 3.2 zu sehen.

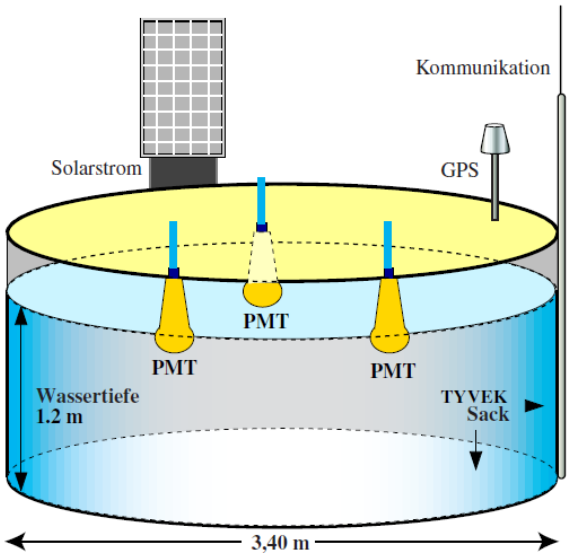

Abbildung 3.2: Schematischer Aufbau eines Detektortanks [Kam00]

3.2 Fluoreszenzdetektor

Die 24 Fluoreszenzteleskope – aufgeteilt auf vier Beobachtungsstationen – beobachten die Atmosphäre oberhalb des Bodendetektorfelds. Wie in Abbildung 3.3 erkennbar, bestehen sie im Wesentlichen aus einem 12 m^2 großen sphärischen Spiegel (Radius: $3,4$ m), der das ankommende Licht bündelt und auf eine Kamera abbildet. Die Kamera besteht dabei aus 440 hexagonalen Photomultipliern, jeweils 20 in horizontaler und 22 in vertikaler Richtung, mit denen das Fluoreszenzlicht detektiert wird.
Das Fluoreszenzlicht entsteht für Primärteilchenenergien oberhalb von $E = 10^{16}$ eV, wenn die geladenen Teilchen des Schauers mit den Stickstoffmolekülen der Atmosphäre wechselwirken. Im Detail besteht die Wechselwirkung aus Anregung und Ionisation von Stickstoffmolekülen. Gehen jene angeregten Moleküle wieder in ihren Grundzustand über, so emittieren sie sogenanntes *Fluoreszenzlicht*[8]. In mondlosen Nächten kann das Fluoreszenzlicht von den PMTs detektiert werden und die komplette Entwicklung eines Luftschauers wird nachvollziehbar.

[8] Aus diesem Grund befindet sich auch ein Filter (Abbildung 3.3) mit einer Wellenlänge von $300-450$ nm vor den Teleskopen, der Licht oberhalb von 450 nm absorbiert und so das Streulicht auf ein Minimum reduziert.

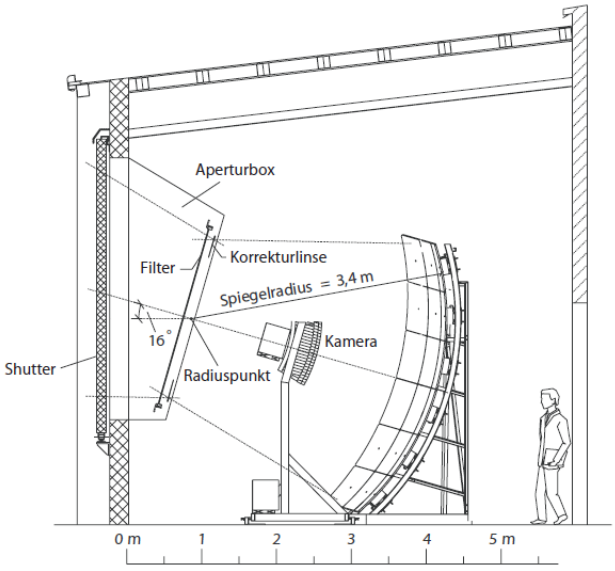

Abbildung 3.3: Schematischer Aufbau eines Fluoreszenzdetektors [Bet03]

3.3 Schauerbeispiel

Zur Illustration des Zusammenspiels von Boden- und Fluoreszenzdetektor ist in Abbildung 3.4 ein Beispiel für die Beobachtung eines Luftschauers gegeben.

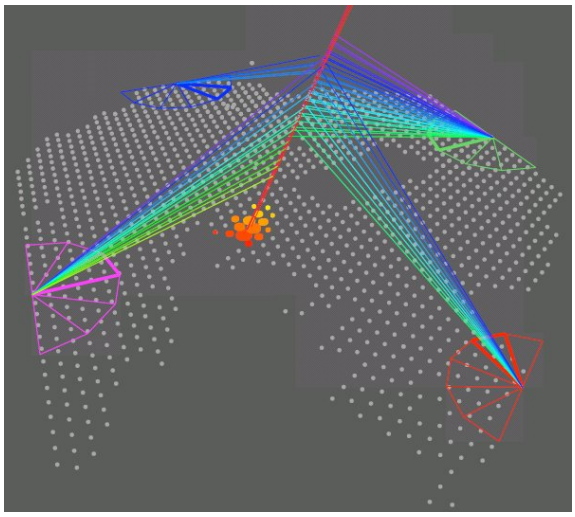

Abbildung 3.4: Beobachtung eines Luftschauers [augorg]

Da hier sogar alle vier Beobachtungsstationen dieses Ereignis registrieren und die Entwicklung verfolgen, muss es sich um ein extrem hochenergetisches Teilchen handeln, dass eine enorme Anzahl an Sekundärteilchen produziert.
Klar zu erkennen sind die Vorteile der verwendeten Hybrid-Technologie: Auf der einen Seite werden Messunsicherheiten sowie systematische Fehler auf ein Minimum

reduziert, da beide Detektortypen unabhängig voneinander die Richtung, Energie und Zusammensetzung des primären Teilchens rekonstruieren und so eine gegenseitige Überprüfung möglich wird. Auf der anderen Seite ergänzen sich die gewonnen Daten. Da das Bodendetektorfeld vor allem Informationen über die *laterale* Schauerentwicklung sammelt und die Fluoreszenzteleskope die *longitudinale* Verteilung vermessen, kann so zusammenfassend der gesamte ausgedehnte Teilchenschauer rekonstruiert werden.

3.4 Erweiterungen

Das mittlerweile vollständig aufgebaute Pierre-Auger-Observatorium in Argentinien soll mindestens die nächsten zehn Jahre Präzisionsmessungen kosmischer Teilchen im Energiebereich von $10^{18} - 10^{21}$ eV durchführen.
Allerdings hat die Auger Kollaboration beschlossen neue Komponenten dem Experiment hinzuzufügen, um so den Energiebereich um eine Dekade nach unten auszudehnen, eine Verbesserung des Myonnachweises zu erreichen und neue Messtechniken zu erproben. Die Entwicklung und Installation der Erweiterungen hat bereits begonnen und soll Ende des Jahres 2010 abgeschlossen sein.
Die physikalische Motivation hinter der Ausdehnung des Energiebereichs auf $10^{17} - 10^{21}$ eV liegt in der Untersuchung des zweiten Knies[9], da man dort den Übergang von galaktischer zu extragalaktischer kosmischer Strahlung erwartet. Dieser Übergang geht mit einer Änderung der Zusammensetzung der primären kosmischen Strahlung einher. Wie schon erläutert ist für die Bestimmung der Art des primären Teilchens das Elektron-Myon-Verhältnis auf der Detektorebene notwendig[10]. Um dieses besser messen zu können, wurde die Erweiterung zur Verbesserung des Myonnachweises beschlossen.
Im Folgenden sollen die Erweiterungen des Auger-Experiments im Detail erläutert werden.

3.4.1 AMIGA

Eine der Erweiterungen wird *AMIGA (Auger Myons and Infill for the Ground Array)* genannt und besteht aus zwei Teilen.
Zum einen soll das in Abbildung 3.5 dargestellte Bodendetektorfeld (Ground Array) auf einer Fläche von 25 km^2 (AMIGA Infill Array) verdichtet werden, indem weitere 85 Wassertanks zwischen die vorhandenen Tanks gesetzt werden und so die Gitterweite auf einen Abstand von 750 m (grüne Punkte in Abbildung 3.5) bzw. 433 m (rote Punkte in Abbildung 3.5) verringert wird. Die höhere Bodendetektordichte hat zur Folge, dass die Nachweiswahrscheinlichkeit für ausgedehnte Teilchenschauer im Energiebereich von $10^{17} - 10^{18}$ eV deutlich erhöht wird.
Zum anderen besteht die AMIGA Erweiterung aus zusätzlichen Myonzählern. In der

[9] Siehe Kapitel 2.3 auf Seite 4.
[10] Siehe Kapitel 2.6 auf Seite 11.

unmittelbaren Nähe dieser 85 Bodendetektoren werden in 3 m Tiefe je vier dieser 10 m^2 großen Myonzähler vergraben[11]. Mittels dieser Myonzähler ist ein genauerer Nachweis der Myonen möglich, wodurch die Bestimmung der Art des primären Teilchens erleichtert wird.

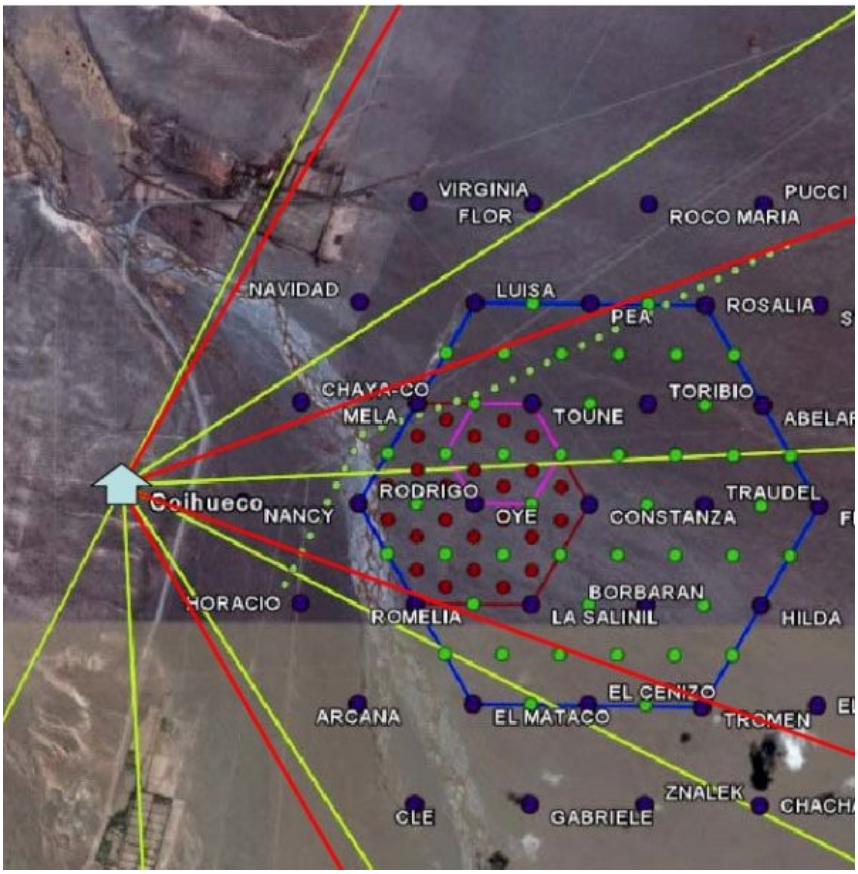

Abbildung 3.5: Darstellung des AMIGA Infill Arrays (Bodendetektoren: grüne Punkte) und der HEAT Teleskope (Sichtfeld: rote Linien) [augde]

3.4.2 HEAT

Die sogenannte *HEAT (High Evaluation Auger Telescopes)* Erweiterung besteht aus drei weiteren Fluoreszenzteleskopen, die bei der Beobachtungsstation Coihueco eingebaut werden sollen. Diese sollen zusätzlich den Nachthimmel über dem AMIGA Infill Array beobachten. Die Besonderheit der HEAT Teleskope besteht in einem größeren Sichtfeld in polarer Ausdehnung (28° - 58°), wodurch es möglich wird Teilchenschauer bereits ab einer Energie von $E = 10^{17}$ eV zu detektieren, da diese eher aussterben. Zusammen mit der AMIGA Erweiterung können somit auch Schauerprofile von $10^{17} - 10^{18}$ eV präzise vermessen werden.

[11]Der genaue Aufbau des Myonzählers wird in in Kapitel 4 auf Seite 23 erklärt.

3.4.3 Radio antenna

Eine weitere Erweiterung ist das als *Radio antenna* bezeichnete etwa 20 km^2 große Testfeld von Radioantennen. Mittels den in das AMIGA Infill Array eingebauten Radioantennen sollen zusätzlich hochenergetische Teilchenschauer vermessen werden. Ziel ist es diese Methode zu testen und zu optimieren, um sie später eventuell als kostengünstigere und effektivere Messmethode einzusetzen.

3.4.4 Auger Nord

Bisher wurde nur der südliche Teil des Pierre-Auger-Observatoriums erwähnt. Aber derzeit ist ein erheblich größerer nördlicher Teil in Planung. Stationiert werden soll dieser in Colorado (USA) nahe der Stadt Lamar. Geplant sind 4000 Bodendetektoren auf einem quadratischen Gitter mit einer Meile Maschenweite. Insgesamt wird also eine Bodendetektorfläche von über 10000 km^2 erreicht. In klaren, mondlosen Nächten wird diese Fläche von 18 Fluoreszenzteleskopen überschaut. Die verwendeten Techniken basieren auf den erprobten Bauteilen in Argentinien und werden den in Colorado herrschenden Witterungsverhältnissen angepasst.
Bisher konnte mit dem südlichen Observatorium lediglich kosmische Strahlung untersucht werden, die auf die südliche Erdhalbkugel trifft. Mit der Fertigstellung des Observatoriums in Colorado kann das Pierre-Auger-Observatorium den gesamten Himmel untersuchen.

4 Der AMIGA Myonzähler

Wie bereits erläutert, besteht der zweite Teil der AMIGA Erweiterung aus zusätzlichen Myonzählern, die in der unmittelbaren Nähe der 85 Tanks des AMIGA Infill Arrays in 3 m Tiefe vergraben werden. Dadurch wird sichergestellt, dass es sich bei den detektierten Teilchen um Myonen handelt, da alle anderen Teilchenarten aufgrund ihrer vergleichbar hohen Wechselwirkungswahrscheinlichkeit von der Erde absorbiert werden[1]. Derzeit sind vier Zähler mit einer Gesamtfläche von 30 m^2 pro Tank geplant.
Der gesamte Aufbau eines Myonzählers und vor allem die Ausleseelektronik ist schematisch in Abbildung 4.1 zu sehen.

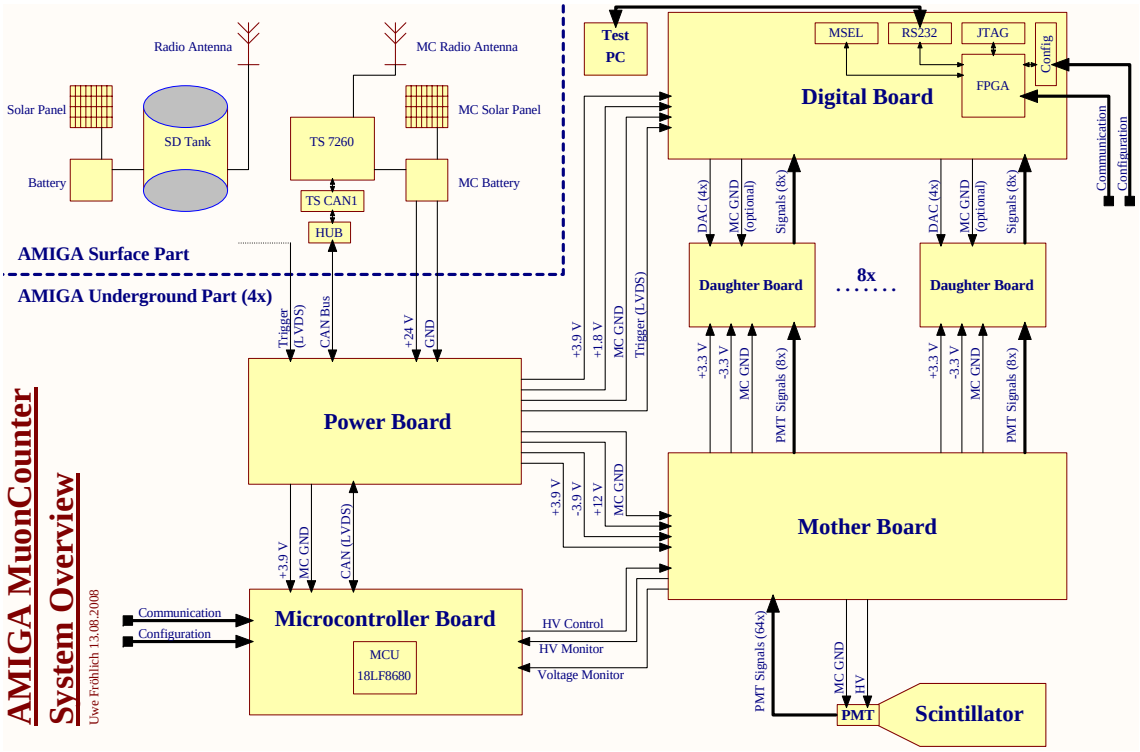

Abbildung 4.1: Steuer- und Ausleseelektronik des AMIGA Myonzählers [Frö09]

Ersichtlich ist, dass der vom Bodendetektor unabhängige Myonzähler aus einem ober- und einem unterirdischen Teil aufgebaut ist.
Dabei besteht der oberirdische Teil aus Stromversorgung und Datenübertragung zur

[1]Eine Ausnahme stellen Neutrinos dar. Da diese so gut wie gar nicht wechselwirken, können sie weder absorbiert noch detektiert werden.

Basisstation. Die Stromversorgung wird durch Solarzellen und eine Pufferbatterie (24 V) sichergestellt. Die Kommunikation mit der Basisstation für das Übermitteln der aufgenommenen Daten und den Empfang von Befehlen wird durch eine Radioantenne (802, 11 WiFi) realisiert.
Unterirdisch sind die vier Szintillatormodule und ihre Ausleseelektroniken positioniert. Im Detail besteht ein solches Szintillatormodul aus einem hochauflösenden Szintillator[2] mit 64 Szintillatorstreifen. Diese Streifen haben in zwei der vier Module eine Länge von 400 cm und in den anderen zwei eine Länge von 200 cm. Die Breite und Höhe der Streifen ist in allen Modulen gleich. Sie sind 4, 1 cm breit und 1 cm hoch. Somit wird insgesamt von dem Myonzähler eine Fläche von $2 \cdot 5\ \mathrm{m}^2 + 2 \cdot 10\ \mathrm{m}^2 = 30\ \mathrm{m}^2$ abgedeckt.
Trifft nun ein Myon auf einen Szintillatorsreifen, werden Photonen aus dem Szintillatormaterial ausgelöst und über die entsprechende optische Fiber (Glasfaserkabel) an einen 64 Pixel Photomultiplier[3] (PMT) weitergeleitet, der diese dann in elektrische Signale umwandelt. Die so erzeugten Signale werden anschließend von der Ausleseelektronik verarbeitet.

[2]Körper, der beim Durchgang von energiereichen geladenen Teilchen angeregt wird und die Anregungsenergie in Form von Licht wieder abgibt. Die im Szintillator deponierte Energie jedes einzelnen Stoßvorgangs lässt sich durch Messung der Lichtmenge bestimmen.
[3]Spezielle Elektronenröhre, die aus einer Photokathode und nachgeschaltetem Sekundärelektronenvervielfacher besteht, und so schwache Lichtsignale detektieren und in ein entsprechend großes elektrisches Signal umwandeln kann.

4.1 Die AMIGA Ausleseelektronik

Die in Abbildung 4.1 dargestellte Ausleseelektronik des AMIGA Myonzählers besteht aus fünf verschiedenen Elektronikkarten, die ineinander gesteckt und mit dem PMT verbunden werden.

4.1.1 Power Board

Die als *Power Board* bezeichnete Elektronikkarte ist für die Stromversorgung der anderen Bauteile zuständig. Die nötige 24 V Betriebsspannung bezieht das Power Board von der Batterie auf der Oberfläche. Diese Spannung wird über interne Schaltkreise in die entsprechenden Versorgungsspannungen umgewandelt und an das Digital, Mother und Microcontroller Board weitergeleitet. Durch die Verwendung eines Transformators werden Eingangs- und Ausgangsspannung auf dem Power Board galvanisch voneinander getrennt, so dass die Oberflächenelektronik und die Ausleseelektronik unterschiedliche Groundlevel besitzen.
Neben der Zuführung der Versorgungsspannung besitzt das Power Board Schnittstellen für das Triggersignal als LVDS (Low Voltage Differential Signal) und den CAN[4]-Bus von dem oberirdischen Teil. Das CAN-Bus Signal wird anschließend an das Microcontroller Board und das Triggersignal an das Digital Board weitergeleitet.

4.1.2 Mother Board

Der PMT und die acht Daughter Boards sind mit dem sogenannten *Mother Board* verbunden. Diese werden darüber auch mit den benötigten Versorgungsspannungen und dem Myonzähler Groundlevel versorgt. Bei der Spannungsversorgung für den PMT handelt es sich um Hochspannung, die aus 12 V erzeugt und mit einem High Voltage Divider vermessen wird.
Zusätzlich ist der PMT über seine 64 Auslesekanäle mit dem Mother Board verbunden. Die ankommenden 64 PMT Signale werden auf die acht Daughter Boards verteilt.

4.1.3 Daughter Board

Wie bereits erwähnt sind die sogenannten *Daughter Boards* mit dem Mother Board verbunden. Jedes liest jeweils acht der 64 Kanäle aus. Ein implementierter Operationsverstärker invertiert und verstärkt zunächst die ankommenden analogen PMT Signale. Im Anschluss vergleicht ein Comparator das verstärkte Signal mit einer Referenzspannung und erzeugt ein digitales Signal, das 0 V (Low Level) ausgibt, wenn die Amplitude des Signals über der Referenzspannung liegt. Der Referenzwert kann dabei für jeden der acht Kanäle eines Daughter Boards separat über einen DAC (Digital-Analog-Converter) gesteuert werden.

[4]Controller Area Network. CAN gehört zu den asynchronen Feldbussen, also zu einem standardisiertem Kommunikationssystem, das ursprünglich für die Automobilindustrie entwickelt wurde.

Letztendlich wird also ein digitales Signal ausgegeben, dem man entnehmen kann, ob in dem entsprechendem Kanal ein Myon detektiert wurde oder nicht.

4.1.4 Digital Board

Die Elektronikkarte, welche nun die digitalen Signale der Daughter Boards empfängt, wird *Digital Board* genannt. Es hat etwa die gleichen Abmessungen wie das Mother Board und ist parallel dazu angeordnet, da die Daughter Boards senkrecht in beide Karten gesteckt werden. Zur Illustration ist in Abbildung 4.2 das Sandwich aus Mother Board, Digital Board und den Daughter Boards dargestellt.

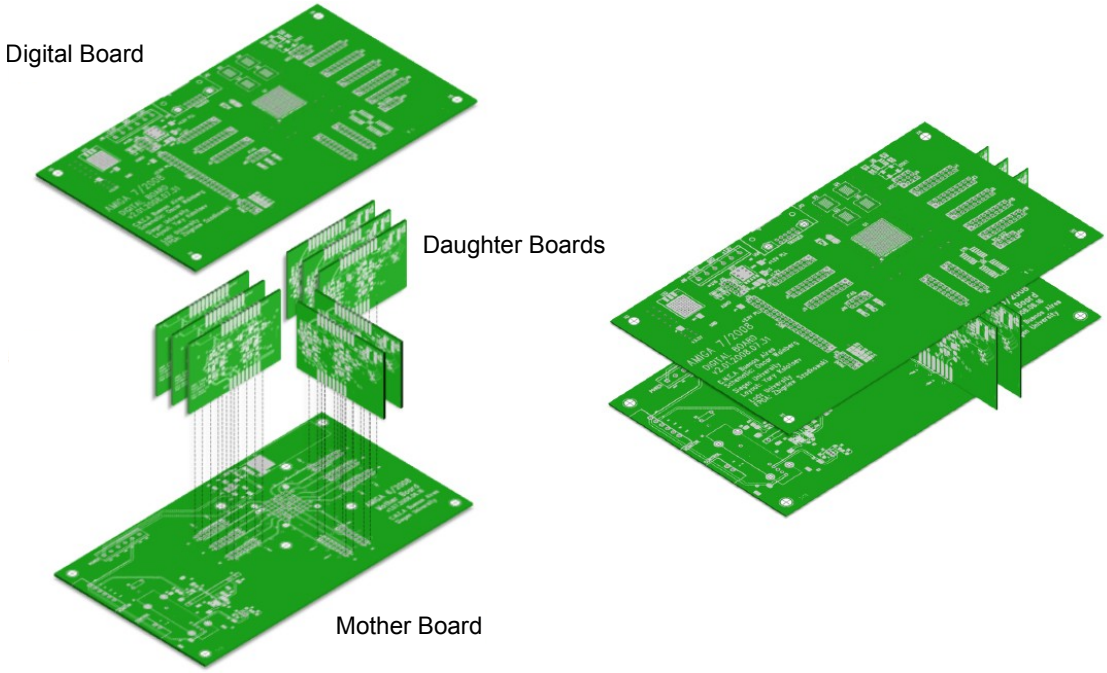

Abbildung 4.2: Darstellung von Mother Board, Digital Board und den Daughter Boards [Frö09]

Die Hauptaufgabe für das Digital Board ist die Auswertung der ankommenden 64 Signale. Dies geschieht mit einem FPGA (Field Programmable Gate Array), der die Signale in einem externen RAM zwischenspeichert und mit zusätzlichen Informationen im Falle eines Triggers an den Microcontroller weiterleitet.
Das ganze System ist dabei auf die ankommenden Luftschauer getriggert. Detektiert der zugehörige Wassertank auf der Oberfläche einen Luftschauer, so wird ein Triggersignal an den Myonzähler gesendet. Dieses wird dann über das Power Board an das Digital Board weitergeleitet und der FPGA beginnt mit der Auswertung der Signale.

4.1.5 Microcontroller Board

Zusammenfassend laufen die vom PMT erzeugten elektrischen Signale, die den detektierten Myonen im Szintillator entsprechen, zum Mother Board, wo sie auf die acht Daughter Boards aufgeteilt werden. Dort werden die ankommenden analogen Signale in digitale Signale umgewandelt und an den FPGA auf dem Digital Board übertragen, der sie dann bei entsprechendem Trigger auswertet und im externen RAM zwischenspeichert. Zusammen mit zusätzlichen Informationen sendet der FPGA im Anschluss die aufgenommenen Daten an den Microcontroller, welcher auf dem sogenannten *Microcontroller Board* platziert ist und die Schnittstelle zwischen FPGA und Basisstation realisiert. Der Microcontroller liest die Messdaten aus und leitet diese über das Power Board an die Oberflächenelektronik via CAN-Bus weiter. Von dort aus werden sie schließlich als Funksignal an die Basisstation gesendet.
Allerdings ist auch eine Kommunikation in anderer Richtung möglich, sodass beispielsweise der FPGA über den Microcontroller programmiert und konfiguriert werden kann. Der Microcontroller bildet also die Schnittstelle zwischen FPGA und der Außenwelt.
Eine weitere Aufgabe des Microcontrollers ist neben der Kommunikation zwischen FPGA und Basisstation die Überwachung und Kontrolle der Hochspannung des PMTs, die auf dem Mother Board erzeugt wird. Dazu werden Informationen über die Hochspannung versendet und Konfigurationsbefehle empfangen.

An dem aktuellen Design des Microcontrollers wird derzeit in Argentinien gearbeitet. Dort verwendet man die neueste Version des Microcontrollers[5], die zusätzliche Möglichkeiten zur schnelleren Kommunikation mit dem FPGA bereitstellt. Die aktuelle Planung sieht im abschließenden Design vor, dass der MCU (MicroController Unit) nicht auf einem eigenen Board (dem vorgestellten Microcontroller Board), sondern ebenfalls auf dem Digital Board platziert werden soll.

Da jedoch das Design aller anderen Boards feststeht und Prototypen gebaut wurden, können diese bereits auf ihre Funktionalität getestet werden. Dieses soll an der Universität Siegen geschehen. Zur Realisierung wird in Siegen die Vorgängerversion des MCU[6], der auf einem Development Board platziert ist, verwendet.

In den folgenden Kapiteln wird das Development Board zusammen mit dem verwendeten Microcontroller und den entwickelten Programmen zur Kommunikation mit dem FPGA vorgestellt.

[5]Development Board: TMS470 Kickstart Development Kit der Firma Texas Instruments.
[6]Development Board: PICDEM CAN-LIN 3 Development Kit der Firma Microchip.

5 Der Microcontroller

Die wesentliche Aufgabe des an der Universität Siegen verwendeten Microcontrollers besteht in der Realisierung der Kommunikation zwischen FPGA und dem PC, um die gesamte Hardware der AMIGA Ausleseelektronik auf ihre Funktionalität zu testen.
Bevor die einzelnen Boards für die Massenproduktion freigegeben werden, muss die Funktionsweise der einzelnen Boards und deren korrektes Zusammenspiel mit der gesamten Ausleseelektronik sichergestellt sein. Sogar bei hohen Temperaturschwankungen, die denen der Pampa in Argentinien entsprechen und in einer Klimakammer simuliert werden, muss die Weiterleitung der PMT Signale einwandfrei sichergestellt sein. Dieses kann jedoch nur überprüft werden, wenn man die Ausgabe kennt und mit den erwarteten Werten vergleichen kann. Dazu ist der Microcontroller notwendig, denn jener überträgt die vom FPGA aufgenommenen Daten an den PC, wodurch ein Vergleich erst möglich wird.

Allerdings ist beispielsweise für die Übertragung der Daten vom MCU an den PC eine serielle Schnittstelle, für die Programmierung des MCU eine Programmierschnittstelle und für die Zwischenspeicherung ein Speicher notwendig. Diese zusätzlichen Komponenten sind zusammen mit dem Microcontroller auf einen Development Board platziert.

5.1 Development Board

Das verwendete Development Board namens *PICDEM CAN-LIN 3* der Firma *Microchip* besitzt insgesamt drei Microcontroller, die untereinander via CAN- und LIN[1]-Bus kommunizieren können.
Die Hauptbestandteile des Development Boards sind in der folgenden Abbildung 5.1 veranschaulicht:

[1]Local Interconnect Network. LIN gehört ebenso wie CAN zu den Feldbussen und stellt im Vergleich zu CAN eine einfachere und kostengünstigere Kommunikation dar. Auch LIN wurde ursprünglich für die Automobilindustrie entwickelt.

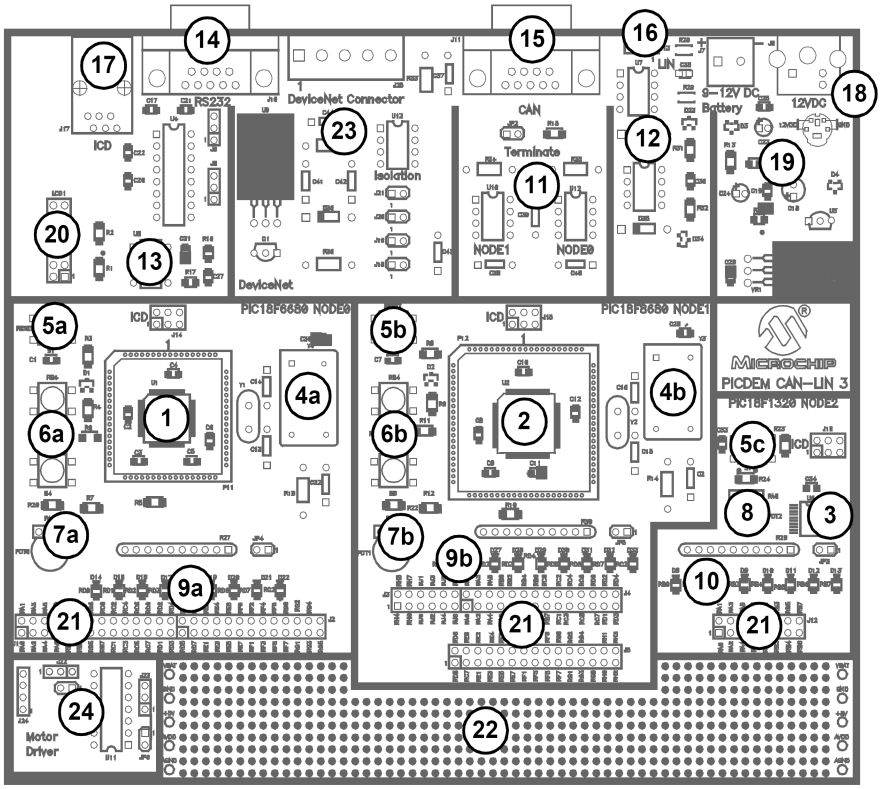

Abbildung 5.1: Schematische Übersicht des Development Boards [Mic03]

1. **Microcontroller Node0**: Hierbei handelt es sich um einen 64-Pin MCU des Typs *PIC18F6680*. Er ist der Hauptmirocontroller, der als einziger via serieller Schnittstelle RS-232 mit dem PC kommunizieren kann. Zusätzlich besitzt er ein integriertes ECAN (Enhanced CAN) Modul, kann über den LCD Konnektor (20) mit einem LCD Modul verbunden werden und besitzt einen optionalen Zugang zu einem externen EEPROM (13).

2. **Microcontroller Node1**: Dies ist ein 80-Pin MCU des Typs *PIC18F8680*. Er besitzt ebenfalls ein integriertes ECAN Modul und kann mit Node0 und jedem anderen externen CAN Modul über den CAN Konnektor (15) kommunizieren. Zusätzlich agiert er als LIN Master bei der Kommunikation via LIN-Bus mit dem MCU Node2 bzw. mit anderen Modulen über den optionalen LIN Konnektor (16).

3. **Microcontroller Node2**: Dies ist ein 20-Pin MCU des Typs *PIC18F1320*. Er agiert als LIN Slave bei der einzig möglichen Kommunikation mit Node1.

4. **Oszillatoren**: Node0 und Node1 besitzen drei teilweise konfigurierbare Oszillatoren (Kristall, RC und CAN), die die Arbeitsfrequenzen der MCUs vorgeben. Der CAN Oszillator ist mit einer Frequenz von 25 MHz voreingestellt.

Die maximale Taktfrequenz beträgt 40 MHz. Node2 verfügt nur über einen RC Oszillator ohne Konfigurationsmöglichkeit.

5. **Reset Push Button**: Jeder MCU kann über einen Reset Push Button zurückgesetzt werden.

6. **Push Buttons**: Zusätzlich besitzen die CAN-MCUs (Node0 und Node1) zwei Push Buttons, die einen digitalen Input simulieren.

7. **Potentiometer: Node0 (7a) und Node1 (7b)**: Mit den analogen Potentiometern der CAN MCUs werden digitale Inputs auf den eigenen Controllern simuliert und zusätzlich jeweils die Helligkeit der LEDs des anderen Controllers kontrolliert.

8. **Potentiometer: Node2**: Dies ist ein analoger Potentiometer zur Simulation digitaler Inputs für Node2.

9. **LED Bank: Node0 und Node1**: Jeder der CAN-MCUs besitzt seine eigene Bank von 9 LEDs. Davon zeigen acht den digitalen Status von dem I/O-Port $PortD$ und einer den Status des dritten I/O-Pins ($PinC2$) von $PortC$ an.

10. **LED Bank: Node2**: Node2 besitzt ebenfalls eine Bank mit acht LEDs, die den Status von $PortA$ und $PortB$ zeigen.

11. **CAN Transceiver**: Damit werden die digitalen Signale des CAN-Busses auf ein für Node0 und Node1 kompatibles Level transformiert.

12. **LIN Transceiver**: Hiermit werden die Hochspannungssignale des LIN-Busses auf ein für Node1 und Node2 kompatibles Niveau gewandelt.

13. **Externer EEPROM**: Optional kann Node0 über dieses Interface mit einem externen EEPROM kommunizieren.

14. **RS-232 Konnektor**: Dieser stellt eine standardisierte serielle Verbindung dar, die eine Kommunikation von Node0 mit dem PC ermöglicht.

15. **CAN Konnektor**: Dieser Konnektor erlaubt eine Verbindung von Node1 mit externen CAN Modulen.

16. **LIN Konnektor Pad**: Hier kann optional ein LIN Konnektor eingebaut werden.

17. **ICD Konnektor**: Über die ICD (In Circuit Debugger) Verbindung können die einzelnen MCUs programmiert werden. Zur Auswahl des entsprechenden Controller müssen die dazugehörigen Jumper gesetzt werden. In Abbildung 5.2, in der alle Jumperpositionen auf dem Board illustriert sind, sind dies der **3**. Jumper (J14) für Node0, der **5**. Jumper (J15) für Node1 und der **7**. Jumper (J16) für Node2.

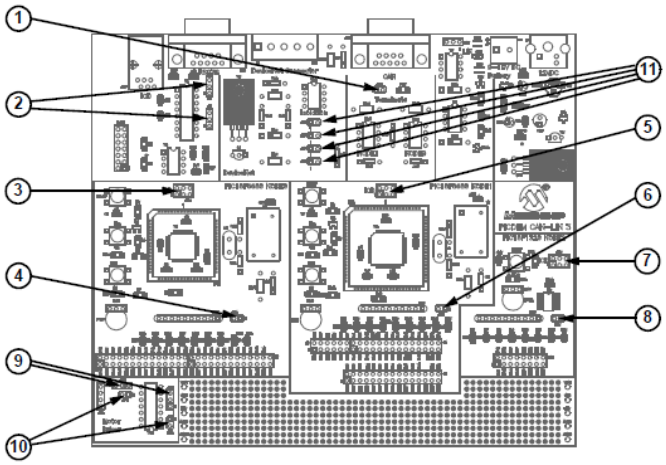

Abbildung 5.2: Jumperpositionen [Mic03]

18. **Power Konnektor**: Darüber hinaus wird das Development Board mit einer Spannung von 12 V_{DC} versorgt. Neben dem Netzteilanschluss existiert ein zusätzlicher Anschluss für eine Gleichspannungsquelle ($9 - 12$ V_{DC}), wobei beide gegen fälschliches Umpolen abgesichert sind. Spannungsregler auf dem Board stellen Spannungen von $+5$ V_{DC} für digitale und analoge Stromkreise zu Verfügung.

19. **Power LED**: Diese leuchtet auf, wenn das Board mit Spannung versorgt ist.

20. **LCD Konnektor**: Über den LCD Konnektor lässt sich optional ein Standard LCD Modul anbringen.

21. **I/O-Portpins**: Mit den I/O-Pins der einzelnen MCUs können Signale abgegriffen bzw. angelegt werden, wodurch eine direkte Kommunikation mit anderer Elektronik (FPGA) ohne Verwendung von CAN- oder LIN-Bus möglich wird.

22. **Freie Steckplätze**: Die freien Steckplätze können beliebig genutzt werden, um zusätzliche Elektronik auf dem Board zu platzieren.

23. **DeviceNet[2] Interface**: Dieser Bereich ist dafür vorgesehen ein DeviceNet Interface für Node0 zu installieren.

24. **H-Brücke[3]**: Hier kann ein H-Brücke zur Spannungskontrolle installiert werden.

[2]Ein auf CAN basierender zusätzlicher Feldbus.
[3]Eine H-Schaltung (auch H-Brücke) bezeichnet eine elektrische Schaltung, bei der die Grundform ein H darstellt.

5.1.1 Microcontroller (Node0)

Ausschließlich der Microcontroller Node0 ist in der Lage via serieller Schnittstelle mit dem PC zu kommunizieren und stellt daher den Hauptmicrocontroller dar. Aus diesem Grund wird auch nur dieser MCU für die Kommunikation mit dem FPGA verwendet. Der zweite MCU Node1, welcher Node0 sehr ähnlich ist (ohne serielle Schnittstelle), wird lediglich zur Simulation des FPGA als Kommunikationspartner verwendet und spielt somit eine untergeordnete Rolle. In diesem Kapitel wird deswegen nur der Hauptmicrocontroller im Detail vorgestellt.

Allgemein lässt sich ein Microcontroller als ein Halbleiterchip bezeichnen, der einen Prozessor mit bestimmten Peripherifunktionen, wie beispielsweise CAN-Bus und serielle Schnittstelle, vereint.

Das ist der Fall für Microcontroller der PIC18 Familie. Der verwendete PIC18F6680 besitzt diverse periphere Funktionen, die über die 64 Pins gesteuert werden können. In Abbildung 5.3 ist die Pinbelegung des 64 TQFP[4] PIC18F6680 dargestellt.

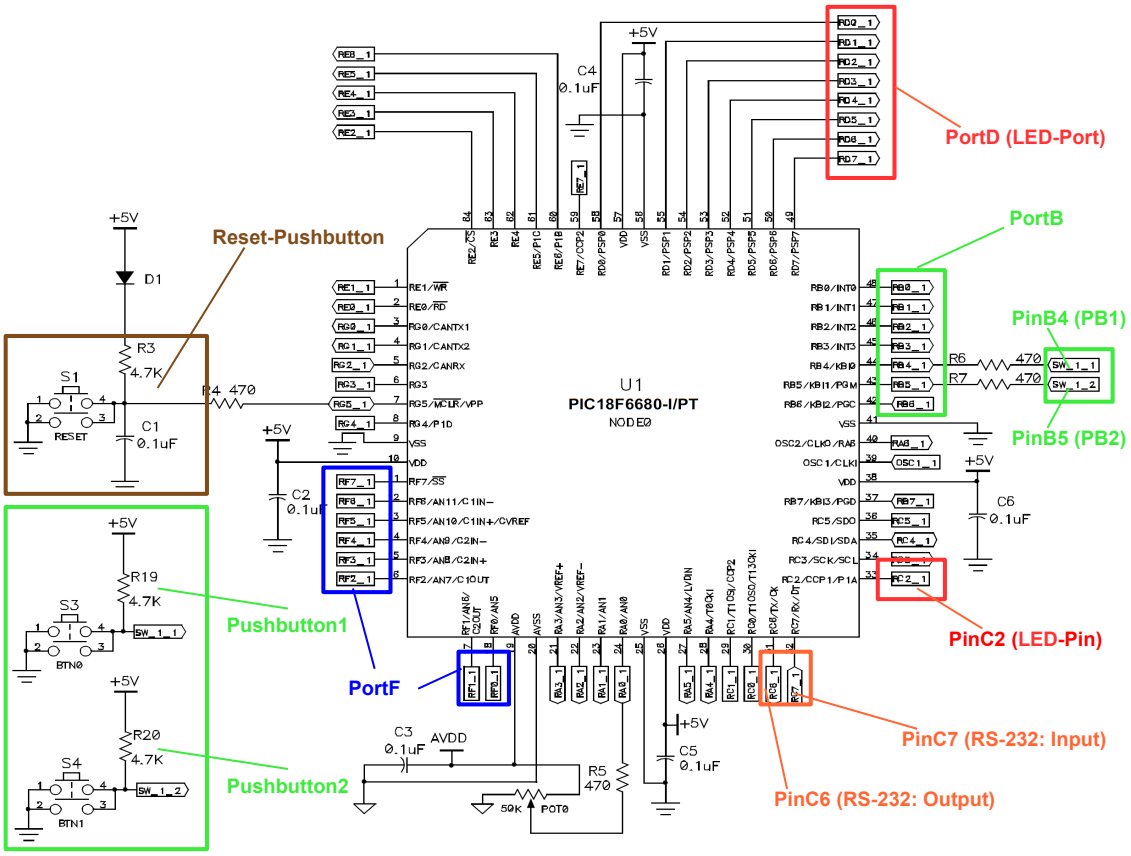

Abbildung 5.3: Pinbelegung von Microcontroller Node0 [Mic03]

[4]Thin Quad Flat Pack. Bezeichnung des Chipgehäuses.

Die für die entwickelten Programme verwendeten Pins und die entsprechenden Pushbuttons sind farbig markiert: Braun der Reset-Pushbutton, grün die zwei weiteren Pushbuttons, deren Input über die Pins $PinB4$ und $PinB5$ im Programm verwendet werden kann, grün der gesamte $PortB$ als digitalen I/O-Port, blau $PortF$, der optional[5] als digitaler I/O-Port verwendet werden kann, orange die zwei Pins $PinC6$ (Ausgang: MCU→PC) und $PinC7$ (Eingang: PC→MCU), über die die serielle Schnittstelle angesprochen wird, rot der digitale I/O-Port $PortD$ über den gleichzeitig acht LEDs gesteuert werden und schließlich - ebenfalls rot - $PinC2$, über den sich zusätzlich die neunte LED steuern lässt.

5.1.2 Serielle Schnittstelle (RS-232)

Sollen nun Daten über die dem Standard *RS-232*[6] entsprechende serielle Schnittstelle gesendet bzw. empfangen werden, so geschieht dies mit den vorhergehend genannten Pins $PinC6$ (Ausgang) und $PinC7$ (Eingang). Zum einen sind sie mit einer standardisierten DE9-Buchse[7] und zum anderen mit zwei I/O-Portpins, an denen zusätzlich die zu übertragenden Signale abgegriffen werden können, verbunden. Um nun die Datenblöcke seriell über eine Datenleitung zu übertragen, ist ein bestimmtes Timing einzuhalten, damit die Kommunikationspartner wissen, wann sie welche Daten auszulesen haben. In Abbildung 5.4 ist ein Timingdiagramm zu sehen, das beispielhaft den Aufbau und das Timing eines seriell zu übertragenden Datenblock illustriert.

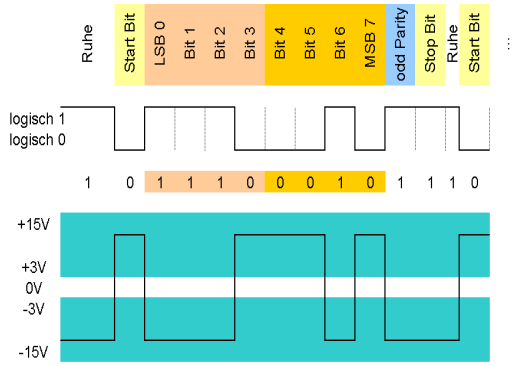

Abbildung 5.4: Timingdiagramm für eine serielle Datenübertragung [wikdeEIA]

Vor jedem Datenblock liegt stets der Ruhepegel an, der aus einer logischen 1 (entsprechender Pin wird high gesetzt) besteht. Dann folgt der Datenblock, der mit dem

[5] $PortF$ ist verbunden mit analogen peripheren Funktionen: Komparator In- und Outputs und Spannungsreferenzen. Allerdings kann er auch als digitaler I/O-Port verwendet werden, indem entsprechende Register gesetzt werden. Für die entwickelten Programme wurde $PortF$ ausschließlich als I/O-Port verwendet. Siehe dazu die Portinitialisierung bei den Beispielprogrammen im Anhang A auf Seite 65.

[6] Recommended Standard. Häufig auch EIA-232 (Electronic Industries Alliance) genannt.

[7] 9-polige D-Sub Steckverbindung. Benannt nach ihrer D ähnlichen Form. Siehe Bauteil 14 in Abbildung 5.1 auf Seite 30.

sogenannten *Startbit* (logische 0) beginnt. Anschließend kommen die eigentlichen Daten, die aus acht Bits (einem Byte) bestehen, wobei zunächst das LSB (Least Significant Bit) und zuletzt das MSB (Most Significant Bit) gesendet wird. In dem aufgeführten Beispiel beträgt das Datenwort: Binär: 01000111; Hex: 47; ASCII: *G*. Wahlweise folgt nach dem Datenwort ein sogenanntes *Paritätsbit*[8], welches zur Erkennung von Übertragungsfehlern verwendet werden kann, worauf allerdings in den entwickelten Programmen verzichtet wird. Abgeschlossen wird der Datenblock mit einer logischen 1, die als *Stoppbit* bezeichnet wird und gleichzeitig den Ruhepegel für den darauf folgenden Datenblock darstellt. Dabei spielt die Länge der Ruhezeit zwischen zwei Datenblöcken keine Rolle.
Die zeitliche Dauer eines jeden Bits ist jedoch genau von der eingestellten sogenannten *Baudrate*[9] abhängig. Um die Totzeit[10] zu verringern, wurde die für den PC maximal zu erreichende Bitrate von 115200 bit/s (Bitdauer: $8,6\ \mu$s) ausgewählt. Auf dem Microcontroller wird die identische Bitrate durch Verwendung entsprechender Delays im Programm eingestellt[11].

5.1.3 Speicherorganisation

Der PIC18F6680 (Node0) besitzt (ebenso wie der PIC18F8680 (Node1)) zwei Speicher. Zum einen ist auf dem MCU ein on-chip Programmspeicher und zum anderen ein on-chip Datenspeicher implementiert.
Der Programmspeicher ist ein sogenannter *Flash-Speicher* mit einer Kapazität von 64 kByte, in dem die Programme des MCU blockweise gespeichert werden. Ein Befehlswort ist 16 bit groß, sodass sich insgesamt 32768 Befehlsworte speichern lassen. Diese werden so lange gespeichert bis sie durch ein neues Programm überschrieben werden. Selbst nach Abschalten der Versorgungsspannung bleibt das gespeicherte Programm erhalten. Es handelt sich also um einen persistenten (nichtflüchtigen) Speichertyp.
Der Datenspeicher setzt sich aus einem 3328 Byte großen als *SRAM (Static Random Access Memory)* bezeichneten Speicher und einen 1024 Byte großen Speicher namens *EEPROM (Electrically Erasable Programmable Read-Only Memory)* zusammen. Beide Speicher werden – im Gegensatz zum Flash-Speicher – nicht blockweise beschrieben, sondern es lässt sich jede ihrer 8 bit großen Speicherzellen durch entsprechende Adressierung ansprechen. Das EEPROM ist ebenfalls ein persistenter Speichertyp, wohingegen der SRAM ein volatiler (flüchtiger) Speicher ist, dessen Daten nach Abschaltung der Versorgungsspannung verloren gehen.

[8] Es gibt drei Einstellungsmöglichkeiten des Paritätsbits: 1. Gerade („EVEN“) Parität, 2. Ungerade („ODD“) Parität und 3. Keine („No“) Parität.

[9] Ein Baud ist die Geschwindigkeit, wenn ein Symbol pro Sekunde übertragen wird. Bei der seriellen Schnittstelle RS-232 entspricht diese gerade der Bitrate, also der Anzahl der zu übertragenden Bits pro Sekunde.

[10] Zeit, die benötigt wird, um Daten vom MCU an den PC zu senden und in der keine neuen Daten gespeichert werden können.

[11] Siehe dazu die Unterfunktion: „sendtopc()“ bei den Beispielprogrammen im Anhang A auf Seite 65.

Da die Schreibzyklen des EEPROMs begrenzt sind, das Beschreiben im Vergleich zum SRAM relativ langsam ist (4 ms) und die Daten des FPGA lediglich zwischengespeichert werden müssen, wird auf die Verwendung des EEPROM verzichtet. Aus diesem Grund wird im Folgenden nur die Adressierung und der Aufbau des SRAMs genauer betrachtet:

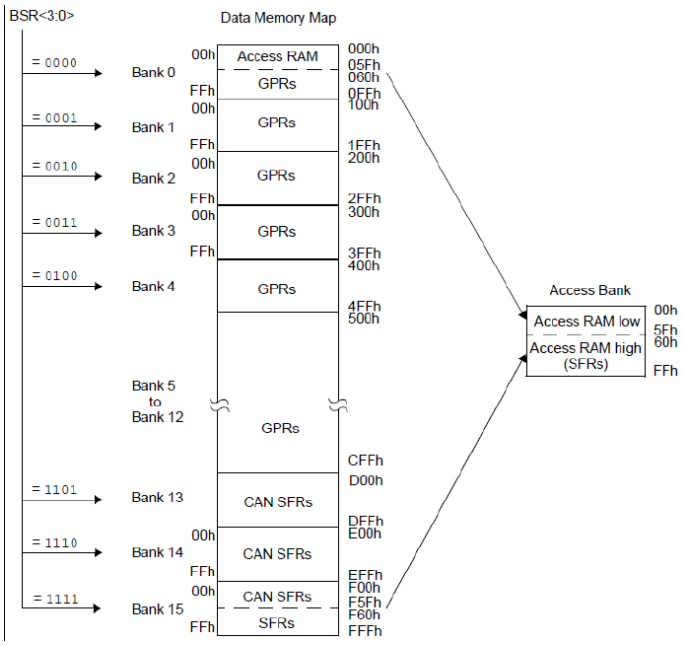

Abbildung 5.5: Speicherplan des SRAMs [Mic04]

Wie in Abbildung 5.5 zu sehen, unterteilt sich der 4096 Byte große SRAM in 16 Bänke mit je 256 Byte, welche direkt ausgewählt werden können (Bank Selection Register). Allerdings ist auch eine indirekte Adressierung möglich, wo jedes Byte durch eine 12 bit Adresse angesprochen wird. Bei einer direkten Adressierung kann also eine Bank direkt angesprochen werden, wohingegen bei der indirekten Adressierung eine Bank nur indirekt angesprochen wird, da diese sich nur auf die einzelnen Bytes konzentriert. Die Aufteilung in Bänke ist also für die indirekte Adressierung ohne Bedeutung.
Der SRAM beinhaltet zum einen sogenannte *Special Function Register*, die für die Nutzung des MCU und seine peripheren Funktionen reserviert sind, und zum anderen sogenannte *General Purpose Register*, die für die Datenspeicherung durch den Benutzer vorgesehen sind. Die GPRs haben insgesamt eine Größe von 3328 Byte. Sie bilden den frei verfügbaren Datenspeicher.
Zusätzlich ist noch ein spezieller Bereich (Hex: $000 - 05F$ und $F60 - FFF$) vorhanden, der als *Access Bank* bezeichnet wird. Dieser Bereich stellt eine Erweiterung dar, die sehr nützlich für Programme ist, die in der Programmiersprache *C* verfasst wurden. Der Grund dafür ist, dass der *C*-Compiler[12] diesen Bereich beispielsweise für die Speicherung globaler und lokaler Variablen nutzt.

[12]Der *C*-Compiler wandelt *C*-Quellcode in die Maschinensprache Assembler um.

6 Entwickelte Programme

In diesem Kapitel sollen nun die eigenständig entwickelten Programme des Microcontrollers in chronologischer Reihenfolge vorgestellt werden. Dabei wird auf die Vorstellung der ersten einfach strukturierten Programme, die lediglich zum Verständnis des MCU angefertigt wurden, verzichtet.

Zu Beginn wird das erste Handshakeprogramm präsentiert. Danach folgen die stetigen Erweiterungen, die letztendlich zum abschließenden Handshakeprogramm zwischen FPGA und MCU führen, welches für den finalen Hardwaretest verwendet werden soll.

Der für jedes Programm notwendige Versuchsaufbau wird stets anhand eines Blockdiagramms illustriert. Ebenso werden die an den PC übertragenen Daten tabellarisch aufgeführt. Im Einzelnen sind zur Simulation des PMTs externe Signale notwendig. Um die Struktur und zeitliche Entwicklung der Signale nachvollziehen zu können, werden diese mit Hilfe eines Oszilloskops grafisch dargestellt.

Eine wichtige Rolle bei dem verwendeten Handshake zwischen FPGA und MCU besitzt das Timing der Signale. Da der FPGA und der MCU mit verschiedenen Taktfrequenzen arbeiten, müssen zusätzliche Handshakeleitungen verwendet werden, um die Kommunikation der verschiedenen Bauteile miteinander zu synchronisieren. In den entwickelten Programmen legt der FPGA Signale auf die Handshakeleitung und der MCU liest nur dann Daten aus, wenn er auf der Handshakeleitung ein abgestimmtes Signal detektiert. Um eine maximale Geschwindigkeit der Datenübertragung zu erreichen, müssen die Signallängen möglichst kurz gehalten werden. Allerdings muss gleichzeitig gewährleistet sein, dass der Microcontroller genügend Zeit zum Auslesen und Verarbeiten der Daten hat, damit keine Daten verloren gehen. Es muss also ein Kompromiss gefunden werden, der die genaue Kenntnis über die Struktur und das Timing der Signale voraussetzt. Aus diesem Grund werden neben den simulierten PMT Signalen auch die verwendeten Handshakesignale grafisch mit dem Oszilloskop dargestellt, so dass man auch hier die Struktur und zeitliche Entwicklung der Signale nachvollziehen kann.
Zusätzlich wird bei einigen Programmen anhand von Simulationen des FPGA Programms und erstellten Timingdiagrammen das Zusammenspiel zwischen anliegenden Datenwörtern und den Handshakesignalen illustriert.

6.1 Programm 1: „Comparison“

Das erste Handshakeprogramm dient dem Vergleich von Daten, die zum einen von Node0 und zum anderen von Node1 generiert werden und wird daher *Comparison* genannt. Alle für dieses Programm benötigten Komponenten sind in Abbildung 6.1 aufgeführt.

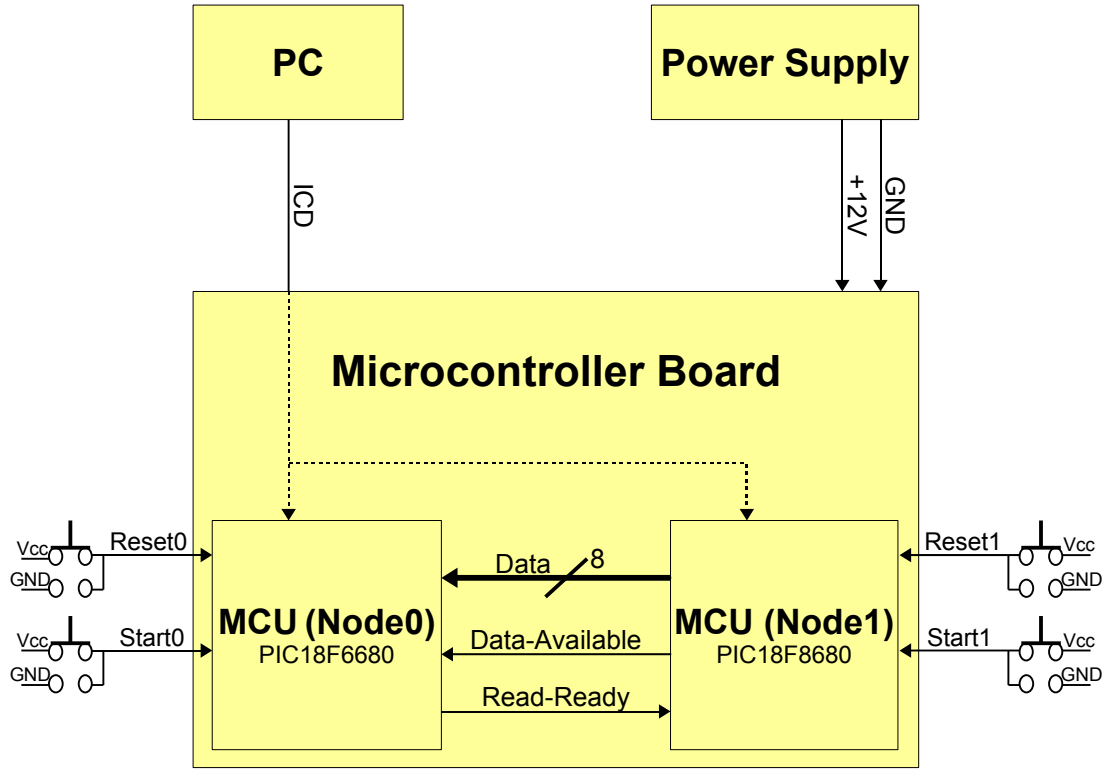

Abbildung 6.1: Schematischer Aufbau: „Comparison“

Sie beinhalten die beiden Microcontroller Node0 und Node1, die auf dem Microcontroller Board platziert sind, den PC, über den die beiden MCUs via ICD Verbindung programmiert werden, und den Power Supply, welcher die Stromversorgung des Boards sicherstellt. Beide Microcontroller werden über einen Pushbutton ($PinB4$)[1] gestartet und über den Reset-Pushbutton zurückgesetzt. Die Daten werden über acht Datenleitungen übertragen ($PortF$) und die Kommunikationssteuerung zwischen den beiden MCUs wird über zwei Handshakeleitungen realisiert.
Wie an der Pfeilrichtung der Datenleitungen erkennbar ist, dient Node1 als Sender und Node0 als Empfänger der Daten. Sind nun Daten für Node0 verfügbar, so sendet Node1 das sogenannte *Data-Available* Signal ($PinB1$) an Node0. Dieser liest nun die Daten aus und sendet im Anschluss unmittelbar das als *Read-Ready* bezeichnete

[1] Die Pinbelegung von Node1 entspricht der Pinbelegung von Node0. Siehe Kapitel 5.1.1 auf Seite 33.

Signal ($PinB3$) an Node1 zurück, der anschließend neue Daten zu Verfügung stellt und sich somit der Prozess wiederholt.
Der eigentliche Vergleich wird nun vom Hauptmicrocontroller Node0 durchgeführt. Er vergleicht die empfangenen Daten mit den zum entsprechenden Zeitpunkt selbst generierten Daten und schaltet die neunte LED ($PinC2$) bei Übereinstimmung ein. Stimmen die Daten nicht überein so erlischt die LED.
Zusätzlich werden die Daten der einzelnen MCUs auf den jeweiligen acht LEDs ($PortD$) dargestellt und entsprechende Delays[2] in das Programm eingebaut, sodass eine Überprüfung durch den Anwender möglich ist.

Als Test des Programms wurden beispielsweise folgende Daten an Node0 nacheinander übertragen: Hex: FF, $F2$, 15 und folgende Daten selbst generiert: Hex: FF, $F0$, 15. Die neunte LED leuchtete dabei nur für das erste und letzte Datenword, was der Erwartung entspricht.

[2] Verzögerungen im Programm. In einer Unterfunktion wird eine entsprechende Anzahl von NOP-Befehlen gesetzt. Bei einem NOP-Befehl wartet der MCU einen Rechenschritt (160 ns). Siehe dazu die Unterfunktion: „delay-bit()“ bei den Beispielprogrammen im Anhang A auf Seite 65.

6.2 Programm 2: „Check“

Eine Erweiterung von Comparison ist das entwickelte Programm namens *Check*, wobei ebenfalls empfangene Daten anhand von selbst generierten Daten überprüft werden. Der Versuchsaufbau (Abbildung 6.2) und das Handshakeprotokoll sind dabei identisch.

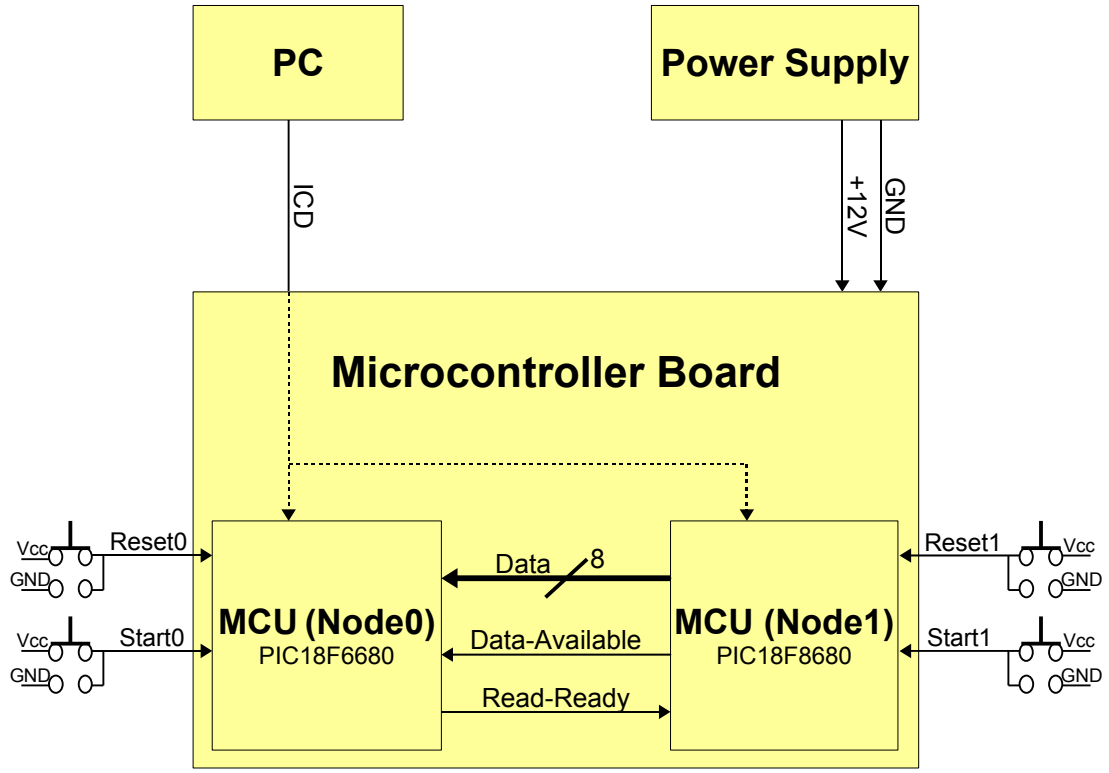

Abbildung 6.2: Schematischer Aufbau: „Check“

Allerdings wird nun anstelle von drei fixen Datenwörtern ein 8 bit breiter Counter[3] (Dez: $0 - 255$; Hex: $00 - FF$) verwendet, der in einer Endlosschleife läuft und somit eine Überprüfung der Hardware von beliebiger Dauer ermöglicht.
Es ist sowohl ein Counter in Node0 als auch in Node1 implementiert. Beide Counter starten mit dem Wert null.
Der Verlauf der Kommunikation sieht also wie folgt aus: Zu Beginn werden beide MCUs mit den entsprechenden Pushbuttons ($PinB4$) gestartet. Anschließend sendet Node1 seinen Counterwert über die acht Datenleitungen ($PortF$) an Node0. Dieser liest den Wert aus und vergleicht ihn mit seinem eigenen Counterwert, nachdem er das Data-Available Signal ($PinB1$) empfangen hat. Danach sendet Node0 das Read-Ready Signal ($PinB3$) zurück, beide Counter zählen hoch und das nächste Datenword wird gesendet. Stimmen die Datenwörter nicht überein, leuchtet die

[3]Zähler, programmiert in einer FOR-Schleife. Nach einem Überlauf startet der Counter wieder bei Null.

neunte LED auf($PinC2$) und das Programm bricht ab. Da die Counterwerte auch an den acht LEDs ($PortD$) angezeigt werden, lässt sich nach dem Abbruch die Abweichung der Datenwörter erkennen.

Programmiert man beispielsweise eine Abweichung in den Counter von Node1 an einer bestimmten Stelle ein, so bricht das Programm wie erwartet an dieser Stelle ab, die neunte LED leuchtet auf und man erkennt die voneinander abweichenden Werte an den jeweiligen acht LEDs.

6.3 Programm 3: „Check_FPGA“

Das Ziel der hier beschriebenen Programme ist die Realisierung der Kommunikation zwischen FPGA und dem Hauptmicrocontroller Node0. Node1 diente in den vorangegangenen Programmen lediglich zur Simulation des FPGA, um das geschriebene Programm zu testen und zu verbessern.
Da das Programm Check erfolgreich getestet werden konnte, kann nun der FPGA anstelle von Node1 als Kommunikationspartner benutzt werden. Allerdings sind dafür einige Anpassungen nötig, die in dem Programm namens *Check_FPGA* zusammengebracht werden.

Die Programmierung des FPGA ist nicht Bestandteil dieser Arbeit. Für diesen Versuchsaufbau wurde das FPGA Programm im Rahmen einer Bachelorarbeit [Neu08] entwickelt und getestet.

Das Digital Board mit dem FPGA, das Power Board, PC und Power Supply bilden neben dem Microcontroller Board mit dem MCU die Hauptbestandteile dieses Versuchsaufbaus und sind in Abbildung 6.3 dargestellt.

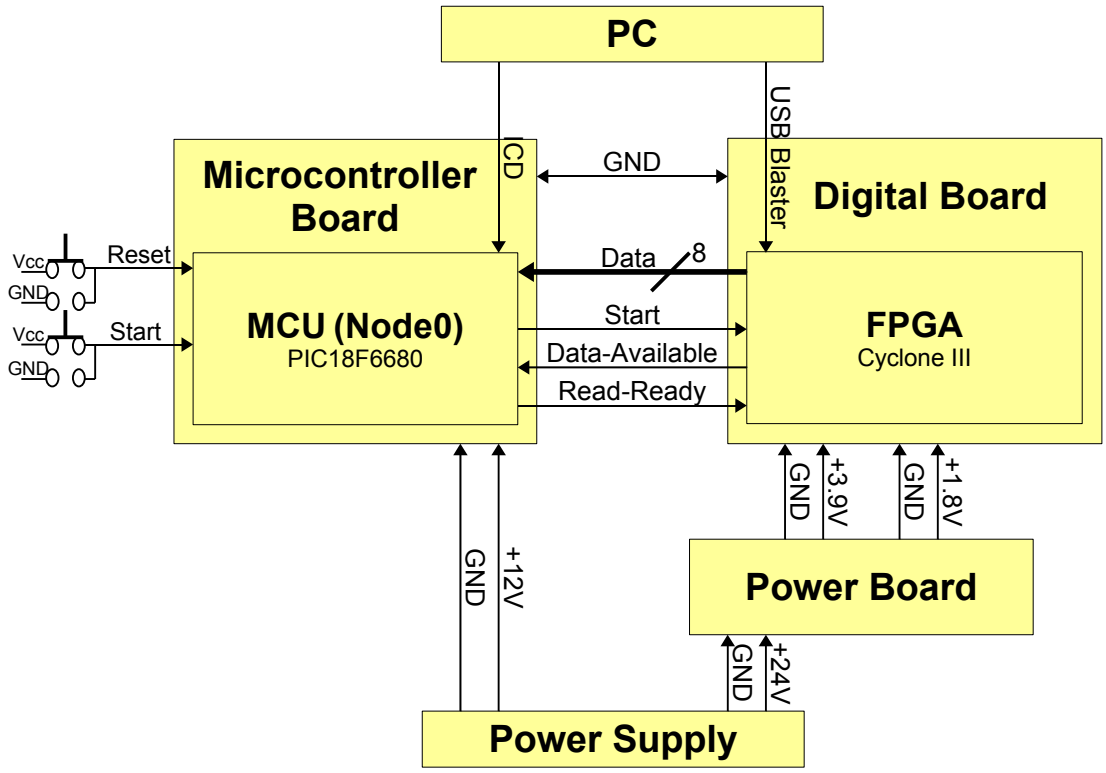

Abbildung 6.3: Schematischer Aufbau: „Check_FPGA“

Der FPGA mit der Bezeichnung *Cyclon III* ist wie bereits erwähnt auf dem Digital Board platziert und kann nur verwendet werden, wenn er mit genau definierten

Spannungen versorgt wird (GND, $+1,8$ V und GND, $+3,9$ V). Diese werden durch das Power Board, welches wiederum seine Betriebspannung (GND, $+24$ V) von einem Power Supply erhält, zu Verfügung gestellt. Programmiert wird der FPGA mit dem sogenannten *USB Blaster*, der eine USB Verbindung zwischen PC und FPGA herstellt.
Weiterhin ist in dem Versuchsaufbau neben den bekannten acht Datenleitungen und den zwei Handshakeleitungen Data-Available und Read-Ready eine zusätzliche Verbindung namens *Start* zu erkennen. Diese Verbindung realisiert eine dritte Handshakeleitung ($PinB0$), mit der der FPGA gestartet wird. Das Startsignal wird direkt zu Beginn des Microcontrollerprogramms gesendet und besitzt – ebenso wie die anderen Handshakesignale – eine genau definierte Länge von 8 μs[4]. Grund für die genau definierte Länge der Signale ist die Problematik des Timings. Der FPGA arbeitet mit Clocks, also mit exakt definierten Taktsignalen und entsprechender Taktfrequenz. Würden die Signaldauern beispielsweise variieren, so bestünde die Möglichkeit, dass der FPGA in einem Intervall ein Signal mehrfach detektiert und die Datenübertragung wäre fehlerhaft. Die vergleichsweise lange Periodendauer von 8 μs wurde nur zu Testzwecken gewählt. Da zusätzlich noch Delays verwendet werden, um das Hochzählen des Counters anhand der LEDs nachvollziehen zu können, stellt diese Wahl des Timings sicherlich nicht das Minimum dar.
Neben der Timing-Problematik, spielt noch die Wahl des Ground Levels eine wichtige Rolle. Da die Spannungen für den FPGA von dem Power Board zu Verfügung gestellt werden, besitzt das Digital Board einen anderen Ground Level als das Microcontroller Board. Werden die Boards nur mit den Kommunikationsleitungen verbunden, werden unerklärliche Signale detektiert. Um dies zu vermeiden, ist eine zusätzliche GND Verbindung zwischen den Boards und deren Ground Level erforderlich, sodass das gesamte System nur über einen einzigen gemeinsamen Ground Level verfügt.

Werden nun beispielsweise Abweichungen in den Counter von FPGA bzw. MCU programmiert, so stoppt das Programm an der entsprechenden Stelle und die neunte LED leuchtet auf. Ohne programmierte Abweichung läuft das Programm auf unbestimmte Zeit weiter und es lässt sich ein Langzeittest der Elektronik durchführen. Auf diese Art und Weise wurde ein erfolgreicher Langzeittest von 24h durchgeführt, der keine Fehler aufzeigte.

[4] 8 μs entsprechen 50 Rechenzyklen ($50 \cdot 160$ ns). Ein Rechenschritt wird benötigt, um den entsprechenden Pin high zu setzen, danach folgen 49 NOP-Befehle. Die Dauer wurde experimentell mit dem Oszilloskop bestätigt.

6.4 Programm 4: „Check_Gray“

Eine zusätzliche Erweiterung des Programms Check ist das Programm namens *Check_Gray* mit dem in Abbildung 6.4 dargestellten Versuchsaufbau.

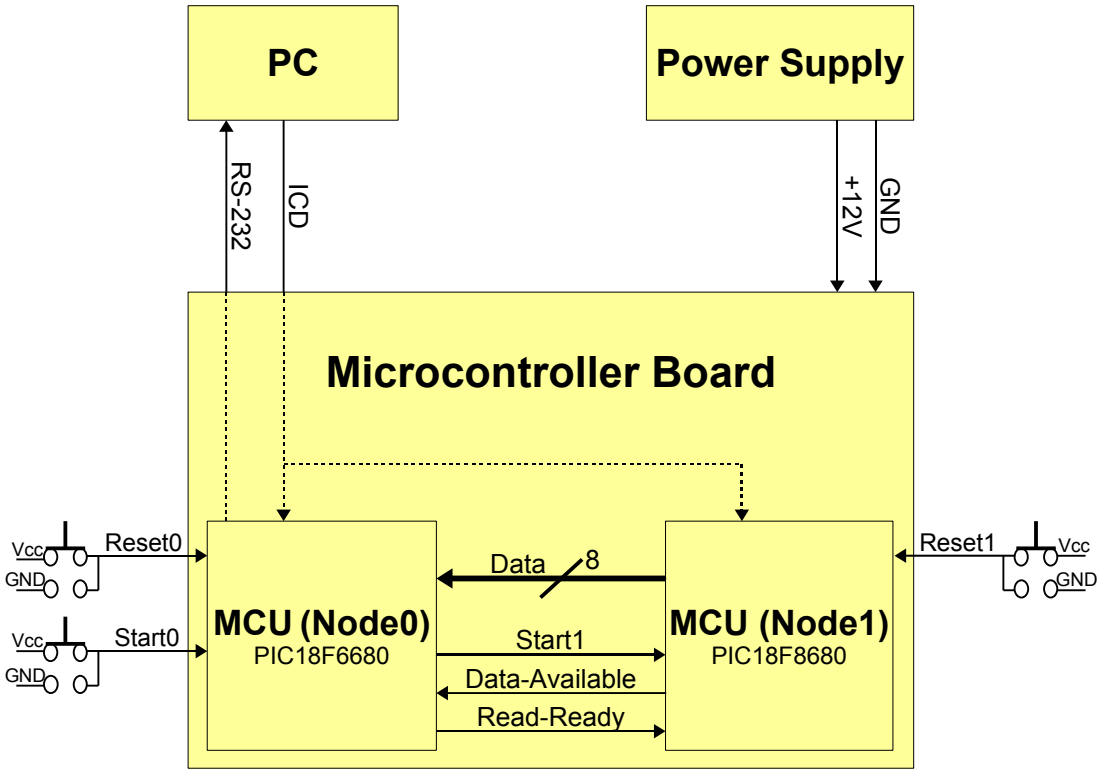

Abbildung 6.4: Schematischer Aufbau: „Check_Gray“

Im Wesentlichen gibt es hier drei Erweiterungen. Zum einen werden die Daten an den PC ausgegeben, zum anderen wird der letzte Datenstrom vor dem Auftreten einer Abweichung zwischengespeichert. Ferner wird anstelle eines Counters ein sogenannter *Gray-Code*[5] verwendet.

Ein Auszug aus einem 8 bit breiten Gray-Code mit den 256 sich ergebenden Kombinationsmöglichkeiten ist in Tabelle 6.1 zu sehen.

[5]Benannt nach dem Physiker Frank Gray handelt es sich um eine symmetrische Codierungsart, bei der sich benachbarte Werte jeweils nur in einem Bit unterscheiden.

00000000	00000001	00000011	00000010
00000110	00000111	00000101	00000100
00001100	00001101	00001111	00001110
00001010	00001011	00001001	00001000
	...		
10001000	10001001	10001011	10001010
10001110	10001111	10001101	10001100
10000100	10000101	10000111	10000110
10000010	10000011	10000001	10000000

Tabelle 6.1: Gray-Code (Binär)

Der Vorteil bei der Verwendung eines solchen Codes ist, dass bei aufeinander folgenden Werten jeweils nur ein Bit geändert wird und man auf diese Weise genau nachvollziehen kann, welche Datenleitung einen möglichen Defekt aufweist. Generiert wird der Gray-Code aus einem Binärcode durch folgende Anweisung im Quellcode:

$$g \;=\; b \,\hat{}\, (b \;>>\; 1) \qquad , \tag{6.1}$$

wobei b für den Binärcode, $>>$ für den bitweisen Rechts-Shift, $\hat{}$ für eine XOR-Verknüpfung[6] und g für die gewünschte Zahl im Gray-Code steht.
Startet man bei null, erhöht den Wert von b jeweils um eins und wandelt jedes b in den entsprechenden Gray-Code g um, so erhält man den in Tabelle 6.1 zu sehenden Auszug eines 8 bit breiten Gray-Codes.

Sobald die Kommunikation von Node0 gestartet wird, indem jener das Startsignal ($PinB0$) (analog zur Kommunikation mit dem FPGA) an Node1 sendet, übermittelt dieser seine Daten unter Verwendung des bekannten Handshakeprotokolls an Node0, der sie unmittelbar vergleicht. Stimmen die Daten überein, so wird der richtige Wert zwischengespeichert ($PortE$). Stimmen die Werte nicht überein, so sendet Node0 zunächst das letzte korrekte Datenwort aus dem Zwischenspeicher (zweimal) und anschließend die zu vergleichenden Daten, die eine Abweichung aufweisen, an den PC.

In Tabelle 6.2 ist ein Beispiel für eine Ausgabe an den PC dargestellt.

10000101	10000101
10000111	10101010

Tabelle 6.2: Ausgabe an PC (Binär): „Check_Gray“

Hier wurde in das Programm von Node0 eine Abweichung (Binär: 10101010) für das 251. Datenwort (Binär: 10000111) programmiert.

[6] Bitweise Addition, die Eins ergibt, wenn beide Summanden unterschiedlich sind, und Null ist, wenn beide Summanden gleich sind. Bsp.: $0 \wedge 0 = 0$, $1 \wedge 0 = 1$, $0 \wedge 1 = 1$ und $1 \wedge 1 = 0$.

Der Vorteil bei diesem Programm ist, dass die maximale Rechengeschwindigkeit der Bauteile gewählt werden kann, da keine Delays, die den Ablauf des Programms und somit die Kommunikation verlangsamen, notwendig sind, um eventuelle Abweichungen zu detektieren.

Leider konnte dieses Programm nicht mit dem FPGA als Kommunikationspartner getestet werden. Ursache war ein plötzlich aufgetretener Defekt auf dem Digital Board, der einige Zeit zur Reparatur in Anspruch nahm.

In der Zwischenzeit wurde folgendes Programm entwickelt, das durch die Verwendung eines größeren Zwischenspeichers (SRAM) eine noch schneller Kommunikation ermöglicht. Dabei wird auf die Vorstellung der vorweg entwickelten Programme, die lediglich zum Verständnis des SRAMs angefertigt wurden, verzichtet.

6.5 Programm 5: „Test(16)“

Basierend auf dem Programm Check bildet das Programm namens *Test(16)* mit seinen Neuerungen, die anhand des in Abbildung 6.5 dargestellten Versuchsaufbaus nachvollzogen werden können, die Grundlage für den Hardwaretest.

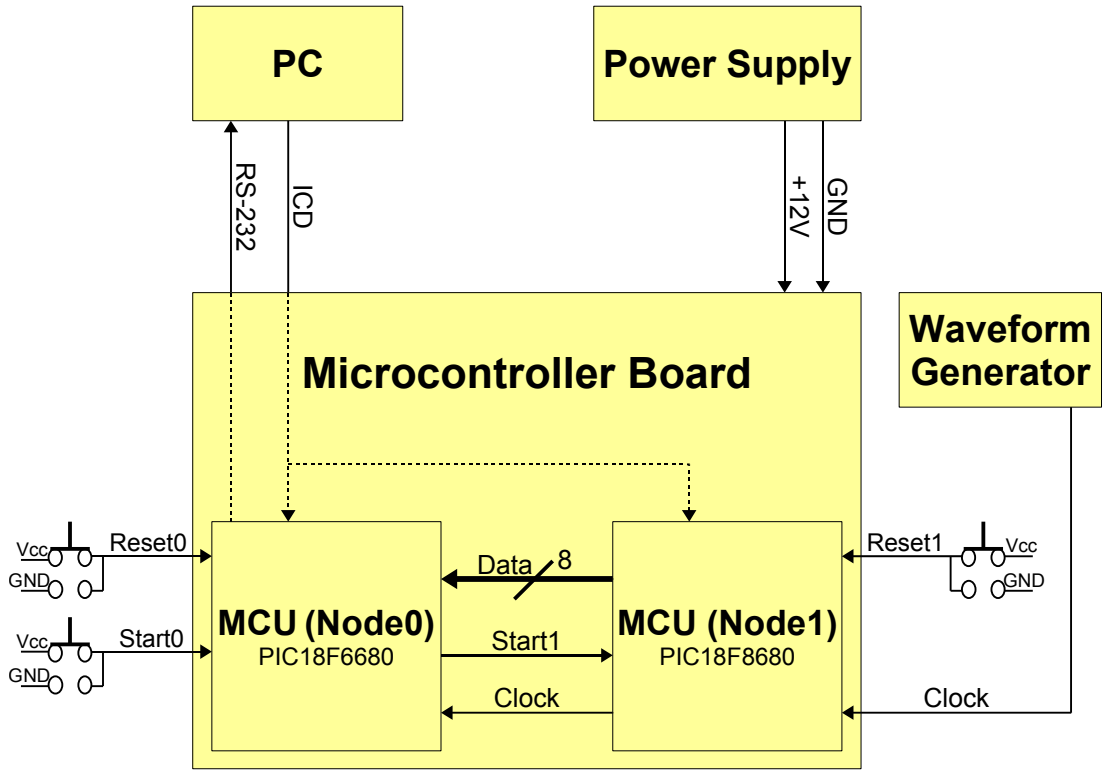

Abbildung 6.5: Schematischer Aufbau: „Test(16)“

In Vorbereitung auf die Kommunikation mit dem FPGA werden im konkreten Fall 16 Datenwörter von Node1 nacheinander ausgelesen und zwischengespeichert. Nachdem alle 16 Byte gespeichert wurden, werden diese über die serielle Schnittstelle an den PC übermittelt. Dort werden sie anschließend ausgewertet und auf Richtigkeit überprüft. Der Zwischenspeicher besteht dabei aus einem 16 Byte großen Array[7], das automatisch in der Access Bank[8] des SRAMs angelegt wird. Die Daten, die Node0 von Node1 erhält, entstammen einem 8 bit breiten Counter, der in Node1 implementiert ist.

Wie man dem Versuchsaufbau entnehmen kann, ist neben der bekannten Handshakeleitung Start1 ($PinB0$) und den Datenleitungen ($PortF$) nur noch eine Verbindung namens *Clock* ($PinB1$) erkennbar. Dieses Signal simuliert die Clock des

[7]Siehe dazu den Quellcode für dieses Programm im Anhang A.1 auf Seite 65.
[8]Siehe Kapitel 5.1.3 auf Seite 35.

FPGA, die hier als Data-Available Signal zu verstehen ist. Node0 liest also nur Daten aus, wenn das Signal high ist. Auf das Read-Ready Signal wurde verzichtet, um die Geschwindigkeit weiter zu optimieren.
Da somit die Empfangsbestätigung von Node0 wegfällt, spielt das Timing der Signale eine noch wichtigere Rolle. Aus diesem Grund wurde dem Versuchsaufbau ein externer Waveform Generator hinzugefügt. Über jenen wird die Clock modifiziert und an Node1 übermittelt. Node1 leitet in Folge das Signal an Node0 weiter ohne Änderungen daran vorzunehmen.
In Abbildung 6.6 ist das verwendete Signal zu sehen, das mit Hilfe eines Oszilloskops vermessen wurde.

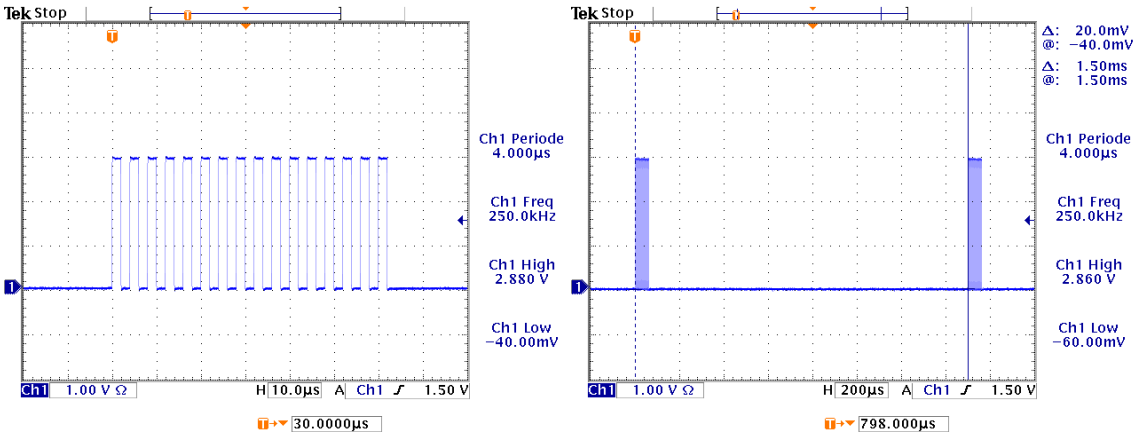

Abbildung 6.6: Darstellung der Clock: „Test(16)“

Der Waveform Generator besitzt ebenso wie der FPGA einen sogenannten *Burst* Modus, mit dem man die Anzahl der Pulse und die Periode, mit der sich diese Anzahl wiederholt, einstellen kann. Da 16 Datenwörter auszulesen sind, werden folglich 16 Pulse eingestellt, wie man in dem linken Oszilloskopbild erkennen kann. Das rechte Bild zeigt die zugehörige Burstperiode, die mit Hilfe von zwei Messbalken auf dem Bildschirm des Oszilloskops vermessen wurden und rechts oben bei der Spannungs- und Zeitdifferenz der Balken angezeigt wird. Dabei entspricht die Ruhephase nach den 16 Clocks der Zeit die notwendig ist, um die Daten an den PC zu senden, also genau der Zeit, in der keine weiteren Datenwörter mehr empfangen werden können. Am unteren Rand eines jeden Oszilloskopbildes können die Skalaeinteilung (Spannung vertikal, zeitlicher Verlauf horizontal) und die Triggerkonditionen nachvollzogen werden. Getriggert wird stets auf den ersten Puls. Zusätzlich werden rechts die Periode bzw. Frequenz des Signals und die Spannungspegel angezeigt. Die gemessenen Pegel nehmen Werte von 0 V (Low Level) und ca. +3 V (High Level) an, die der Spannung entsprechen, die auch an den einzelnen Pins ausgegeben wird.

Für die maximale Geschwindigkeit der Kommunikation ist die minimale Clockperiode (links) und die minimale Burstperiode (rechts) zu ermitteln. Dazu kann zunächst bei fester und relativ großer Burstperiode die Clockperiode soweit verringert werden, bis die bekannte Ausgabe an den PC (Counter) Abweichungen aufweist. So lässt sich die minimale Clockperiode von 4 μs, also eine maximale Clockfrequenz von 250 kHz (links), bestimmen. Node0 braucht also 4 μs, um ein Datenwort auszulesen und zwischenzuspeichern. Da nun die minimale Clockperiode bekannt ist, kann dieser Wert fixiert und die Burstperiode soweit verringert werden bis die Ausgabe an den PC erneut Abweichungen aufweist. Somit kann die minimale Burstperiode von $1,5$ ms, die einer maximalen Burstfrequenz von 666 Hz (rechts) entspricht, ermittelt werden. Für den Datentransfer der 16 Bytes an den PC sind folglich ca. $1,4$ ms (1500 μs - $16 \cdot 4$ μs = 1436 μs) erforderlich.
Berechnet man die Zeit für den Datentransfer anhand der maximalen Bitrate von 115200 bit/s, so ergibt sich folgender Wert: $16 \cdot 8$ bit / (115200 bit/s) = $1,11$ ms. Berücksichtigt man nun noch die zusätzlichen Rechenschritte, die beispielsweise notwendig sind, um die entsprechenden Funktionen im Programm aufzurufen, dann stimmt dieser Wert gut mit der Messung überein.

Um zu verhindern, dass während eines Clockpulses ein Datenblock zweimal ausgelesen wird, ist es unabdingbar das Zusammenspiel zwischen anliegenden Daten und dem Clocksignal genau zu kennen. Zur Verdeutlichung ist in Abbildung 6.7 ein Timingdiagramm der Daten und der Clock dargestellt.

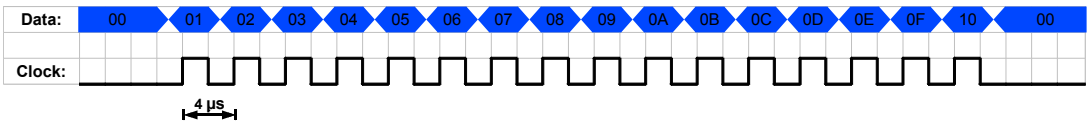

Abbildung 6.7: Timingdiagramm von Daten und Clock: „Test(16)“

Man erkennt, dass die Clock stets innerhalb eines neuen Datenworts auf dem High Level liegt. Dieser Wechsel (low → high) wird vom Microcontroller registriert und dieser beginnt unmittelbar mit dem Auslesen der Daten. Auf diese Art und Weise ist das korrekte Auslesen der Daten sichergestellt.

Mit den ermittelten maximalen Frequenzen, werden die in Tabelle 6.3 illustrierten Daten an den PC ausgegeben.

01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10
11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	20
21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F	30
31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F	40
41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	50
51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F	60
61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F	70
71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F	80
81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F	90
91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F	A0
A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF	B0
B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF	C0
C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF	D0
D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF	E0
E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF	F0
F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	FF	00
01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10

Tabelle 6.3: Ausgabe an PC (Hex): „Test(16)“

Wie man sieht, stimmen die einzelnen Zeilen, die einer Ausgabe der 16 Datenwörter entsprechen, genau mit den erwarteten Daten des endlosen 8 bit Counters überein.

6.6 Programm 6: „Test_FPGA_extTrigger(16)“

Da das Programm Test(16) erfolgreich getestet wurde, kann nun der FPGA als Kommunikationspartner verwendet werden. Mit dem Programm namens *Test_FPGA_extTrigger(16)* wird die Kommunikation zwischen FPGA und MCU unter der Verwendung eines externen Triggersignals realisiert.

Die Programmierung des FPGA wurde hier von dem wissenschaftlichen Mitarbeiter der Universität Siegen Yury Kolotaev übernommen. Auch in Zusammenarbeit mit ihm wurde dieser Versuch aufgebaut und die Kommunikation zwischen FPGA und MCU getestet.

Der Versuchsaufbau kann anhand von Abbildung 6.8 nachvollzogen werden.

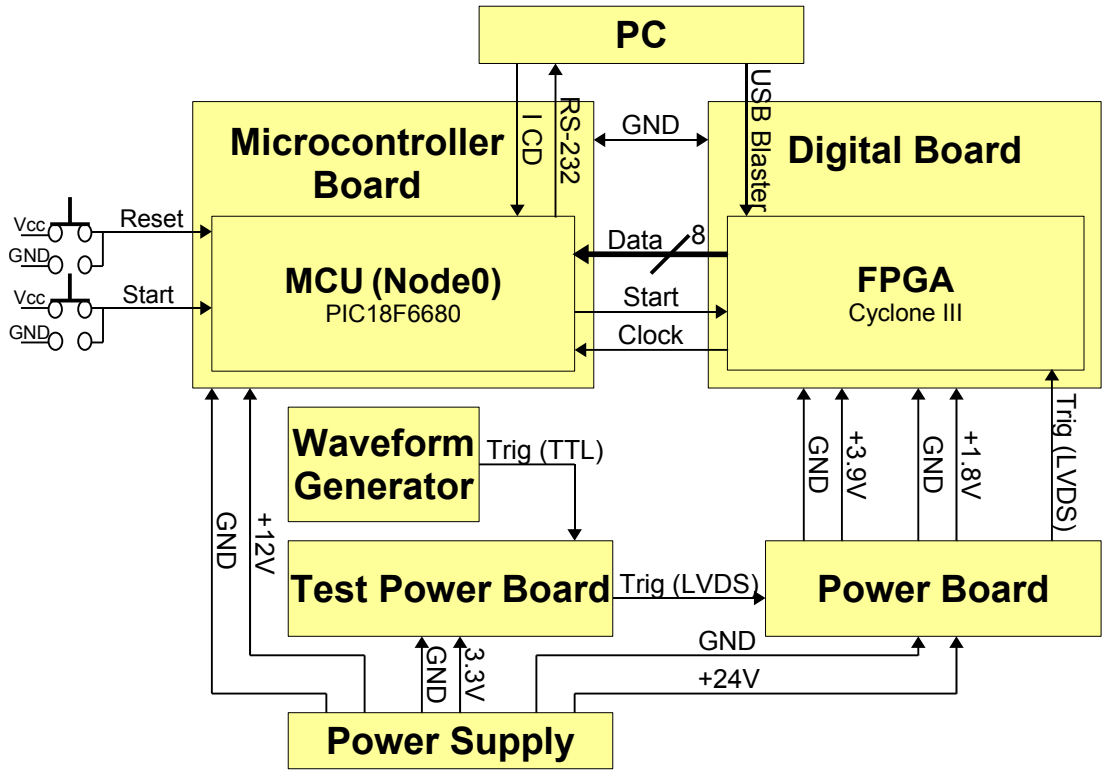

Abbildung 6.8: Schematischer Aufbau: „Test_FPGA_extTrigger(16)“

Anstelle von Node1 wird nun das Digital Board mit dem FPGA verwendet. Neben den bekannten Datenleitungen ($PortF$) und dem Startsignal ($PinB0$), wird analog zum Programm Test(16) die zugehörige Clock vom FPGA an den MCU ($PinB1$) gesendet.
Das Programm des MCU ist dabei identisch zu Test(16). Die Neuerungen beschränken sich also auf den Versuchsaufbau. Neu sind hier zwei Komponenten. Zum einen der Waveform Generator, der zwar schon verwendet wurde, aber nun eine neue Funk-

tion besitzt, und zum anderen das Test Power Board, welches ebenso wie das Power Board und das Microcontroller Board von dem Power Supply mit Spannung versorgt wird (GND und $+3,3$ V).

Die neue Aufgabe des Waveform Generator besteht darin, das Triggersignals für den FPGA bereit zu stellen. Mit dem Triggersignal soll ein detektiertes Myon simuliert werden. Erreicht der Trigger, nachdem er auf dem Test Power Board von einem TTL (Transistor-Transistor-Logik) Signal zu einem LVDS (Low Voltage Differential Signaling) Signal umgewandelt wurde, den FPGA, so zeichnet dieser Informationen zu diesem *Ereignis* auf. Anschließend überträgt er jene Informationen, die in einem Datenblock mit je 16 Datenwörtern zusammengefasst sind, an den MCU.
In Abbildung 6.9 ist das verwendete Triggersignal zu sehen.

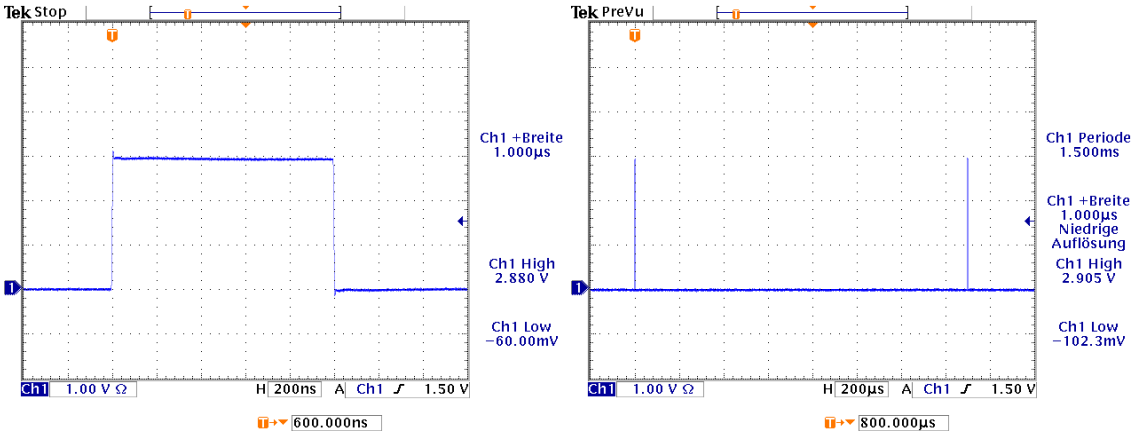

Abbildung 6.9: Darstellung des Triggersignals: „Test_FPGA_extTrigger(16)“

Die Gestalt des Signals lässt sich in dem linken Oszilloskopbild nachvollziehen. Gewählt wurde ein einfacher Rechteckimpuls mit einer Amplitude von 0 V (Low Level) bis ca. +3 V (High Level) und einer Pulsbreite von ca. 1 μs. Die Pulsbreite spielt jedoch eine untergeordnete Rolle, da der FPGA lediglich die ansteigende Flanke detektiert. Die vergleichsweise lange Periode wurde gewählt, um sicher zu stellen, dass das Signal detektiert wird.
Die Periode des Signals, die in dem linken Oszilloskopbild vermessen wurde, ist allerdings von entscheidender Bedeutung. Mit einer Dauer von $1,5$ ms entspricht sie der minimalen Burstperiode des Clocksignals vom FPGA an den MCU. Da der FPGA nach Empfang des Triggersignals die 16 Datenwörter mit den dazugehörigen 16 Pulsen der Clock an den Microcontroller sendet, besitzt der MCU im Anschluss daran genügend Zeit (ca. $1,4$ ms), um die Daten an den PC zu übermitteln.

Die Daten, die an den MCU und anschließend an den PC übertragen werden, besitzen eine besondere Struktur. Wie bereits erwähnt besteht ein Datenblock aus 16 Datenwörtern.
Damit man den Datenblock als solchen erkennt, wird zu Beginn und Abschluss des

Blocks ein fixer Wert gesetzt. Er beginnt mit dem sogenannten 16 bit großen *Magic Word*, das auf zwei Datenwörter (1., 2.) mit folgenden festen Werten aufgeteilt wird: Hex: 41, CB. Nach dem Magic Word folgen dann die Informationen des PMT, die als *PMT-Data* bezeichnet und auf acht Datenwörter (3.-10.) aufgeteilt werden. Zur Simulation wurde hier ein 8 bit breiter Counter verwendet. Um die Daten dem zugehörigen Ereignis zuordnen zu können, schließt sich die Ereignisnummer namens *Eventnumber* an. Sie besteht aus einem 16 bit breiten Counter, der ebenfalls auf zwei Datenwörter (11., 12.) aufgeteilt wird, wobei das letzte Datenwort (LSB) zuerst gesendet wird. Jeder Datenblock wird also in aufsteigender Reihenfolge durchnummeriert. Zusätzlich zu den Informationen des PMT und der Ereignisnummer, sind die darauf folgenden zwei Byte (13., 14.) für die Zeitdifferenz zwischen zwei Ereignissen vorgesehen. Sie werden als *Time Stamp* bezeichnet und bestehen hier aus fixen Werten: Hex: $1A$, $2B$. Abgeschlossen wird der Datenblock mit dem sogenannten 16 bit großen *End of Block*, der ebenfalls einen fixen Wert darstellt und auf zwei Bytes (15., 16.) aufgeteilt wird: Hex: 42, $4F$.

In der Simulation des FPGA-Programms, welche in Abbildung 6.10 zu sehen ist, kann der Aufbau des ersten Datenblocks nachvollzogen werden.

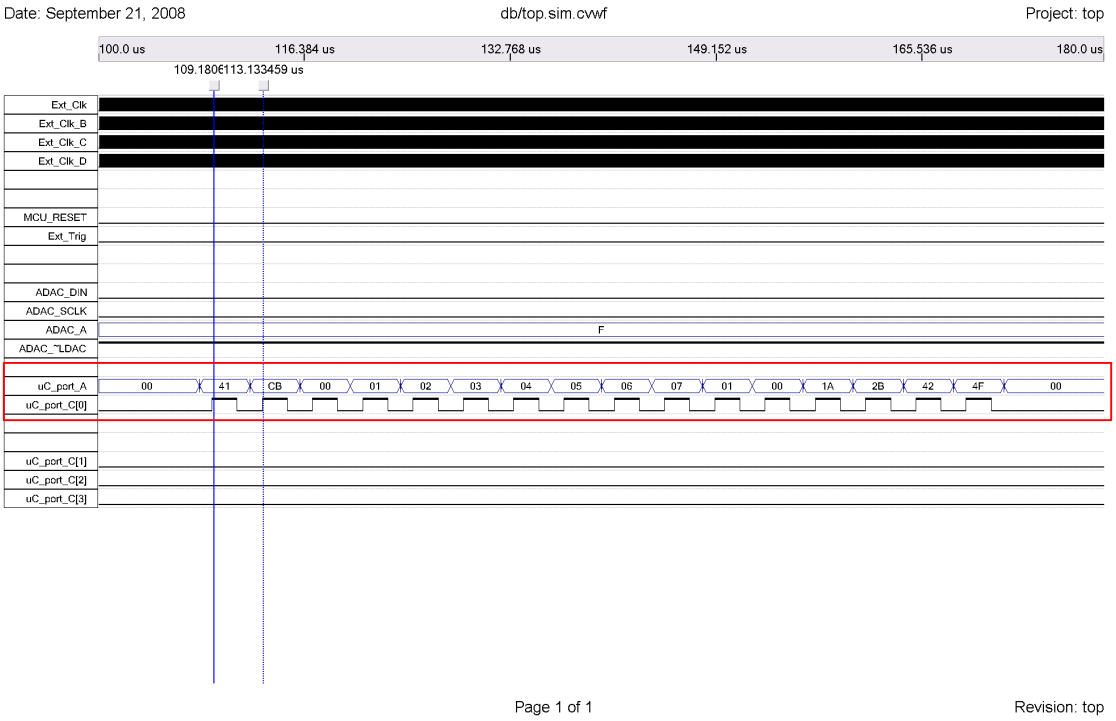

Abbildung 6.10: Simulation des FPGA: „Test_FPGA_extTrigger(16)“ [Kol08]

Die Signale, die den MCU betreffen, sind dort rot markiert. Im Detail sind es die Clock, die über den Pin auf dem Digital Board namens *uC_port_C[0]* ausgegeben

wird, und die Datenwörter, welche über den Port *uC_port_A* ausgegeben werden. Man erkennt die erwarteten fixen Werte und die Werte der Counter.

Werden nun die Komponenten gemäß des vorgestellten Versuchsaufbaus miteinander verbunden und startet man das Programm des MCU, womit gleichzeitig das FPGA-Programm gestartet wird, so werden die in Tabelle 6.4 illustrierten Daten an den PC ausgeben.

41	CB	00	01	02	03	04	05	06	07	01	00	1A	2B	42	4F
41	CB	08	09	0A	0B	0C	0D	0E	0F	02	00	1A	2B	42	4F
41	CB	10	11	12	13	14	15	16	17	03	00	1A	2B	42	4F
41	CB	18	19	1A	1B	1C	1D	1E	1F	04	00	1A	2B	42	4F
41	CB	20	21	22	23	24	25	26	27	05	00	1A	2B	42	4F
41	CB	28	29	2A	2B	2C	2D	2E	2F	06	00	1A	2B	42	4F
41	CB	30	31	32	33	34	35	36	37	07	00	1A	2B	42	4F
41	CB	38	39	3A	3B	3C	3D	3E	3F	08	00	1A	2B	42	4F
41	CB	40	41	42	43	44	45	46	47	09	00	1A	2B	42	4F
41	CB	48	49	4A	4B	4C	4D	4E	4F	0A	00	1A	2B	42	4F
41	CB	50	51	52	53	54	55	56	57	0B	00	1A	2B	42	4F
41	CB	58	59	5A	5B	5C	5D	5E	5F	0C	00	1A	2B	42	4F
41	CB	60	61	62	63	64	65	66	67	0D	00	1A	2B	42	4F
41	CB	68	69	6A	6B	6C	6D	6E	6F	0E	00	1A	2B	42	4F
41	CB	70	71	72	73	74	75	76	77	0F	00	1A	2B	42	4F
41	CB	78	79	7A	7B	7C	7D	7E	7F	10	00	1A	2B	42	4F
41	CB	80	81	82	83	84	85	86	87	11	00	1A	2B	42	4F
41	CB	88	89	8A	8B	8C	8D	8E	8F	12	00	1A	2B	42	4F

Tabelle 6.4: Ausgabe an PC (Hex): „Test_FPGA_extTrigger(16)“

Pro Zeile ist ein Datenblock dargestellt. Die fixen Werte für das Magic Word, Time Stamp und End of Block zeigen die vorgestellten Werte. Betrachtet man die PMT-Data und die Eventnumber, so entsprechen diese den Counterdaten.
Die Ausgabe an den PC ist somit identisch zur Erwartung und der Simulation des FPGA.

6.7 Programm 7: „Test(50)“

Das nachfolgende FPGA Programm sieht die Ausgabe von 50 Datenwörtern vor, da nun mehr Informationen über das PMT Signal aufgezeichnet werden sollen und ein breiterer Counter für den Time Stamp verwendet wird[9]. Aus diesem Grund wurde eine neue Version des MCU Programms namens *Test(50)* entwickelt, die in der Lage ist 50 anstelle von 16 Bytes auszulesen, zwischenzuspeichern und an den PC zu senden.
Der dazugehörige Versuchsaufbau ist in Abbildung 6.11 veranschaulicht.

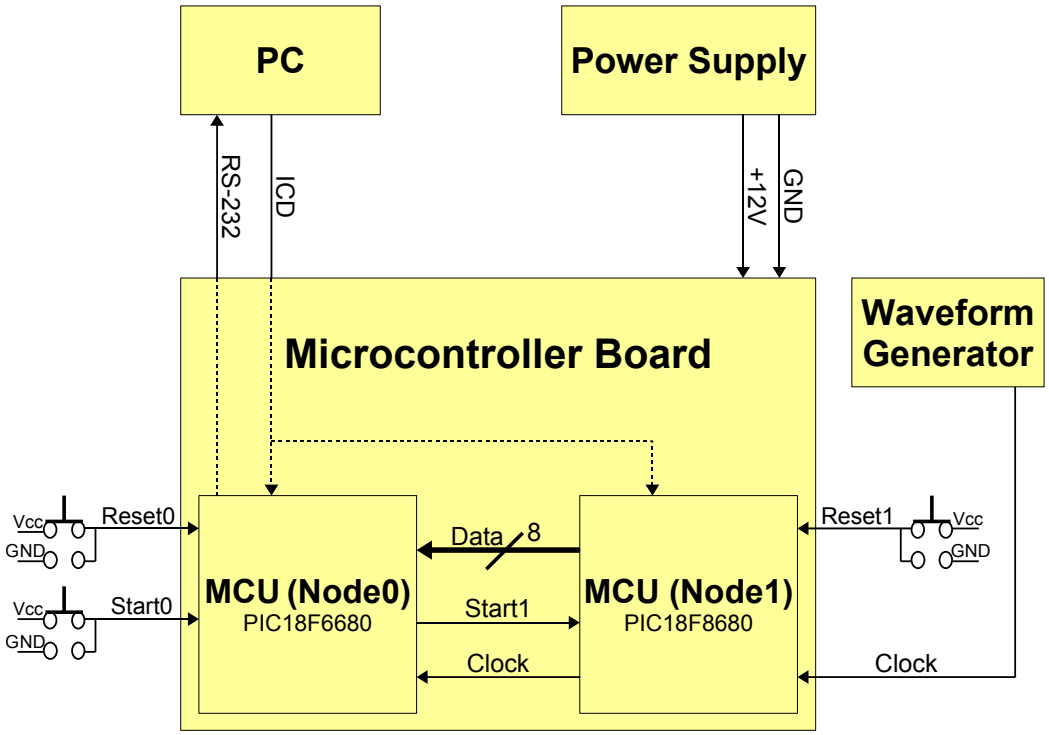

Abbildung 6.11: Schematischer Aufbau: „Test(50)“

Wie man der Abbildung entnehmen kann, ist der Versuchsaufbau identisch zum Programm Test(16). Die Modifizierung beschränkt sich lediglich auf die Anzahl der Wörter. Demnach ist in dem Programm anstelle des 16 Byte großen Arrays ein 50 Byte großes Array[10] wieder zu finden.

Da 50 Byte ausgelesen werden, benötigt der Microcontroller 50 Pulse vom FPGA. Die Zeit, die zum Auslesen und Zwischenspeichern eines Datenwörts nötig ist, ändert sich nicht. Folglich beträgt die minimale Periode der Clock weiterhin 4 μs. Allerdings reicht die bisher verwendete Bursperiode nicht mehr aus. Errechnet man die nötige

[9]Siehe dazu Kapitel 6.8 auf Seite 58. Dort wird der genaue Aufbau der 50 Datenwörter erläutert.
[10]Siehe dazu den Quellcode für dieses Programm im Anhang A.2 auf Seite 68.

Zeit für die Datenübertragung von 50 Bytes anhand der maximalen Bitrate von 115200 bit/s, so ergibt sich folgender Wert: $50 \cdot 8$ bit / (115200 bit/s) = 3,47 ms. Die Messung (analog zu Test(16)) ergibt eine minimale Burstperiode von 4,3 ms. Dabei lässt sich die Abweichung ebenfalls durch zusätzlich benötigte Rechenschritte erklären. Die Gestalt der Clock kann anhand der Oszilloskopbilder in Abbildung 6.12 nachvollzogen werden.

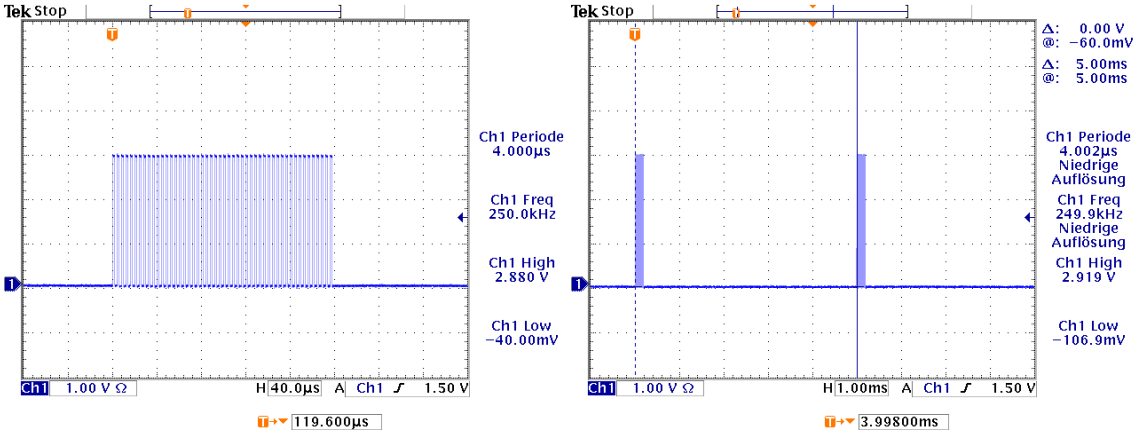

Abbildung 6.12: Darstellung der Clock: „Test(50)“

Für die einfachere Handhabung wird anstelle der minimalen Burstperiode die gezeigte Periode von 5 ms (rechts), die einer Burstfrequenz von 200 Hz entspricht, verwendet.

Das Timing bei dem Zusammenspiel zwischen Daten und der Clock ist dabei analog zum Programm Test(16), wie man in Abbildung 6.13 erkennen kann.

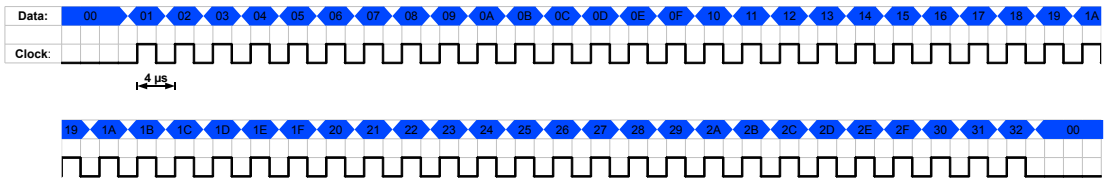

Abbildung 6.13: Timingdiagramm von Daten und Clock: „Test(50)“

Es wird somit auch hier sichergestellt, dass das korrekte Datenwort zur richtigen Zeit (Clock high) vom MCU ausgelesen wird.

Startet man die Kommunikation zwischen Node0 und Node1, so werden folgende Daten - zu sehen in Tabelle 6.5 - an den PC ausgegeben.

01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19
1A	1B	1C	1D	1E	1F	20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F	30	31	32
33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F	40	41	42	43	44	45	46	47	48	49	4A	4B
4C	4D	4E	4F	50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F	60	61	62	63	64
65	66	67	68	69	6A	6B	6C	6D	6E	6F	70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D
7E	7F	80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F	90	91	92	93	94	95	96
97	98	99	9A	9B	9C	9D	9E	9F	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF
B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF	C0	C1	C2	C3	C4	C5	C6	C7	C8
C9	CA	CB	CC	CD	CE	CF	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF	E0	E1
E2	E3	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA

Tabelle 6.5: Ausgabe an PC (Hex): „Test(50)“

Erkennbar sind die einzelnen Datenblöcke mit je 50 Datenwörtern, die in der Darstellung auf zwei Zeilen aufgeteilt sind und die erwarteten Counterdaten zeigen.

6.8 Programm 8: „Test_FPGA_intTrigger(50)“

Das letzte und abschließende Handshakeprogramm dient der Realisierung der Kommunikation von MCU und FPGA, wobei der FPGA intern getriggert wird. Dementsprechend lautet der Name des Programms *Test_FPGA_intTrigger(50)*.

Die Programmierung des FPGA wurde auch in diesem Fall von dem wissenschaftlichen Mitarbeiter der Universität Siegen Yury Kolotaev übernommen. Ebenso wurde zusammen mit ihm der Versuch aufgebaut und die Kommunikation getestet.

Das Programm des MCU ist identisch zu Test(50), jedoch ist der Versuchsaufbau – dargestellt in Abbildung 6.14 – vollkommen verschieden.

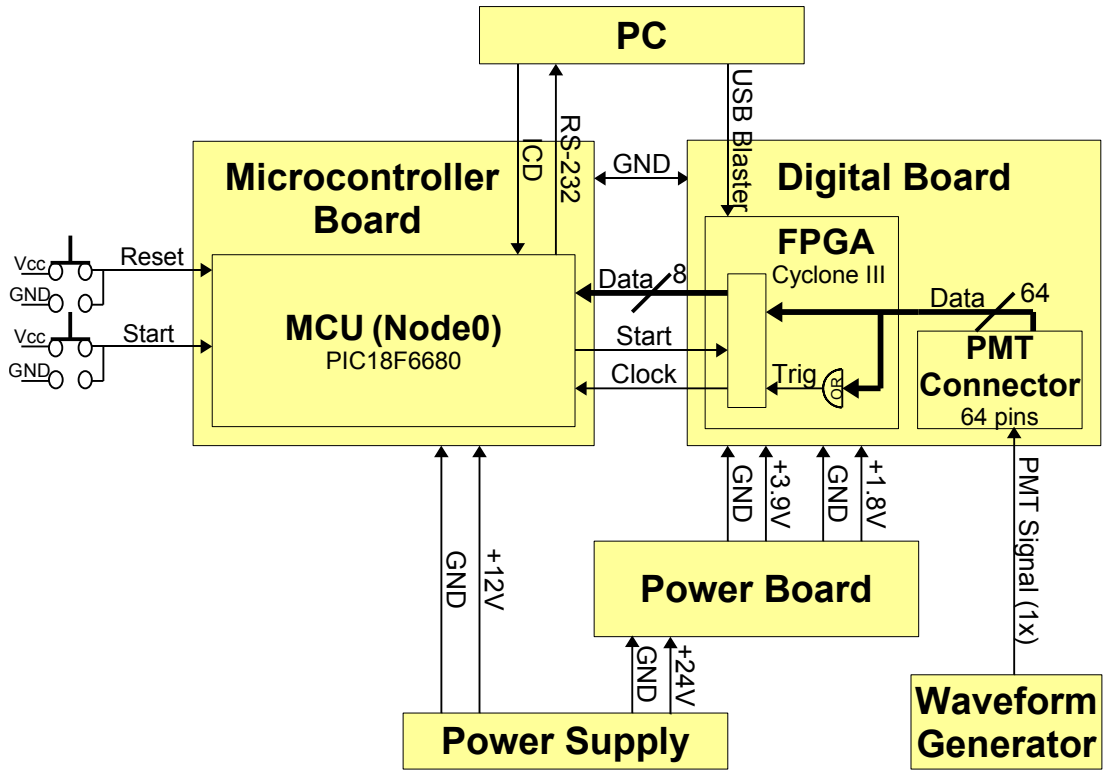

Abbildung 6.14: Schematischer Aufbau: „Test_FPGA_intTrigger(50)“

Die illustrierten Komponenten, Spannungsversorgungen, Handshake- und Datenleitungen zwischen FPGA und MCU, Schnittstellen zum PC und Pushbuttons des Microcontrollers werden in der bereits vorgestellten Art und Weise benutzt. Somit beschränken sich die wesentlichen Neuerungen auf die Verwendung des Waveform Generators und die Erzeugung des internen Triggersignals.

Der Waveform Generator simuliert bei diesem Versuchsaufbau eines der 64 PMT Signale, dessen Gestalt in Abbildung 6.15 zu sehen ist.

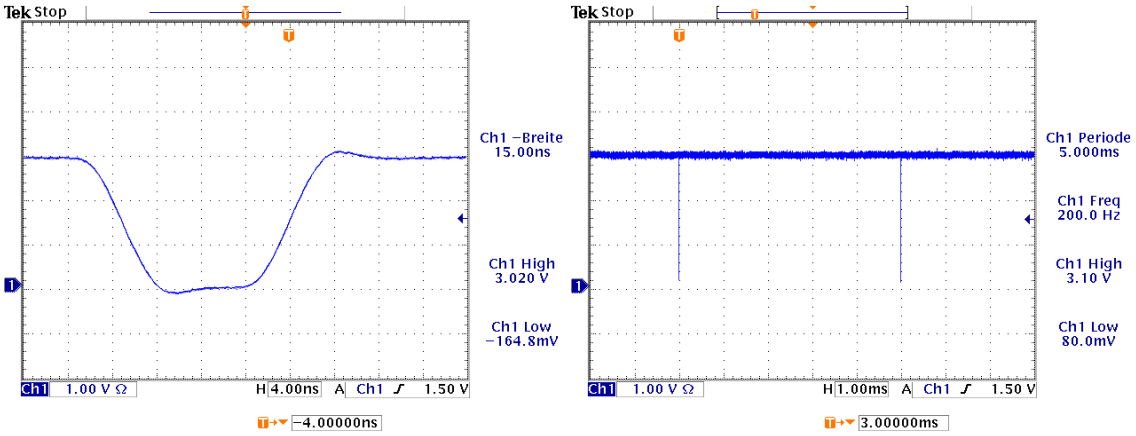

Abbildung 6.15: Darstellung des PMT Signals: „Test_FPGA_intTrigger(50)“

Im linken Oszilloskopbild erkennt man, dass es sich bei dem PMT Signal um einen negativen Puls handelt. Der Ruhepegel liegt also auf dem High Level (+3 V), wohingegen das Minimum des Signals auf dem Low Level (0 V) liegt. Die Breite des Signals ist variabel und beträgt hier 15 ns.
Dem rechten Oszilloskopbild ist die Periodendauer des Signals entnehmbar. Sie beträgt 5 ms und entspricht der Burstperiode des Programms Test(50).

Erreicht das mit dem Waveform Generator simulierte PMT Signal bei der programmierten 64 Pin Schnittstelle namens *PMT Connector* das Digital Board, so wird dieses intern an den FPGA weitergeleitet und aufgesplittet. Zum einen wird über ein OR-Gatter[11], an dem alle 64 Eingangskanäle anliegen, ein Trigger erzeugt, sobald an mindestens einem der Eingangskanäle ein Signal anliegt. Dieser Trigger initialisiert die Datenübertragung. Zum anderen werden Daten über den PMT Connector gesammelt, zwischengespeichert und für die Datenübertragung an den Microcontroller vorbereitet. Anschließend wird ein Datenblock aus 50 Datenwörtern mit der dazugehörigen Clock – identisch zum Programm Test(50) – an den Microcontroller gesendet. Wegen der Periode des PMT Signals, also der Burstperiode der Clock von 5 ms, hat der MCU im Anschluss daran genügend Zeit den Datenblock an den PC zu übermitteln.

Der Datenblock ist ähnlich zu jenem in Programm Test_FPGA_extTrigger(16) aufgebaut. Um einen Block aus 50 Byte als solchen erkennen zu können beginnt auch dieser mit einem 16 bit großen Magic Word mit den bekannten fixen Werten: Hex: 41, CB, das auf zwei Byte (1., 2.) aufgeteilt wird.
Die danach folgenden 40 Byte (3.-42.) sind für PMT-Data reserviert. Unterteilt wer-

[11] Ein OR-Gatter ist ein Gatter mit mehreren Eingängen und einem Ausgang, bei dem der Ausgang genau dann ein High-Signal liefert, wenn an mindestens einem Eingang ein High-Signal anliegt.

den diese in fünf identisch aufgebaute Blöcke mit je acht Datenwörtern. Dabei lassen sich jeweils nur aus den ersten Bytes Informationen über das PMT Signal ablesen, die anderen Bytes beinhalten fixe Werte: Hex: 22, 33, 44, 55 66, 77 und 88. Die Informationen über das PMT Signal kommen durch das sogenannte *Sampling* zustande. Dazu wird das Signal in äquidistanten Zeitabständen abgetastet und es entsteht ein zeitdiskretes Signal. Die verwendete Abtastrate, auch *Samplingrate* genannt, beträgt 160 MHz und entspricht einer Periode von $6,25$ ns. Demnach wird das vorgestellte zeitkontinuierliche PMT Signal (Signalbreite von 15 ns) zu einem zeitdiskreten Signal, mit einer Signalbreite von $2 \cdot 6,25 \text{ ns} = 12,5 \text{ ns} < 15$ ns. Es entsteht also ein digitales Signal das zweimal eine logische 1 (zeitdisrketes Signal high) und danach mehrfach eine logische 0 (zeitdiskretes Signal low) ausgibt, da durch das Sampling zweimal ein logische 1 (PMT Signal high) registriert wird. Ist das zeitdiskrete Signal high, so wird der programmierte hexadezimale Wert 10 ausgegeben. Bei Low Level wird 11 (Hex) angezeigt. Demnach werden je für die ersten Bytes der fünf Datenblöcke der PMT-Data im konkreten Beispiel folgende Werte ausgegeben: Hex: 10, 10, 11, 11 und 11.
Im Anschluss an PMT-Data folgt die 16 bit breite Eventnumber, welche die Nummer des Ereignisses – aufgeteilt auf zwei Datenwörter (43., 44.) – wiedergibt. Die Eventnumber ist identisch zu der im Programm Test_FPGA_extTrigger(16).
Die nächsten vier Byte stehen für den 32 bit großen Time Stamp. Auch hier wird das letzte Byte (LSB) zuerst übermittelt. Jedoch sind die ausgegeben Daten keine fixen Werte, sondern Werte eines Counters, der mit einer Frequenz von 40 MHz hochzählt. Jener zählt solange hoch, bis ein neues PMT Signal anliegt, und startet im Anschluss wieder bei null. So wird allein die zeitliche Differenz zwischen zwei Signalen ermittelt. Für das verwendete PMT Signal beträgt die zeitliche Differenz ca. 5 ms. Somit errechnet sich die folgende Anzahl der Zählschritte: $0,005 \text{ s} \cdot 40000000 \text{ Hz} = 200000$. Es müsste also der hexadezimale Wert $00030D40$ zwischen zwei Signalen ausgegeben werden.
Der gesamte Datenblock wird durch den 16 bit großen End of Block, der auch auf zwei Bytes (49., 50.) mit den bekannten Werten: Hex: 42, $4F$ aufgeteilt wird, abgeschlossen.

Zur Illustration des ersten Datenworts ist in Abbildung 6.16 die Simulation des FPGA-Programms zu sehen. Die für die Simulation notwendigen externen Signale wurden zuvor in das Simulationsprogramm eingegeben. So wurden für das PMT Signal die gezeigten Einstellungen eingefügt.

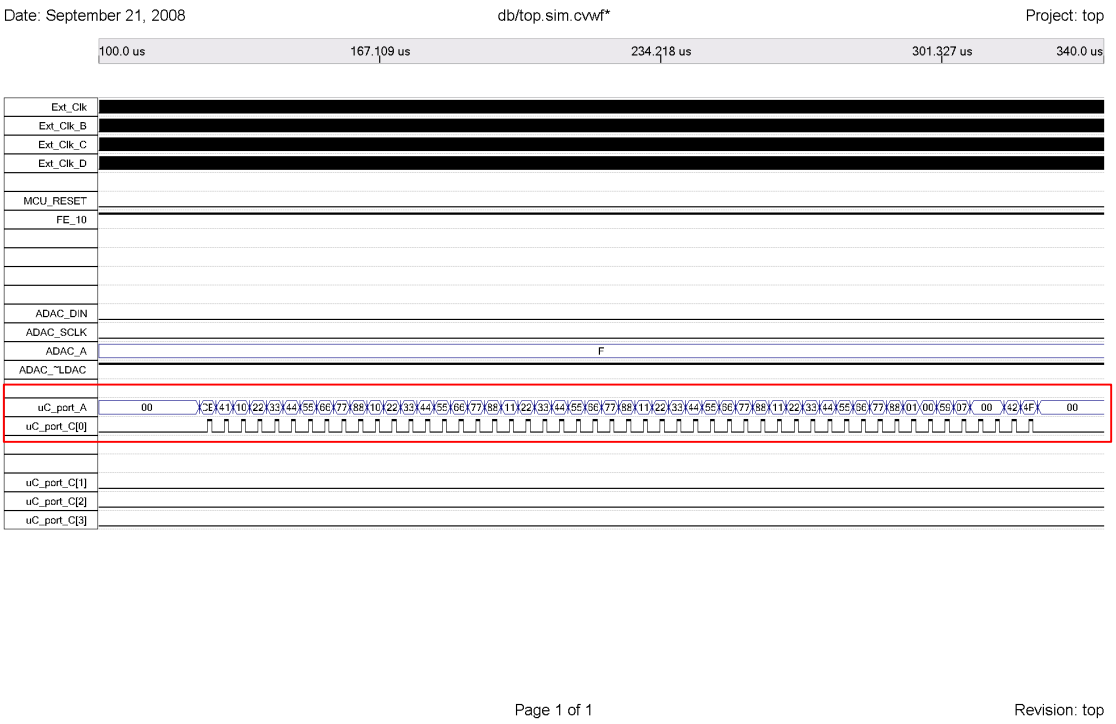

Abbildung 6.16: Simulation des FPGA: „Test_FPGA_intTrigger(50)“ [Kol08]

Betrachtet man den relevanten rot markierten Bereich, so stellt man fest, dass PMT-Data, ebenso wie die fixen Werte und die Eventnumber übereinstimmen.
Nur der Time Stamp zeigt eine Abweichung. Diese lässt sich dadurch erklären, dass es sich um den ersten Datenblock handelt. Es kann also folglich keine Differenz gemessen werden.

Überprüft werden kann der Time Stamp also nur an der Ausgabe der Daten an den PC. Für die genannten Einstellungen und den präsentierten Versuchsaufbau werden die in Tabelle 6.6 veranschaulichten Daten ausgegeben.

CB	41	10	22	33	44	55	66	77	88	10	22	33	44	55	66	77	88	11	22	33	44	55	66	77
88	11	22	33	44	55	66	77	88	11	22	33	44	55	66	77	88	01	00	FD	77	50	06	42	4F
CB	41	10	22	33	44	55	66	77	88	10	22	33	44	55	66	77	88	11	22	33	44	55	66	77
88	11	22	33	44	55	66	77	88	11	22	33	44	55	66	77	88	02	00	3D	0D	03	00	42	4F
CB	41	10	22	33	44	55	66	77	88	10	22	33	44	55	66	77	88	11	22	33	44	55	66	77
88	11	22	33	44	55	66	77	88	11	22	33	44	55	66	77	88	03	00	3C	0D	03	00	42	4F
CB	41	10	22	33	44	55	66	77	88	10	22	33	44	55	66	77	88	11	22	33	44	55	66	77
88	11	22	33	44	55	66	77	88	11	22	33	44	55	66	77	88	04	00	3D	0D	03	00	42	4F
CB	41	10	22	33	44	55	66	77	88	10	22	33	44	55	66	77	88	11	22	33	44	55	66	77
88	11	22	33	44	55	66	77	88	11	22	33	44	55	66	77	88	05	00	3C	0D	03	00	42	4F

Tabelle 6.6: Ausgabe an PC (Hex): „Test_FPGA_intTrigger(50)“

Auch hier stimmen die Werte für Magic Word, PMT-Data, Eventnumber und End of Block mit der Erwartung überein.
Der Time Stamp gibt ab einschließlich dem zweiten Datenblock einen Wert aus, der sich regelmäßig wiederholt. Er zeigt zwei Werte: Hex: $00030D3D$, Dez: 199997 und Hex: $00030D3C$, Dez: 199996. Diese unterscheiden sich also untereinander nur um einen Zählschritt, was durch mögliche Rundungen innerhalb eines Zählschrittes erklärt werden kann. Zieht man den errechneten Wert: Hex: $00030D40$, Dez: 200000 der Betrachtung hinzu, ist eine Abweichung beider Werte von weniger als 1 ‰ vorhanden. Ein möglicher Grund dafür ist in der zu ungenauen Berechnung der Zeitdifferenz zu sehen, da man von der Periode mindestens die Signalbreite abziehen müsste.

Zusammenfassend deckt sich die Ausgabe an den PC mit der Erwartung und der Simulation des FPGA. Die Kommunikation zwischen FPGA und MCU konnte also unter der Verwendung eines internen Triggers erfolgreich getestet werden.

7 Zusammenfassung und Ausblick

Das abschließend präsentierte Handshakeprogramm Test_FPGA_intTrigger(50) bildet die Grundlage für die Kommunikation von FPGA und MCU beim finalen Hardwaretest.
Mit diesem Programm ist es bereits gelungen die einzelnen Kanäle der AMIGA Ausleseelektronik, bestehend aus Mother Board, Daughter Boards, Digital Board, Power Board und Microcontroller Board, einzeln zu überprüfen. Dazu wurde in Zusammenarbeit mit Yury Kolotaev, Michael Pontz und Uwe Fröhlich aus der Arbeitsgruppe Teilchenphysik der Universität Siegen das Sandwich aus Mother Board, Daughter Boards, Digital Board und dem Power Board zusammengebaut und extern mit dem Microcontroller Board verbunden. Über den PMT Konnektor auf dem Mother Board wurde dann ein vom Waveform Generator simuliertes PMT Signal auf die einzelnen Kanäle gegeben, von dem Mother Board an das dazugehörige Daughter Board weitergeleitet und über die Daughter Boards entsprechend moduliert und verstärkt an das Digital Board mit dem FPGA übermittelt. Dort wurde es analog zur vorgestellten Kommunikation von MCU und FPGA unter Verwendung des internen Triggers detektiert. Die gesammelten Informationen sind anschließend an den Microcontroller und daraufhin mit dem bekannten Kommunikationsprotokoll an den PC übertragen worden. Auf diese Weise konnten Defekte von vier der 64 Leitungen registriert werden, die sich auf mögliche Kurzschlüsse zurückführen lassen.
In Vorbereitung auf den Anschluss eines 32 bit Multiplexer, der es ermöglicht mehrere Datenleitungen parallel zu überprüfen, ist das sogenannten *PMT Housing Board* entwickelt und gebaut worden. Dieses Board wird mit dem Mother Board über den PMT Konnektor verbunden und liefert für den Multiplexer 64 kompatible Anschlüsse.
Prinzipiell wäre es also in Kürze möglich, das Mother Board und die Daughter Boards abschließend auf ihre Funktionalität zu überprüfen. Dazu fehlen allerdings noch die notwendigen Testroutinen, die in naher Zukunft entwickelt werden sollen. Geplant sind u.a. Tests auf mögliches Übersprechen und tote Kanäle. Außerdem soll das Verhalten der Ausleseelektronik bei verschiedenen Temperaturen in der Klimakammer untersucht werden.

Um bei den Hardwaretests der Realität möglichst nahe zu kommen, ist die aktuelle Version des Microcontrollers und das dazugehörige Development Board namens *TMS470 Kickstart Development Kit* der Firma *Texas Instruments* erworben worden. Zurzeit wird am Verständnis der neuen Struktur und der Art und Weise der Programmierung gearbeitet. Es ist vorgesehen, das alte Development Board so bald wie möglich durch das neuere Modell zu ersetzen, sodass anstelle von 8 Bits auch

32 Bits ausgelesen werden können. Nach Möglichkeit soll außerdem der CAN-Bus anstelle der seriellen Schnittstelle für den Datentransfer zum PC genutzt werden.

Zusätzlich konnte Mariela Videla in Argentinien einen Prototypen des aktuellen Microcontroller Boards fertig stellen. Dieser wird in Kürze vorgestellt und getestet werden, sodass das abschließende Design des Digital Boards mit FPGA und integriertem MCU entwickelt und Prototypen gebaut werden können.

Weiterhin ist ein Test des finalen FPGA Codes geplant. Für diesen Zweck wurde bereits die aktuelle Version (entwickelt von Zbigniew Szadkowski) mit dem Testsetup der AMIGA Ausleseelektronik in Siegen überprüft. Es traten einige Probleme mit der Kommunikation des FPGA und dem externen RAM auf, an deren Lösung derzeit gearbeitet wird.

Sobald ein vollständiger Prototyp der AMIGA Ausleseelektronik gebaut und erfolgreich getestet werden konnte, soll der Szintillator sowie der PMT mit der Elektronik verbunden werden. Ab diesem Zeitpunkt wird ein finaler Test des gesamten AMIGA Myonzählers möglich und die Massenproduktion aller Bestandteile kann nach erfolgreicher Überprüfung beginnen.

A Anhang: Beispielprogramme

A.1 Quellcode Test(16)/Test_FPGA_extTrigger(16)

```
/*************************************************************
    Author:      Martin Tigges
                 Exp. Teilchenphysik
                 Universität Siegen
    Version:     Test(16)/Test_FPGA_extTrigger(16)
    MCU:         Node0
    Datum:       September 2008
*************************************************************
    Kommunikation zwischen FPGA/MCU(Node1) und MCU(Node0):
    Auslesen, Speichern und Senden an PC (16 Bytes)
*************************************************************/
//Header-Datei für Prozessor (allgemein):
#include <p18cxxx.h>

//Konfigurationen:
#pragma config OSC = HS    //CPU=20 MHz
#pragma config PWRT = ON   //Power-up-Timer
#pragma config BOR = OFF   //Brown-out-Reset
#pragma config WDT = OFF   //Watchdog Timer
#pragma config LVP = OFF   //Low Voltage ICSP
/*************************************************************/
//Unterprogramm: 8,6us Bit-Delay für Baudrate: 115200bps
void delay_bit (void)
{
  Nop(); Nop(); Nop(); Nop(); Nop(); Nop(); Nop(); Nop();
  Nop(); Nop(); Nop(); Nop(); Nop(); Nop(); Nop(); Nop();
  Nop(); Nop(); Nop(); Nop(); Nop(); Nop(); Nop(); Nop();
  Nop(); Nop(); Nop(); Nop(); Nop(); Nop(); Nop(); Nop();
  Nop(); Nop(); Nop(); Nop(); Nop(); Nop(); Nop(); Nop();
  Nop(); Nop(); Nop(); Nop(); Nop();
}
/*************************************************************/
//Unterprogramm: Senden der Daten zum PC (115200bps)
void sendtopc (void)
{
//Ausgabe von PORTD an PC:
  PORTCbits.RC6 = 0;     //STARTBIT (0)
  delay_bit();           //Aufruf von "delay-bit"

  PORTCbits.RC6 = PORTDbits.RD0;  //BIT1 (LSB)
  delay_bit();           //Delay
```

```
42    PORTCbits.RC6 = PORTDbits.RD1;  //BIT2
43    delay_bit();           //Delay
44    PORTCbits.RC6 = PORTDbits.RD2;  //BIT3
45    delay_bit();           //Delay
46    PORTCbits.RC6 = PORTDbits.RD3;  //BIT4
47    delay_bit();           //Delay
48    PORTCbits.RC6 = PORTDbits.RD4;  //BIT5
49    delay_bit();           //Delay
50    PORTCbits.RC6 = PORTDbits.RD5;  //BIT6
51    delay_bit();           //Delay
52    PORTCbits.RC6 = PORTDbits.RD6;  //BIT7
53    delay_bit();           //Delay
54    PORTCbits.RC6 = PORTDbits.RD7;  //BIT8 (MSB)
55    delay_bit();           //Delay
56
57    PORTCbits.RC6 = 1;     //Stopbit (1)
58    delay_bit();           //Delay
59  }
60  /*********************************************************/
61  //Deklaration der Variablen (automatische im SRAM):
62  unsigned char a[16];  //Array für Daten: 16x8Bits
63  unsigned char i;      //Schleifenvariable
64  /*********************************************************/
65  //Hauptprogramm:
66  void main (void)
67  {
68  //Portinitialisierung:
69    //LED-Port:
70    TRISD = 0x00; //Setzen des Registers (1:Input, 0:Output)
71    PORTD = 0x00; //Setzen des Ports auf Null
72
73    //Sende-Port:
74    CMCON = 0x7;  //Konfiguration von Port...
75    ADCON1 = 0xF; //...als digitalen I/O
76    TRISF = 0xFF; //PortF als Input-Port
77    PORTF = 0x00; //Setzen des Ports auf Null
78
79    //Handshake-Port:
80    TRISB = 0x32; //RB1, RB4, RB5 als Input-Pin; Rest: Output
81    PORTB = 0x30; //RB4, RB5 (Pushbuttons) werden high gesetzt
82
83    //RS-232-Port:
84    TRISC = 0x80; //RB7: Input-Pin, RB6: Output-Pin
85    PORTC = 0xC0; //RB6 und RB7 werden high gesetzt
86
87  //Programmstart auf Knopfdruck (Pushbutton RB4):
88    while (PORTBbits.RB4 != 0); //Warten bis PB gedrückt
89    while (PORTBbits.RB4 != 1); //Warten bis PB losgelassen
90
91  //Startsignal für FPGA Signaldauer: 1us
92    PORTBbits.RB2 = 1;  //Setze RB2 high
93    Nop(); Nop(); Nop(); Nop(); Nop(); Nop(); //Warte 6x160ns
94    PORTBbits.RB2 = 0;  //Setze RB2 low
```

```
95
96  start:       //Sprungmarke
97
98  //16 Bytes speichern:
99    for (i=0; i < 16; i++)  //Schleife: 16 Zählschritte
100   {
101     while (PORTBbits.RB1 != 1); //Warten bis RB1 high ...
102                                 //...(Clock von FPGA/Node1)
103     a[i] = PORTF; //Speichern von PortF in ...
104                   //...entsprechendes Byte des Arrays
105     while (PORTBbits.RB1 != 0); //Warten bis RB1 low ...
106   }                             //...(Clock von FPGA/Node1)
107
108 //16 Bytes an PC senden:
109   for (i=0; i < 16; i++)  //Schleife: 16 Zählschritte
110   {
111     PORTD = a[i]; //Gespeicherten Wert an PORTD ausgegeben
112     sendtopc();   //PortD an PC senden
113   }
114
115 goto start; //Springe zur Marke: "start"
116 }
117 /*********************************************************/
```

A.2 Quellcode Test(50)/Test_FPGA_intTrigger(50)

```
/*************************************************************
    Author:      Martin Tigges
                 Exp. Teilchenphysik
                 Universität Siegen
    Version:     Test(50)/Test_FPGA_intTrigger(50)
    MCU:         Node0
    Datum:       September 2008
*************************************************************
    Kommunikation zwischen FPGA/MCU(Node1) und MCU(Node0):
    Auslesen, Speichern und Senden an PC (50 Bytes)
*************************************************************/
//Header-Datei für Prozessor (allgemein):
#include <p18cxxx.h>

//Konfigurationen:
#pragma config OSC = HS    //CPU=20 MHz
#pragma config PWRT = ON   //Power-up-Timer
#pragma config BOR = OFF   //Brown-out-Reset
#pragma config WDT = OFF   //Watchdog Timer
#pragma config LVP = OFF   //Low Voltage ICSP
/*************************************************************/
//Unterprogramm: 8,6us Bit-Delay für Baudrate: 115200bps
void delay_bit (void)
{
  Nop(); Nop(); Nop(); Nop(); Nop(); Nop(); Nop(); Nop();
  Nop(); Nop(); Nop(); Nop(); Nop(); Nop(); Nop(); Nop();
  Nop(); Nop(); Nop(); Nop(); Nop(); Nop(); Nop(); Nop();
  Nop(); Nop(); Nop(); Nop(); Nop(); Nop(); Nop(); Nop();
  Nop(); Nop(); Nop(); Nop(); Nop(); Nop(); Nop(); Nop();
  Nop(); Nop(); Nop(); Nop(); Nop();
}
/*************************************************************/
//Unterprogramm: Senden der Daten zum PC (115200bps)
void sendtopc (void)
{
//Ausgabe von PORTD an PC:
  PORTCbits.RC6 = 0;     //STARTBIT (0)
  delay_bit();           //Aufruf von "delay-bit"

  PORTCbits.RC6 = PORTDbits.RD0;  //BIT1 (LSB)
  delay_bit();           //Delay
  PORTCbits.RC6 = PORTDbits.RD1;  //BIT2
  delay_bit();           //Delay
  PORTCbits.RC6 = PORTDbits.RD2;  //BIT3
  delay_bit();           //Delay
  PORTCbits.RC6 = PORTDbits.RD3;  //BIT4
  delay_bit();           //Delay
  PORTCbits.RC6 = PORTDbits.RD4;  //BIT5
  delay_bit();           //Delay
  PORTCbits.RC6 = PORTDbits.RD5;  //BIT6
```

```
51    delay_bit();            //Delay
52    PORTCbits.RC6 = PORTDbits.RD6;  //BIT7
53    delay_bit();            //Delay
54    PORTCbits.RC6 = PORTDbits.RD7;  //BIT8 (MSB)
55    delay_bit();            //Delay
56
57    PORTCbits.RC6 = 1;      //Stopbit (1)
58    delay_bit();            //Delay
59  }
60  /*********************************************************/
61  //Deklaration der Variablen (automatische im SRAM):
62  unsigned char a[50];  //Array für Daten: 50x8Bits
63  unsigned char i;      //Schleifenvariable
64  /*********************************************************/
65  //Hauptprogramm:
66  void main (void)
67  {
68  //Portinitialisierung:
69    //LED-Port:
70    TRISD = 0x00; //Setzen des Registers (1:Input, 0:Output)
71    PORTD = 0x00; //Setzen des Ports auf Null
72
73    //Sende-Port:
74    CMCON = 0x7;  //Konfiguration von Port...
75    ADCON1 = 0xF; //...als digitalen I/O
76    TRISF = 0xFF; //PortF als Input-Port
77    PORTF = 0x00; //Setzen des Ports auf Null
78
79    //Handshake-Port:
80    TRISB = 0x32; //RB1, RB4, RB5 als Input-Pin; Rest: Output
81    PORTB = 0x30; //RB4, RB5 (Pushbuttons) werden high gesetzt
82
83    //RS-232-Port:
84    TRISC = 0x80; //RB7: Input-Pin, RB6: Output-Pin
85    PORTC = 0xC0; //RB6 und RB7 werden high gesetzt
86
87  //Programmstart auf Knopfdruck (Pushbutton RB4):
88    while (PORTBbits.RB4 != 0); //Warten bis PB gedrückt
89    while (PORTBbits.RB4 != 1); //Warten bis PB losgelassen
90
91  //Startsignal für FPGA Signaldauer: 1us
92    PORTBbits.RB2 = 1;  //Setze RB2 high
93    Nop(); Nop(); Nop(); Nop(); Nop(); Nop(); //Warte 6x160ns
94    PORTBbits.RB2 = 0;  //Setze RB2 low
95
96  start:      //Sprungmarke
97
98  //16 Bytes speichern:
99    for (i=0; i < 50; i++)  //Schleife: 50 Zählschritte
100   {
101     while (PORTBbits.RB1 != 1); //Warten bis RB1 high ...
102                                 //...(Clock von FPGA/Node1)
103     a[i] = PORTF; //Speichern von PortF in ...
```

```
104                     //...entsprechendes Byte des Arrays
105       while (PORTBbits.RB1 != 0); //Warten bis RB1 low ...
106     }                             //...(Clock von FPGA/Node1)
107
108 //16 Bytes an PC senden:
109     for (i=0; i < 50; i++)  //Schleife: 50 Zählschritte
110     {
111       PORTD = a[i]; //Gespeicherten Wert an PORTD ausgegeben
112       sendtopc();   //PortD an PC senden
113     }
114
115 goto start; //Springe zur Marke: "start"
116 }
117 /*************************************************************/
```

A.3 Quellcode Test_Node1

```
/*************************************************************
    Author:      Martin Tigges
                 Exp. Teilchenphysik
                 Universität Siegen
    Version:     Test_Node1
    MCU:         Node1
    Datum:       September 2008
*************************************************************
    Kommunikation zwischen MCU(Node0) und MCU(Node1):
    MCU(Node1) sendet Daten an MCU(Node0)
*************************************************************/
//Header-Datei für Prozessor (allgemein):
#include <p18cxxx.h>

//Konfigurationen:
#pragma config OSC = HS    //CPU=20 MHz
#pragma config PWRT = ON   //Power-up-Timer
#pragma config BOR = OFF   //Brown-out-Reset
#pragma config WDT = OFF   //Watchdog Timer
#pragma config LVP = OFF   //Low Voltage ICSP
/*************************************************************/
//Unterprogramm: 8,6us Bit-Delay für Baudrate: 115200bps
void delay_bit (void)
{
  Nop(); Nop(); Nop(); Nop(); Nop(); Nop(); Nop(); Nop();
  Nop(); Nop(); Nop(); Nop(); Nop(); Nop(); Nop(); Nop();
  Nop(); Nop(); Nop(); Nop(); Nop(); Nop(); Nop(); Nop();
  Nop(); Nop(); Nop(); Nop(); Nop(); Nop(); Nop(); Nop();
  Nop(); Nop(); Nop(); Nop(); Nop(); Nop(); Nop(); Nop();
  Nop(); Nop(); Nop(); Nop(); Nop();
}
/*************************************************************/
//Hauptprogramm:
void main(void)
{
//Portinitialisierung:
  //LED-Port:
  TRISD = 0x00; //Setzen des Registers (1:Input, 0:Output)
  PORTD = 0x00; //Setzen des Ports auf Null

  //Sende-Port:
  CMCON = 0x7;  //Konfiguration von Port...
  ADCON1 = 0xF; //...als digitalen I/O
  TRISF = 0x00; //PortF als Output-Port
  PORTF = 0x00; //Setzen des Ports auf Null

  //Handshake-Port:
  TRISB = 0x3C; //RB2, RB3, RB4, RB5 als Input; Rest: Output
  PORTB = 0x30; //RB4, RB5 (Pushbuttons) werden high gesetzt

```

```
51 //Programmstart durch Startsignal:
52   while (PORTBbits.RB2 != 1);    //Warten bis RB2 high
53   while (PORTBbits.RB2 != 0);    //Warten bis RB2 low
54
55 //Senden der Daten (Counter):
56   while (1)             //Endlosschleife
57   {
58     while (PORTBbits.RB3 != 1); //Warten auf RB3 high ...
59                                 //...(Clock von WG)
60     PORTBbits.RB1 = 1;  //Setze RB1 high (Clock an Node0)
61
62     PORTF++;          //Hochzählen von PortF (Sende-Port)
63     PORTD++;          //Hochzählen von PortD (LED-Port)
64
65     while (PORTBbits.RB3 != 0); //Warten auf RB3 low
66                                 //...(Clock von WG)
67     PORTBbits.RB1 = 0;  //Setze RB1 low (Clock an Node0)
68   }
69 }
70 /************************************************************/
```

Literaturverzeichnis

[Alk75] Alkhofer, O. C.: *Introduction to Cosmic Radiation.* Verlag Carl Thiemig, München, 1975.

[astde] `http://www.astroteilchenphysik.de`, letzter Abruf: 21. April 2009.

[augde] `http://www.auger.de`, letzter Abruf: 21. April 2009.

[augorg] `http://www.auger.org`, letzter Abruf: 21. April 2009.

[Bet03] Bethge, C.: *Entwicklung und Aufbau eines Slow Control Systems für die Pierre Auger Fluoreszenzdetektoren.* Diplomarbeit, Universität Karlsruhe, 2003.

[Blü01] Blümer, H. und Guérard, C.-Kj.: *Die höchsten Energien im Universum.* Nachrichten, Forschungszentrum Karlsruhe, Jahrg. 33, Seiten 95-102, 2001.

[Def04] Deffner, F.: *Kosmische Strahlung.* Präsentation, Universität Erlangen-Nürnberg, `http://www.pi1.physik.uni-erlangen.de/~katz/ws04/atp/talks/fd/FD.ppt`, 2004.

[Etc07] Etchegoyen, A., for the Pierre Auger Collaboration: *AMIGA, Auger Muons and Infill for the Ground Array.* International Cosmic Ray Conference, Mérida, México, 2007.

[Fle08] Fleck, I.: *Astroteilchenphysik.* Vorlesung, Universität Siegen, 2008.

[Frö09] Fröhlich, U.: *Charakterisierung der Szintillatoren und der Ausleseelektronik des AMIGA-Myonsystems.* Masterarbeit, Universität Siegen, 2009.

[Gor05] Gorges, F.: *Astronomie mit Teilchen.* Präsentation, Universität Mainz, `http://www.physik.uni-mainz.de/F-Praktikum/Astronomie%20mit%20Teilchen_Gorges%2005.02.07.pdf`, 2005.

[Gru00] Grupen, C.: *Astroteilchenphysik. Das Universum im Licht der kosmischen Strahlung.* Friedr. Vieweg & Sohn Verlagsgesellschaft mbH, Braunschweig/Wiesbaden, 2000.

[Hau03] Haungs, A.: *Einführung in die Physik der kosmischen Strahlung.* Präsentation, Universität Heidelberg, `http://www-ik.fzk.de/~haungs/CR-5.pdf`, 2003.

[Kam00] Kampert, K.-H. und Blümer, H.: *Die Suche nach den Quellen der kosmischen Strahlung.* Publikation, Universität Wuppertal, `http://astro.uni-wuppertal.de/~kampert/Publications-PDF/Kampert-PB00.pdf`, 2000.

[Kol08] Kolotaev, Y.: *Top.* Simulationsfile, Universität Siegen, 2008.

[Med07] Medina Tanco, G., for the Pierre Auger Collaboration: *Astrophysics Motivation behind the Pierre Auger Southern Observatory Enhancements.* International Cosmic Ray Conference, Mérida, México, 2007.

[Men06] Menke, S. und Bethke, S.: *Kosmische Strahlung: geladene Primärteilchen.* Präsentation, Universität München, `http://www.mppmu.mpg.de/english/bethke_ss06_v09.pdf`, 2006.

[Mic03] Microchip Technology Incorporation: *PICDEM CAN-LIN 3 Development Kit for 64/80-pin Enhanced CAN Devices User´s Guide.* Datasheet, `http://ww1.microchip.com/downloads/en/DeviceDoc/51423a%20Users%20Guide.pdf`, 2003.

[Mic04] Microchip Technology Incorporation: *PIC18F6585/8585/6680/8680 Data Sheet 64/68/80-Pin High-Performance, 64-Kbyte Enhanced Flash Microcontrollers with ECAN Module.* Datasheet, `http://ww1.microchip.com/downloads/en/DeviceDoc/30491c.pdf`, 2004.

[Mue06] Müller, S.: *Ausgedehnte Luftschauer.* Präsentation, Universität Nijmegen, `http://particle.astro.kun.nl/hs/mueller.pdf`, 2006.

[Neu08] Neuser, J.: *Erstellung einer Testumgebung für das digitale Ausleseboard der AMIGA Elektronik.* Bachelorarbeit, Universität Siegen, 2008.

[Ost07] Ostler, M.: *Kosmische Strahlung.* Präsentation, Universität Erlangen-Nürnberg, `http://pulsar.sternwarte.uni-erlangen.de/wilms/teach/astrosem07/kosmische_strahlung.pdf`, 2007.

[Ros08] ROSENTHAL, O.: *Suche nach Clustern im Pierre Auger Experiment.* Bachelorarbeit, Universität Siegen, 2008.

[Sch08] SCHARUN, M.: *Charakterisierung der Vorverstärkerelektronik der Myonzähler des AMIGA Projekts.* Bachelorarbeit, Universität Siegen, 2008.

[welde] `http://www.weltderphysik.de`, letzter Abruf: 21. April 2009.

[wikde] `http://www.wikipedia.de`, letzter Abruf: 21. April 2009.

[wikdeEIA] WIKIPEDIA: *EIA-232.* `http://de.wikipedia.org/wiki/EIA-232`, letzter Abruf: 21. April 2009.

Danksagung

Ich möchte mich an dieser Stelle bei all denen bedanken, die mich bei der Anfertigung meiner Bachelorarbeit unterstützt haben.

Besonders bedanken möchte ich mich bei:

- Herrn Prof. Dr. Ivor Fleck, der als mein Betreuer diese Arbeit erst ermöglicht hat und mir im Anschluss daran die Anstellung als studentische Hilfskraft vermittelte, sodass ich weiter an AMIGA mitwirken kann. Seine Tür stand jederzeit offen, wenn ich auf der Suche nach Antworten und hilfreichen Anregungen war.
- Herrn Prof. Dr. Peter Buchholz und der gesamten Arbeitsgruppe Hochenergiephysik der Universität Siegen für die freundliche Aufnahme in ihre Gruppe.
- Yury Kolotaev, Michael Pontz und Uwe Fröhlich für die enge und gute Zusammenarbeit. Mit ihnen verbrachte ich viele lehrreiche und auch erheiternde Stunden im Labor, wo sie mir diverse Male mit Rat und Tat zur Seite standen.
- Marcus Niechciol und Stefan Gadatsch, mit denen ich vor drei Jahren das Studium begann. Sie schrieben gleichzeitig an ihrer Bachelorarbeit, sodass wir uns gegenseitig hilfreiche Tipps zur Gestaltung des Layouts und Verwendung von Latex geben konnten.
- Thomas Bender, ein guter und langjähriger Freund, der diese Arbeit Korrektur gelesen hat und so einige sprachliche und logische Unstimmigkeiten aufspüren konnte.
- Meinen Eltern für ihre persönliche und finanzielle Unterstützung meines Studiums. Ohne sie wäre es mir nicht ohne Weiteres möglich gewesen, mein Studium aufzunehmen und diese Arbeit zu schreiben.
- Meiner Freundin und ihren Mitbewohnern für ihre unendliche Geduld und ihre Motivation. Neben der großen sprachlichen Hilfe gab mir vorallem meine Freundin stets Rückhalt und zeigte Verständnis, wenn es um meine Arbeit ging.

Vielen Dank!

Eidesstattliche Erklärung

Hiermit versichere ich, dass ich die vorliegende Bachelorarbeit nur unter der Zuhilfenahme der angegebenen Quellen und Hilfsmittel selbstständig angefertigt, sowie Zitate und Ergebnisse Dritter kenntlich gemacht habe.

Siegen, den 21. April 2009 _____

(Unterschrift)