Fernwartung und Verwaltung von Audio-Systemen mit gemeinsamem Takt als Anwendung im Tunnelmonitoring

Diplomarbeit
durchgeführt von

Stefan Leitner

durchgeführt und eingereicht am
Institut für Breitbandkommunikation
an der Technischen Universität Graz

Vorstand: Univ.-Prof. Dr. Gernot Kubin

in Kooperation mit der
Joanneum Research Forschungsgesellschaft mbH

Begutachter: Ao. Univ.-Prof. Dr. Erich Leitgeb (TU Graz)
Betreuer: Dr. Franz Graf (Joanneum Research)

Graz, im Dezember 2010

Kurzfassung

Die Verteilung eines gemeinsamen Systemtaktes an viele Teilsysteme ist in den Ingenieurswissenschaften von erheblicher Bedeutung. Ob in der digitalen Signalverarbeitung, bei Verwendung von Arrays von A/D Konvertern u. Ä. garantiert der Masterclock die Synchronität und Phasenstarrheit zwischen den einzelnen Modulen.
Diese Diplomarbeit beschäftigt sich mit der Möglichkeit der zentralen Steuerung der Teilsysteme über die bestehenden Taktleitungen mit dem Ziel, den Einfluß der Steuerung auf die Taktreinheit des Masterclocks zu minimieren bzw. zu eliminieren.
Hierzu werden traditionelle Möglichkeiten der Taktrückgewinnung (nichtlineare Verzerrung, Entscheidungsrückkopplung) sowie eine Eigenkonstruktion, aber auch digitale Audioschnittstellen und binäre Leitungscodes diskutiert.
Anhand eines bestehenden Systems zur automatischen Überwachung eines Verkehrsweges, wurden die theoretischen Erkenntnisse dann in einer praktischen Ausführung von Hard- und Software geprüft und evaluiert.

Abstract

In engineering the distribution of a common clock signal to various subsystems is of significant relevance. Whether in digital signal processing, or by using arrays of analog-to-digital converters and the like, the masterclock ensures synchronicity and phase-locked states of all the modules.
This master thesis deals with the option of controlling the subsystems centralized via existing clock routes, with a view to minimizing or eliminating the influence of control purposes onto the masterclocks purity.
Therefore traditional approaches to clock recovery (nonlinear distortion, decision feedback coupling) together with a self built, and furthermore digital audio interfaces and binary line codes are being discussed.
By means of an existing system designed for automated traffic surveillancing then the theory gets verified and evaluated in a hard- and software implementation.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Da der Umfang des in den folgenden Kapiteln beschriebenen Systems auf der Diplomarbeit von Markus Koschier [Kos05] basiert, die eine Abhandlung über die synchrone, serielle Übertragung von digitalen Daten über optische Netze darstellt, werden zuerst die Früchte ebendieser Arbeit näher erläutert (Kapitel 1.1).
Aus den sich ergebenden Problemen dieses Aufbaus im praktischen Betrieb leitet sich die Zielsetzung dieser wissenschaftlichen Arbeit ab (Kapitel 1.2).

1.1 Zusammenfassung des bestehenden Systems

Die Erläuterung der Funktionsweise des Systems zum akustischen Tunnelmonitoring (AKUT) erfolgt nachstehend anhand seines speziellen Einsatzes in einem Verkehrstunnel. Der Einsatzbereich beschränkt sich aber keineswegs nur auf diese Anwendung, wird aber aus Gründen der Anschaulichkeit und des hohen Grades des Sicherheitszuwachses zur Illustration gewählt.

1.1.1 Aufbau und Funktionsweise

Die Sensorik und primäre Signalverarbeitung des gesamten AKUT Systems wird von einer Vielzahl von Basissystemen (Abbildung 1) erledigt, welche entlang der zu überwachenden Tunnelröhre angebracht sind.
Die auftretenden Geräusche im Tunnel generieren Mikrophonsignale, diese werden durch den nachfolgenden Mikrophonvorverstärker auf störunempfindlichere Pegel gebracht und an den Transmitter weitergeleitet.
Ausgestattet mit einem $\Sigma\Delta$ A/D Wandler (Cirrus Logic CS 5340) und einem AES-3 Encoder (Cirrus Logic CS 8406) führt der Transmitter die analogen Audiosignale in die digitale Domäne bzw. in ein AES-3 Signal (siehe Kapitel 2.2.1) über.
Essentiell für den Betrieb der zuvor genannten Bausteine ist die Taktversorgung. Zu diesem Zweck versorgt ein Master Clock-Signal *alle* Basissysteme mit einem zentralen Takt, der über den Receiver zum dazugehörigen Transmitter gesendet wird. Da allerdings zur Signalübertragung ein Lichtwellenleiter zum Einsatz kommt, sitzen an besagten Blöcken optische Transceiver, um das Taktsignal elektrisch/optisch und vice versa zu wandeln.
Ein weiterer Lichtwellenleiter wird benutzt, um den AES-3 Datenstrom in die Tunnelzentrale zu leiten. Dort erfolgt wiederum eine optisch/elektrische Umwandlung des Signals und eine Anpassung an die AES-3 Schnittstellenspezifikationen.
Die auf diese Weise erhaltenen digitalen Audiodaten aller Teilsysteme können nachfolgend im AES3 zu ADAT Konverter zusammengeführt und durch das eigentliche Herzstück des AKUT Systems bearbeitet und analysiert werden.

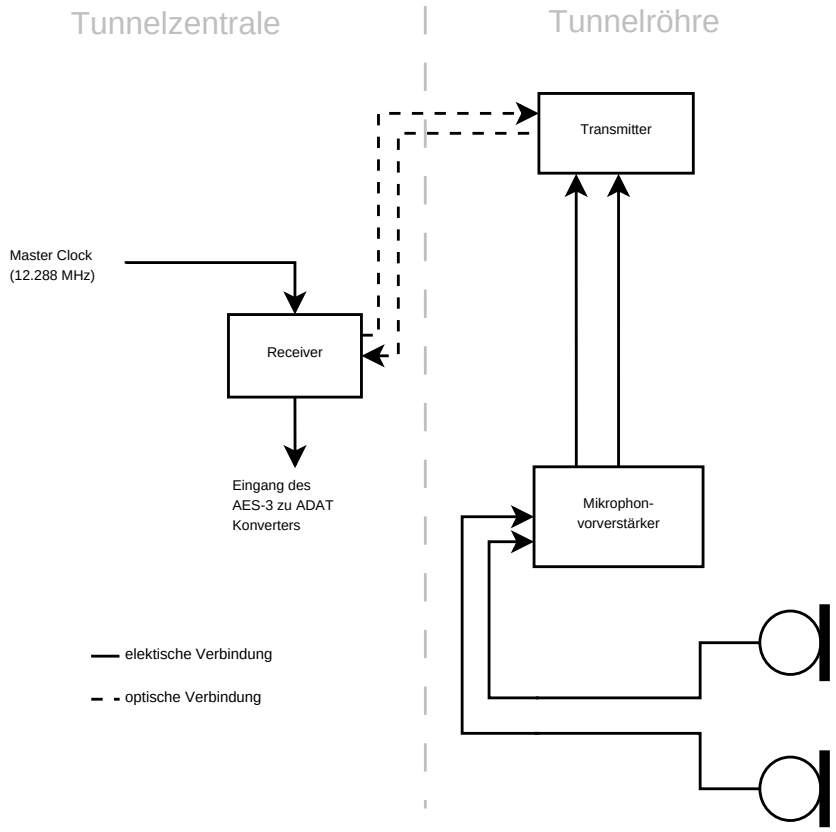


Abbildung 1: Basissystem

1.1.2 Probleme im praktischen Einsatz

Im Testbetrieb des Gesamtsystems zeigte sich, daß es von Zeit zu Zeit passieren kann, daß die verwendeten $\Sigma\Delta$ A/D Konverter ihre Arbeit einstellen (Freezes) oder durch Artefakte das Audiosignal verfälschen. In beiden Störungsfällen hilft ein Reset des Basissystems - dies ist aber nur mittels Taster direkt an der, sich in der Tunnelröhre befindlichen, Transmitterschaltung möglich.
Einen weiteren Stein des Anstoßes stellt die Verstärkung des Mikrophonsignals durch den Mikrophonvorverstärker dar. Aufgrund des zyklisch unterschiedlichen Verkehrsaufkommens zu Nacht und Tage beträgt die Differenz des Schalldruckpegels im Tunnel ca. 60 dB. Es ist also nicht möglich mit einem statischen Verstärkungsfaktor eine optimale Aussteuerung des Einganges des $\Sigma\Delta$ A/D Wandlers zu gewährleisten. Um diesem Umstand entgegenzuwirken, müßte man entsprechend der Höhe des Schalldruckpegels jeden einzelnen Mikrophonvorverstärker in der Tunnelröhre händisch auf eine bestimmte Verstärkung stellen.

1.2 Ziele dieser Diplomarbeit

Es ist augenscheinlich, daß der Bedienkomfort des AKUT Systems unter denen in Kapitel 1.1.2 geschilderten Gegebenheiten leidet.
In den nachfolgenden Abschnitten 1.2.1 und 1.2.2 werden die gewünschten Zielsetzungen dieser Arbeit präzisiert.

1.2.1 Reset via Fernwartung

Anstatt des manuellen Rücksetzens per Taster am Transmitter selbst, wird gefordert, einen willentlichen Reset jedes einzelnen Basissystems unabhängig von allen anderen Teilsystemen aus der Tunnelzentrale heraus zu bewerkstelligen.

1.2.2 Lautstärkenregelung via Fernwartung

Die Lautstärkenregelung soll es ermöglichen, aus der Ferne den momentanen bzw. gewünschten Verstärkungsfaktor des Mikrophonsignals auszulesen bzw. einzustellen. Zusätzlich muß die Option gegeben sein, auch beliebigen Gruppen von Teilsystemen eine bestimmte Verstärkung zuzuweisen oder den bestehenden Faktor um einen vorgegebenen Offsetwert zu erhöhen oder zu vermindern.

2 Theoretische Konzepte

Die in diesem Abschnitt behandelten Theorien und Abhandlungen zielen darauf ab, den Boden, aber auch die Einschränkungen, der Möglichkeiten der praktischen Realisierung und deren zugrundeliegenden Thesen aufzubereiten bzw. aufzuzeigen und verständlich zu machen.

2.1 Binäre Leitungscodes

Als Einführung in die Thematik der Leitungscodes soll am Beginn von Kapitel 2.1 ein Überblick über den Ablauf einer digitalen Nachrichtenübertragung stehen, in der die Leitungscodierung einen wichtigen und wesentlichen Teil einnimmt.
Da aber in dieser Diplomarbeit hauptsächlich Leitungscodes (vor allem lineare) von Bedeutung sind, werden die weiteren Codierverfahren wie Blockcodes oder Faltungscodes, die bei der Kanalkodierung eingesetzt werden, nicht berücksichtigt.
Zusätzlich lassen sich Codes auch durch die Anzahl ihrer darstellbaren logischen Zustände charakterisieren, wobei auch hier die Einschränkung auf zwei Logikzustände getroffen wird.
Der Hinweis auf ternäre, quarternäre, und höherwertigere Leitungscodes soll der Vollständigkeit halber hier aber seine Erwähnung finden.

2.1.1 Digitale Übertragung von Information

In Abbildung 2 ist der prinzipielle Ablaufplan einer digitalen Nachrichtenübermittlung skizziert.

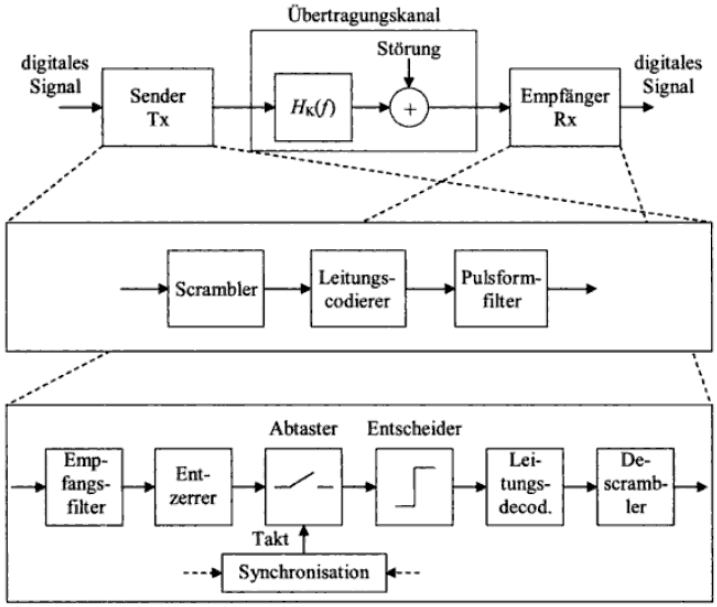


Abbildung 2: Prinzip der digitalen Datenübertragung [Rop06]

Das gewünschte digitale logische Informationssignal, das bereits quellcodiert und kanalcodiert ist, wird an die Sendeeinheit übermittelt und durchläuft dort folgende Stufen:

- Scrambler

  Dieser Baustein besteht zumeist aus einem linear rückgekoppelten Schieberegister und dient dazu, aus dem ursprünglichen Informationssignal eine pseudozufällige Folge von logischen Zuständen zu generieren.
  Dies soll zu langen Datensequenzen ohne Logikwechsel und der daraus folgenden erschwerten Taktrückgewinnung entgegenwirken. Die ursprüngliche Datenfolge ist jederzeit wiederherstellbar, wenn dem Descrambler im Empfängerteil die exakte Methodik der linearen Rückkopplung der Schieberegister bekannt ist.

- Leitungscodierer

  Der Leitungscodierer ist der eigentliche thematische Hauptbestandteil des Kapitels 2.1, dessen Eigenschaften im Laufe dieses Unterabschnittes noch genauer behandelt werden.
  Die Aufgabe dieses Blocks ist es, einer bestimmten Anzahl von einlaufenden Informationsbits ein physikalisches Signalelement wie Spannungspegel, Frequenz oder Phasenbeziehung zuzuordnen, welches dann zum nächsten Element der Kette, dem Pulsformfilter weitergeleitet wird. Im Spezialfall der binären Leitungscodes entspricht dies einer bitweisen Umformung des logischen Digitalsignals.

- Pulsformfilter

  Um nicht Rechteckpulse über den Kanal zu schicken, die eine sehr große, ideal gesehen unendliche Bandbreite aufweisen, wird das leitungscodierte Signal einer Pulsformung unterzogen. Dadurch wird die Bandbreite beschränkt, die Darstellung des Pulses im Zeitbereich aber aufgeweitet.
  Typische Pulsformfilter sind Raised-Cosine Filter, Gauß Filter oder Thomson Filter.

Es folgt die Übertragung des Signals über den Kommunikationskanal, der einerseits signalverzerrend wirkt, d.h. Amplituden-, Frequenz,- und Phaseninformation verändert und andererseits zusätzliche Störungen auf die Information aufbringt.

In der Empfangseinrichtung werden die Veränderungen des Senders am Signal mittels der folgenden Systemblöcke wieder rückgängig gemacht:

- Empfangsfilter

  Soll nur die entsprechende Informationsbandbreite aufweisen, um unerwünschte Einstreuungen am Übertragungskanal weitestgehend zu unterdrücken.

- Entzerrer

  Wie erwähnt besitzt der Kanal nicht ideale Übertragungseigenschaften, sondern verzerrt (bestenfalls linear) die übertragenen physikalischen Signalelemente.
  Die Entzerrerung des Signals, die dadurch notwendig wird, erledigt dieser Block.

- Abtaster

  Tastet das entzerrte Signal zu den, durch den Takt vorgegebenen, Zeitpunkten ab (z.B. Sample & Hold Element).

- Entscheider

  Entspricht einem A/D Wandler, welcher aus der durch die physikalische Übertragung wertkontinuierlichen Sequenz wieder ein Digitalsignal generiert.

- Leitungsdecodierer

  Umformung des leitungscodierten Signals in einen logischen Bitstrom.

- Descrambler

  Rückgängigmachen der Signalmanipulation durch den Scrambler in der Sendeeinheit. Diese erfolgt wie beim Scrambler selbst ebenso mittels linear rückgekoppelter Schieberegister.
  Danach ist das ursprüngliche Signal, das in die Sendeeinheit gespeist wurde, zur weiteren Informationsverarbeitung verfügbar.

2.1.2 Beeinflußbare Signaleigenschaften durch Benutzung von Leitungscodes

Durch das Verwenden von Leitungscodes bei der digitalen Nachrichtenübertragung ist es möglich Signale so zu manipulieren, daß sie den nachstehenden Kriterien entsprechen.

- Einfache Taktrückgewinnung

  Speziell bei synchroner serieller Datenübertragung ist es wünschenswert einen Datenstrom zu empfangen, aus dem das Taktsignal möglichst einfach und genau rekonstruierbar ist (nähere Ausführungen hierzu sind in Kapitel 2.3 zu finden).
  Diese Bedingung ist dann gegeben, wenn regelmäßige Pegelwechsel im Empfangssignal vorhanden sind und damit auch die entsprechenden Spektralanteile, die weiters zur Taktrückgewinnung herangezogen werden können.

- Spektrale Formung des Informationssignals

  Durch geeignete Wahl des Leitungscodes ist es möglich, die spektrale Verteilung des zu übertragenden Signals zu beeinflussen.
  Dies ist in Abbildung 3 dargestellt, wobei der Ordinatenparameter T der Bitzellenperiode bzw. dem Zeitintervall zwischen zwei Abtastvorgängen, in Sekunden entspricht.

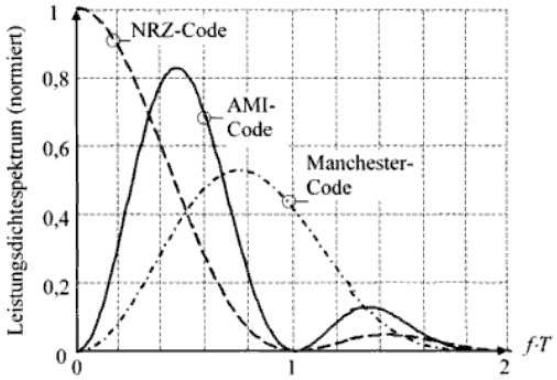


Abbildung 3: Spektrale Leistungsdichte einiger ausgewählter Leitungscodes [Wer09]

Die spektrale Formung ist auch von enormer Wichtigkeit, wenn sich mehrerer Systeme einen Übertragungskanal via Frequency-Division-Multiple-Access teilen.
Dann ist es wünschenswert, daß der Informationsstrom jedes Nutzers ein kompaktes Spektrum aufweist, um die Aufteilung der Frequenzbänder zu erleichtern, ein Übersprechen auf die Bänder der weiteren Teilnehmer zu minimieren und die Datenrate bei gegebener Bandbreite zu maximieren (Bandbreiteneffizienz).

Ein spezieller Fall der spektralen Formung ist das Entfernen des Gleichanteils aus dem Sendesignal. Es folgen einige Szenarien bei denen dies erforderlich ist:

- Der Kanal kann keine Gleichsignalanteile übertragen

  Zum Beispiel bei kapazitiver oder induktiver Kopplung (galvanische Trennung) im Übertragungspfad (siehe Abb. 4), oder bei der Speicherung digitaler Daten auf magnetischen Medien.

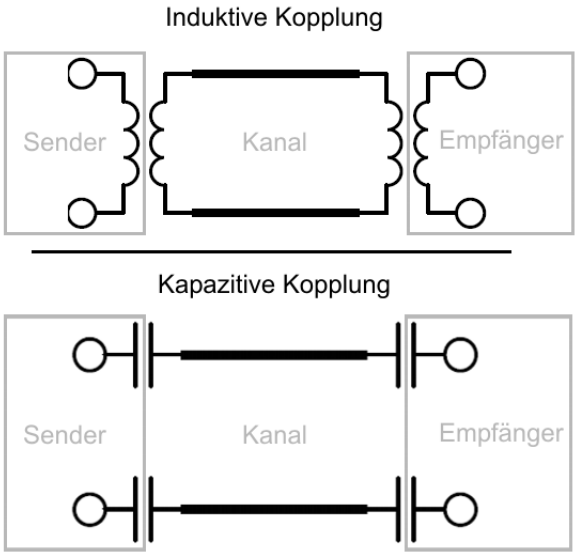


Abbildung 4: Möglichkeiten zur galvanischen Trennung von Sender und Empfänger [Rup05]

- Der Gleichanteil des Kanalspektrums wird anderwertig verwendet

  Bei längeren Übertragungsstrecken werden oftmals Regenerativverstärker (Repeater) eingesetzt, um das durch den Kanal verschliffene Signal wieder aufzubereiten und damit die maximal erreichbare Übertragungsdistanz zu vervielfachen.
  Ist das Informationssignal nun DC-frei, so ist es möglich die Betriebsspannung dieser Repeater über das DC-Spektrum des Übertragungskanals bereitzustellen.

- Übertragung des Gleichanteils ist prinzipiell möglich, aber unerwünscht

  Da der Aufwand der Entzerrung bei tiefsten Frequenzen erheblichen Filteraufwand bedeutet, kann im Vorfeld dieses Problem durch Verwendung eines DC-freien Leitungscodes umgangen werden.

2.1.3 Mögliche Kategorisierung von Leitungscodes

Die nachstehende Abbildung 5 soll die verschiedenen Kategorien von Leitungscodes visualisieren, die in den Unterkapiteln 2.1.3.1 bzw. 2.1.3.2 näher erläutert werden.

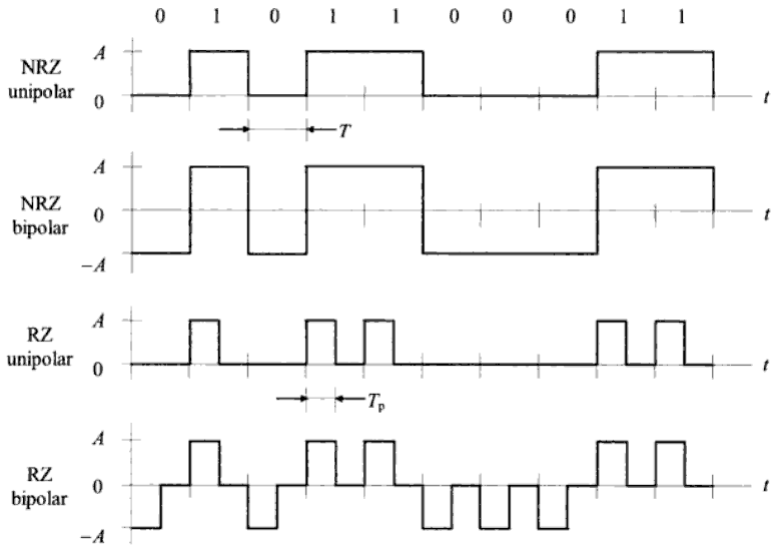


Abbildung 5: Kategorien von Leitungscodes [Rop06]

2.1.3.1 Nach der Impulsdauer T_P

Die Impulsdauer T_P eines Pulses ist die Zeitdauer, während der die Signalamplitude innerhalb einer logisch 1 repräsentierenden Bitzelle konstant bleibt, bevor diese wieder auf den Entsprechungswert für logisch 0 zurückfällt (vgl. Abbildung 5).
Durch diese Klassifizierung können Leitungscodes folgendermaßen eingeteilt werden:

- Non-Return-To-Zero (NRZ) Codes

  Die Pulsdauer T_P entspricht genau der Bitzellenperiode T

- Return-To-Zero (RZ) Codes

  Die Pulsdauer T_P des physikalischen Signalelementes ist kleiner als die Bitzellenperiode T d.h. die Signalamplitude wird innerhalb der Bitzelle wieder zu logisch 0.

2.1.3.2 Nach der Anzahl der physikalischen Zustände

Eine weitere Kategorisierung von Leitungscodes läßt sich durch die Unterscheidung der Anzahl von Verläufen der Signalamplitude erreichen:

- unipolare Codes

  Es gibt einen Verlauf der Signalamplitude zur Darstellung der Informationssymbole.

- bipolare Codes

  Zwei Verläufe der Signalamplituden stehen zur Bildung des physikalischen Symbols zur Verfügung.

- höherpolare Codes

Je mehr Signalamplitudenverläufe zur Codierung der Information zur Verfügung stehen, desto aufwendiger und damit auch teurer gestalten sich die Sende- und Empfangsgeräte.
Durch gezielte Wahl der Leitungscodes nach diesen Kriterien ist es möglich, die gesetzten Prioritäten bei der Informationsübertragung zu präzisieren, z.B. Detektion einer Unterbrechung des Kommunikationskanals durch Verwendung bipolarer Datenübertragung oder Sicherstellung der immanenten Pegelwechsel zur Taktrückgewinnung durch Verwendung bipolarer NRZ-Codes.

2.1.4 Ausgewählte Leitungscodes

Eine Auswahl an erprobten und verbreiteten Leitungscodes sowie die mit ihnen verbundenen Eigenschaften, Vor- und Nachteile sollen im diesem Kapitel vorgestellt werden.

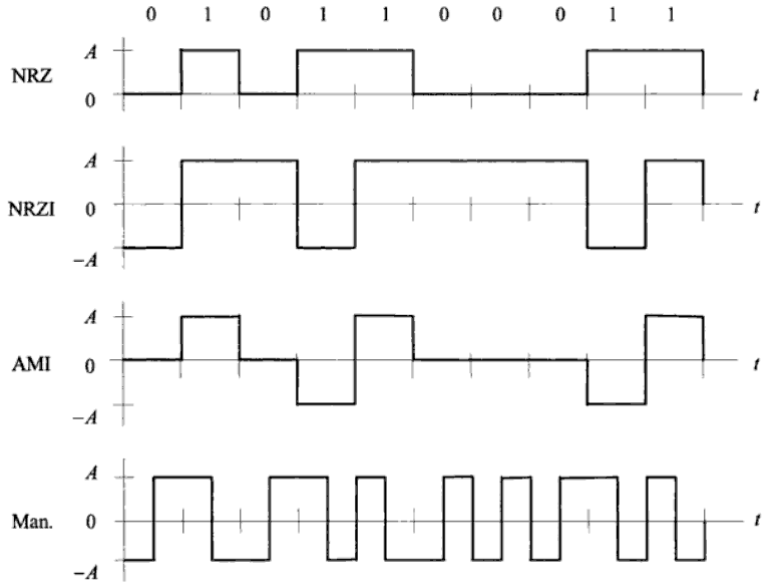


Abbildung 6: Darstellung augewählter Leitungscodes [Rop06]

2.1.4.1 Unipolarer NRZ Code

(siehe Abb. 5 bzw. Abb. 6)

logisch 1 $\rightarrow$ Amplitude A; logisch 0 $\rightarrow$ Amplitude 0

Vorteil: Sehr einfacher, minimalistischer Code

Nachteile: Der Code ist nicht zur Taktrückgewinnung geeignet, wenn sehr lange Sequenzen von logisch 0 oder 1 vorhanden sind. Dieser Nachteil kann durch einen Scrambler (Kapitel 2.1.1) kompensiert werden.
Das codierte Signal weist einen Gleichspannungsanteil auf.

Unipolare NRZ Codes werden meist für interne Übertragungen in Geräten ergo kurze, unproblematische Entfernungen benutzt.

2.1.4.2 Bipolarer NRZ Code

(siehe Abb. 5)

logisch 1 $\rightarrow$ Amplitude A; logisch 0 $\rightarrow$ Amplitude $-A$

Vorteile: Möglichkeit zur Detektion einer Störung des Kommunikationskanals (Signalamplitude von Null entspricht einem Ausfall)
Verringerung des Gleichspannungsanteils gegenüber dem unipolaren NRZ Code

Nachteile: Siehe unipolarer NRZ Code

2.1.4.3 NRZ-I / NRZ-Mark Code

(die bipolare Variante ist in Abb. 6 dargestellt)

logisch 1 $\rightarrow$ Amplitudenwechsel; logisch 0 $\rightarrow$ kein Amplitudenwechsel

Vorteile: Bei Vertauschung der Leitungsenden am Empfänger gibt es keine Umkehrung der logischen Bitfolge.
Siehe je nach Ausführung unipolarer bzw. bipolarer NRZ Code

Nachteile: Siehe unipolarer NRZ Code

Der NRZ-Mark Code gehört zur Familie der differenziellen Codes (Mark Codes), d.h. daß die Information nicht in den absoluten physikalischen Amplitudenverläufen gespeichert ist, sondern in den Änderungen ebendieser.

2.1.4.4 Unipolarer RZ Code

(siehe Abb. 5)

logisch 1 $\rightarrow$ Amplitude A; logisch 0 $\rightarrow$ Amplitude 0

Vorteil: Bezogen auf NRZ Codes enthält das unipolar RZ codierte Signal die doppelte Anzahl an Pegelwechseln was die Taktrückgewinnung erleichtern kann, aber für lange Nullsequenzen trotzdem problematisch bleibt.

Nachteile: Verbreiterung des Signalspektrums verglichen mit dem NRZ Pendant durch die kürzere Pulsdauer $(T_p < T)$
Vorhandener Gleichspannungsanteil

2.1.4.5 Bipolarer RZ Code

(siehe Abb. 5)

logisch 1 $\rightarrow$ Amplitude A; logisch 0 $\rightarrow$ Amplitude $-A$

Vorteile: Taktrückgewinnung ist ohne Einschränkungen möglich
Möglichkeit zur Detektion einer Störung des Kommunikationskanals (Signalamplitude von Null entspricht einem Ausfall)
Verringerung des Gleichspannungsanteils gegenüber dem unipolaren RZ Code

Nachteile: Siehe unipolarer RZ Code

2.1.4.6 Manchester Code

Dieser Code, auch Biphase Code genannt, definiert sich über eine Signalflanke in der Bitzellenmitte. Allerdings finden sich in der Literatur unterschiedliche Definitionen, ob die logische 1 als negative oder positive Flanke gewertet wird (vgl. Manchester vs. Manchester II/Biphase L).

Vorteile: Bei bipolarer Ausführung, wie in Abbildung 6 ganz unten gezeigt, ist dieser Code vollkommen frei von Gleichsignalen.
Die Taktrückgewinnung ist durch die Signalflanke in der Mitte der Bitzelle garantiert.
Sehr einfacher Codier- bzw. Decodiervorgang der unipolaren Ausführung durch EX-OR Verknüpfung des zu codierenden Logiksignals bzw. des manchestercodierten Signals mit dem Taktsignal selbst.

Nachteil: Erhöhter Bandbreitenbedarf (vgl. Abbildung 3)

Der Manchester Code findet unter anderem Verwendung bei der Definition des physikalischen Layers von Ethernet (IEEE 802.3).

2.1.4.7 AMI Code

(siehe Abb. 6)

logisch 1 $\rightarrow$ abwechselnd Amplitude A oder $-A$; logisch 0 $\rightarrow$ Amplitude 0

Dieser Code weist 2 logische, aber 3 physikalische Zustandformen auf und wird deshalb zu den pseudoternären Codes gezählt.

Vorteile: Minimierung des vorhandenen Signalgleichanteils
Spektral effizienter als z.B. Manchestercodierung (vgl. Abb. 3)

Nachteile: Die Taktrückgewinnung ist bei langen Nullfolgen nicht gewährleistet. Deshalb wird die AMI Codierung meist mit einem Scrambler (Kapitel 2.1.1) kombiniert.
Erhöhte Gerätekosten für Sende- und Empfängereinheit, um drei physikalisch unterschiedliche Signalformen zu erzeugen bzw. zu detektieren

Eine Abgewandelte Form des AMI Codes findet praktischen Einsatz im ISDN Protokoll bei der Realisierung des ISDN-S0 Busses.

2.2 Digitale Audiosschnittstellen

Bei der Vorstellung des AKUT Systems in Kapitel 1.1 sind schon Stichworte wie AES-3 und ADAT, bezüglich der Übermittlung der digitalen Audiosignale, gefallen, die nun in diesem Abschnitt genauer unter die Lupe genommen werden.
Zusätzlich zu den beiden vorhergenannten digitalen Audioschnittstellen soll dieser Abschnitt auch noch Informationen über einige andere etablierte Standards bereitstellen.

2.2.1 AES-3 (AES/EBU)

In den Anfängen der 80er Jahre machte es sich die Audio Engineering Society (AES) und die European Broadcasting Union (EBU) zur Aufgabe ein modernes Protokoll zur Übertragung von digitalen Audiodaten zu entwickeln.
Die Zielsetzungen dieses Prozesses waren

- die Übertragung von 24-Bit Stereo Audiodaten und aller relevanten Signale über eine einzelne Datenleitung mit bereits etablierten Verbindungssteckern.
- längere Verbindungsstrecken zwischen den angeschlossenen Geräten zu ermöglichen.
- zusätzliche Informationen wie z.B. Samplingfrequenz, Vorverzerrung, Timecode der Audiodaten, Housekeeping, Gerätesteuerung etc. in diesem Interface zu berücksichtigen.
- preiswerte Sende- und Empfangseinheiten.

Das daraus resultierende digitale Audiointerface AES-3 gliedert sich in die folgenden Bereiche:

2.2.1.1 Electrical Layer

Der Electrical Layer definiert die physikalischen Eigenschaften des AES-3 Übertragungsstandards, dessen zugehörigen Komponenten wie folgt festgelegt sind:

- Codierung

  Zum Einsatz kommt ein Derivat der differentiellen Manchester Codegruppe, namentlich der Biphase-Mark Code (Abb. 7) der sich so definiert:

  - Signalflanken treten immer bei Bitzellenanfang auf
  - Zusätzliche Signalflanken in der Bitzellenmitte = logisch 1
  - Keine Signalflanke in der Bitzellenmitte = logisch 0

  Die Eigenschaften, Vor- und Nachteile von Manchester Codes und differentiellen Codes sind in Abschnitt 2.1.4 ausführlich erklärt.

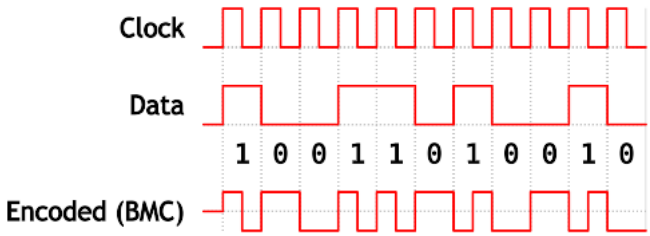


Abbildung 7: Darstellung der Biphase-Mark Codierung [Ale07]

- Verbindungskabel und Stecker

  Laut AES-3 Standard wird eine symmetrische Signalübertragung über Twisted Pair Leitungen empfohlen, welche wiederum mit, in der Studioaudiotechnik sehr verbreiteten, XLR3 Steckern ausgestattet sind.
  Außerdem wird durch induktive Kopplung (Kapitel 2.1.2) die galvanische Trennung von Sender und Empfänger über Trenntrafos gewährleistet.

- Signalamplitude und Terminierung

  Die Wellenimpedanz jedes Kanalteilnehmers ob Sender, Leitung oder Empfänger soll 110 Ω (±20% Toleranz) betragen.
  Die Signalamplitude darf 3,5 V nicht übersteigen und die minimalen Voraussetzungen Jitter und Amplitude betreffend sind durch das folgende Eye Pattern gegeben.

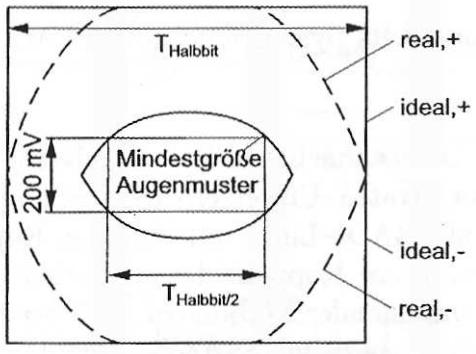


Abbildung 8: Eye Pattern des physikalischen AES-3 Interfaces [Gra06]

- Bandbreite und Entzerrung

  Bei Signalübertragung über längere Distanzen tritt unweigerlich eine frequenzabhängige Dämpfung des elektrischen Signals auf, die es, durch die in Abbildung 9 gezeigte Entzerrungskurve, zu korrigieren gilt. Die relevante Bandbreite des AES-3 Signals erstreckt sich von 0,1 bis 6 MHz.

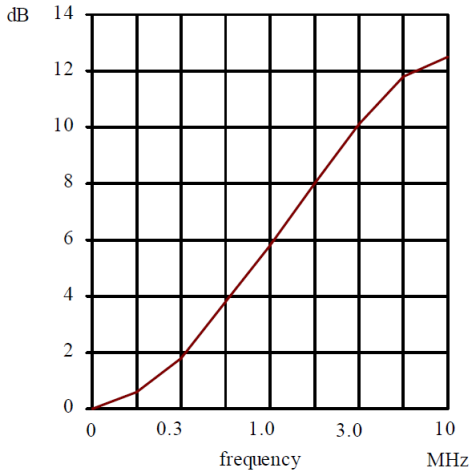


Abbildung 9: Empfohlene Entzerrungskurve [EBU95]

- Präambeln

  Diese Signalwörter dienen der Identifikation der Elemente des Data Layers und werden durch doppelte Verletzung der Biphase-Mark Code Eigenschaften herbeigeführt.

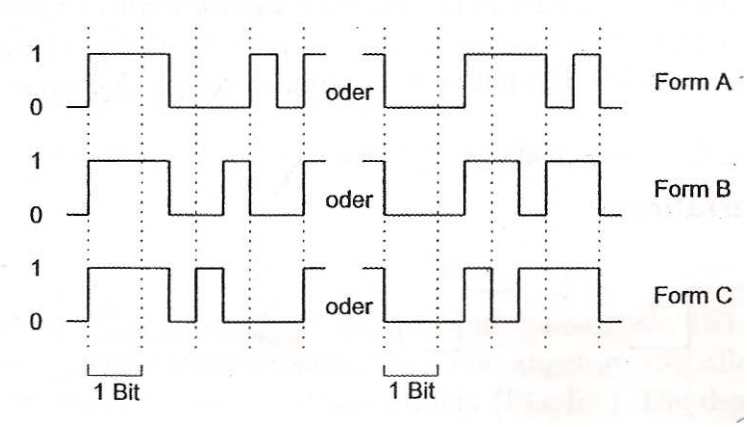


Abbildung 10: Präambeln des AES-3 Standards [Gra06]

Abbildung 10 zeigt die drei unterschiedlichen Strukturen (Form A, B und C), wobei die Doppeldeutigkeit der Darstellungen daher rührt, daß die Polarität bei den differentiellen Codes nicht ausschlaggebend ist (vgl. NRZ Code vs. NRZ-Mark Code in Kapitel 2.1.4).

2.2.1.2 Data Layer

Der Data Layer definiert die Zugehörigkeit der Symbole des Electrical Layers zu den logischen Entitätstypen des AES-3 Übertragungsstandards, die wie folgt festgelegt sind:

- AES-3 Subframe:

  Der AES-3 Subframe umfaßt 32 Informationsbits, welche zu den nachstehend erläuterten Informationsblöcken zusammengefaßt werden.
  Die Illustration eines solchen Subframes ist in Abbildung 11 dargestellt.

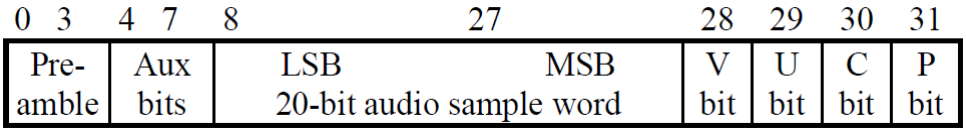


Abbildung 11: Struktur des AES-3 Subframes (32-Bit) [EBU95]

  - Preamble

    Enthält eine der drei gültigen Präambeln, die bei der Beschreibung des Electrical Layers vorgestellt wurden:

    * Form A leitet einen Subframe für den Audiokanal A ein
    * Form B definiert einen Subframe für den Audiokanal B
    * Form C beginnt einen Subframe für den Audiokanal A eines neuen AES-3 Blocks .

  - Auxiliary Bits

    Können einerseits zur Erhöhung der Bittiefe der Audiodaten dienen oder andererseits Audiodaten eines qualitativ geringwertigeren Audiokanals, z.B. eines Voice Channels, enthalten.

  - Audio Data

    Pulse Code moduliertes, digitales Signal des jeweiligen Audiokanals

  - Ancillary Data (V, U, C, P Bits)

    Stellen zusätzliche Informationen wie z.B. Samplingfrequenz, Vorverzerrung, Timecode der Audiodaten, Housekeeping, Gerätesteuerung etc. bereit.
    Eine genauere Beschreibung hierzu findet sich in der Sektion über den Control Layer (Kapitel 2.2.1.3).

- AES-3 Frame: Besteht aus 2 AES-3 Subframes für jeweils Audiokanal A und B
- AES-3 Block: Besteht aus 192 AES-3 Frames

2.2.1.3 Control Layer

Der Control Layer, auch Channel Status Information genannt, wird benutzt, um angeschlossene Audiogeräte über das AES-3 Interface zu steuern.
Die für diese Aufgabe zur Verfügung stehenden Bits sind:

- Validity Bit (V Bit)

  Bietet die Möglichkeit die Audiodaten des Subframes als nicht geeignet zur Analogwandlung zu markieren, indem man es auf logisch 1 setzt.

- User Bit (U Bit)

  Steht zur freien Verfügung des Benutzers

- Channel Status Bit (C Bit)

  Wird über einen gesamten AES-3 Block hinweg zwischengespeichert und realisiert die 192-Bit (24 Byte) Housekeeping Information jedes der beiden Audiokanäle.
  Die einzelnen Parameter für den professionellen Einsatz (erstes Bit von Byte 0 wird auf logisch 1 gesetzt) sind in Abbildung 12 aufgelistet.

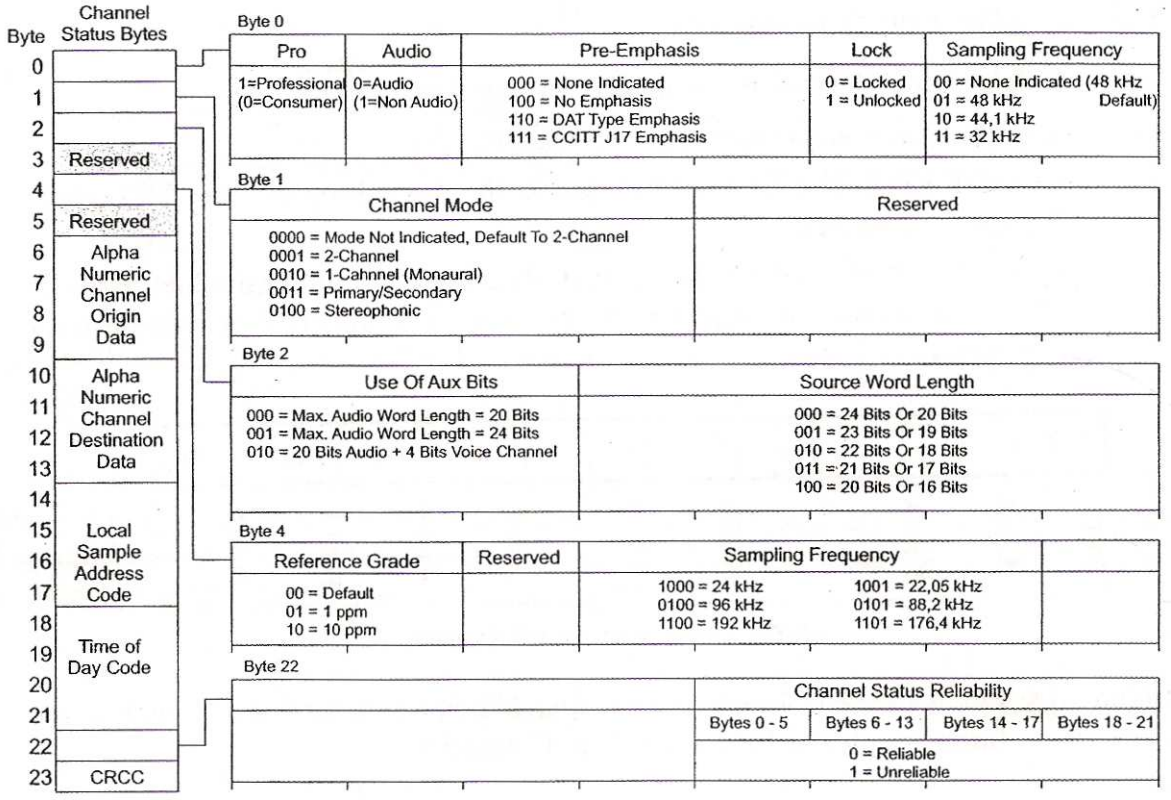


Abbildung 12: Channel Status Parameter für den professionellen Einsatz [Gra06]

- Parity Bit (P Bit)

  Dieses wird so gesetzt, daß im Subframe eine gerade Anzahl von logischen Einsen vorherrscht und dient somit zur simplen Detektion von Fehlern bei der Übertragung des Subframes.

2.2.2 SPDIF

Das Sony Philips Digital Interface (SPDIF) ist eine abgewandelte Form des AES-3 Standards und wird vorwiegend im Consumer Bereich zur digitalen Audioübertragung bei MiniDisc-Playern, DVD-Playern, Home-Theater Komponenten, Soundkarten und anderen am Massenmarkt orientierten Systemen eingesetzt.

2.2.2.1 Electrical Layer

Die folgenden Unterschiede in der Definition des Electrical Layers zwischen dem AES-3 Übertragungsstandard und SPDIF betreffen:

- Verbindungskabel und Stecker

  Zur elektrischen Übertragung, die bei SPDIF unsymmetrisch erfolgt, werden Koaxialkabel verwendet, die wiederum mit handelsüblichen Cinch Steckern abgeschlossen sind.
  Die optische Variante umfaßt einen Lichtwellenleiter mit mechanisch nicht gesicherten TOSLINK Steckverbindungen.

- Signalamplitude und Terminierung

  Die Vorgaben den Wellenwiderstand der SPDIF nutzenden Geräte betreffend, wurden für Sender, Koaxialkabel und Empfänger auf 75 Ω festgelegt.
  Besagte Geräte arbeiten mit einer Signalamplitude von 0,25 V, was zur Folge hat, daß auf der Übertragungsstrecke die Signaldämpfung maximal 6 dB betragen darf, um die Vorgaben des Eye Patterns (Abb. 8) zu erfüllen. Dies beschränkt die Verwendung von SPDIF auf kurze Verbindungen.

2.2.2.2 Control Layer

Die Festlegungen des Control Layers von SPDIF unterscheiden sich von denen des AES-3 Übertragungsstandards durch den Informationsinhalt des C Bits.
Zur Verwendung im Consumer Bereich (erstes Bit von Byte 0 wird auf logisch 0 gesetzt) ergeben sich die Einstellmöglichkeiten in Abbildung 13. Der Großteil der Channel Status Bytes bleibt zu Kopierschutzzwecken reserviert.

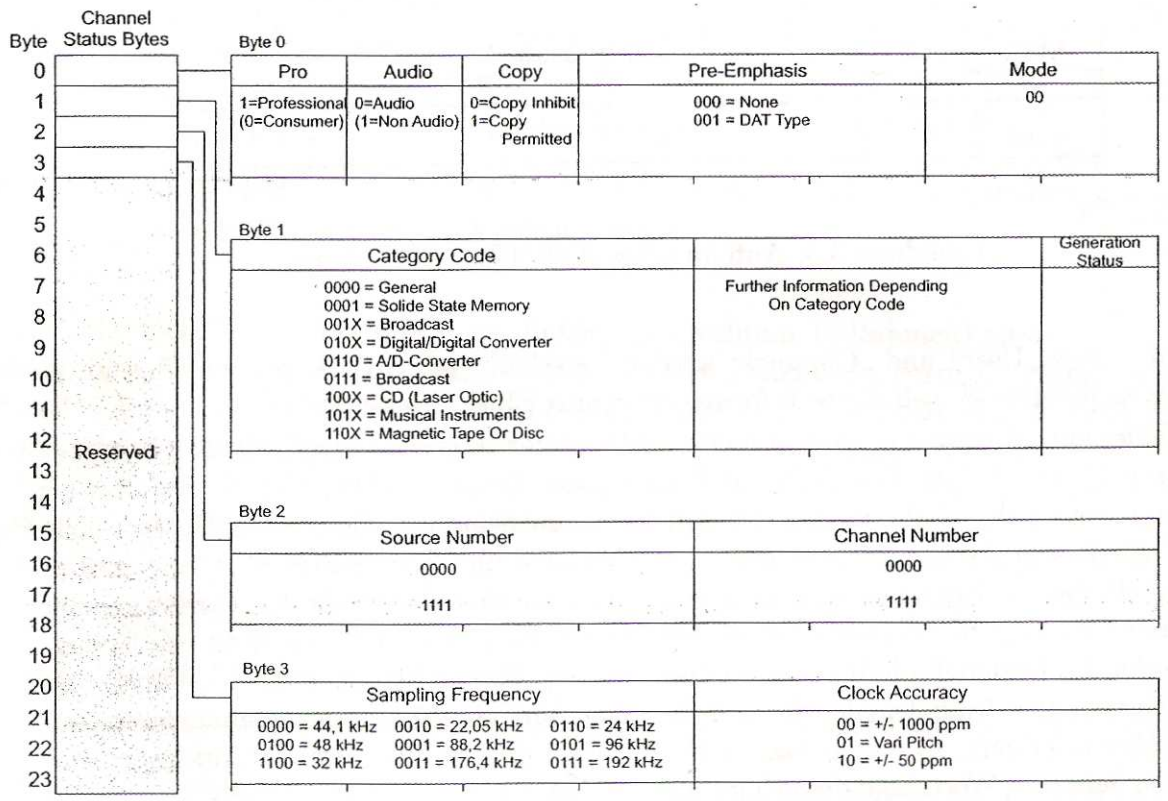


Abbildung 13: Channel Status Parameter für den Consumer Bereich [Gra06]

2.2.3 MADI (AES-10)

In audiotechnisch professionellen Umgebungen (Tonstudios, Live Anwendungen) sind meist mehr als zwei Kanäle in Verwendung, und damit steigt auch der entsprechende Verkabelungsaufwand für eine Vielzahl von Audiosignalen nach dem AES-3 Standard. Diesem Problem nimmt sich das Multichannel Audio Digital Interface (MADI) an, das ein Containerformat für AES-3 darstellt d.h. es werden 56 Audiokanäle (AES-3 Subframes) in einem MADI Frame konzentriert.

MADI subframe	0	1	2	3	4					54	55	0
Audio channel	Ch 0	Ch 1	Ch 2	Ch 3						Ch 54	Ch 55	Ch 0
Sample number	n	n	n	n						n	n	n + 1
AES3 subframe	A	B	A	B						A	B	A

Abbildung 14: Darstellung eines MADI Frames mit 56 aktiven Kanälen [AES03]

Zur Definition der Subframes des MADI Formats werden die AES-3 Auxiliary Daten (4-Bit) dem Audiosignal zugewiesen, und die 4-Bit langen Präambeln am Beginn jedes AES-3 Subframes gemäß Abbildung 15 in MADI Informationsbits umgewidmet.

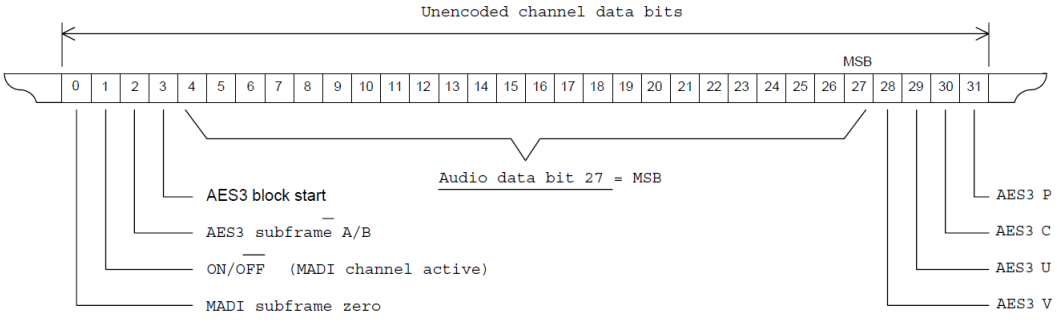


Abbildung 15: Darstellung eines MADI Subframes [AES03]

- MADI Subframe Zero

  Wenn dieses Bit logisch 1 gesetzt ist bedeutet dies den Beginn eines neuen MADI Frames

- MADI Channel Active

  Setzt für den digitalen Audiokanal den entsprechenden Status fest. Falls es gewünscht ist den Kanal inaktiv zu schalten, sollten auch die restlichen Bits des Subframes auf Null gesetzt werden.
  Weiters wird empfohlen die Numerierung der Audiokanäle fortlaufend zu gestalten und die Kanalnummer der inaktiven Signale höher zu wählen als die des letzen aktiven Kanals.

- AES-3 Subframe

  Dadurch, daß die Erkennung der AES-3 Subframes nun nicht mehr über die Präambel erfolgen kann, wird diese Information in das Subframe Bit verpackt:

  Logisch 0 = Subframe A
  Logisch 1 = Subframe B

- AES-3 Block Start

  Dient, gemeinsam mit dem AES-3 Subframe Bit, zur Erhaltung der Kompatibilität zu den Blockdefinitionen im Data Layer des AES-3 Standards.

Bit 2	Bit 3	Two-Channel Form	Description
0	0	Form 2	A subframe
0	1	Form 1	A subframe status block start
1	0	Form 3	B subframe
1	1	Form 4*	B subframe status block start

*Does not conform to AES3.

Abbildung 16: Entsprechungen des AES-3 Subframe bzw. des AES-3 Block Start Bits [AES03]

Die restlichen Bits (20-Bit Audio Data, V, U, C, P Bits) sind dem AES-3 Protokoll gemäß zu verwenden.

Die Verbindung zwischen den MADI Geräten erfolgt unsymmetrisch über 75 Ω (±2 Ω) Koaxialkabel mit BNC Verbindungsbuchsen oder als zweite Option über einen Lichtwellenleiter nach ISO/IEC 9314-3.
Nachstehend sind die einzuhaltenden Eye Pattern Vorgaben bei der Signalübertragung abgebildet.

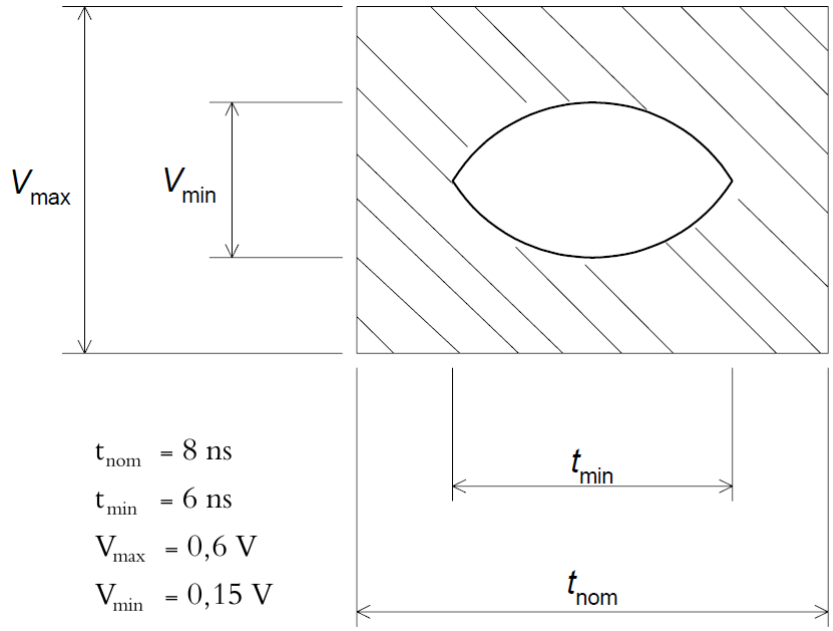


Abbildung 17: MADI Eye Pattern Vorgaben [AES03]

Weitere Vorgaben wie zum Beispiel die 4B5B Leitungscodierung und die Form der 4B5B Präambel sind in [AES03] nachzulesen.
Es sei nur noch erwähnt, daß bei Verwendung der Variante mit Koaxialkabel eine zentrale, externe Taktversorgung zur Datenübertragung vorhanden sein muß, also zusätzlicher Verkabelungsaufwand besteht.
Bei der Übertragung über Lichtwellenleiter hingegen wird am Empfänger der Takt aus dem 4B5B codiertem Datenstrom rückgewonnen.

2.2.4 ADAT Optical Interface (ADAT Lightpipe, ADAT)

Das Alesis Digital Audio Tape (ADAT) Optical Interface ist ein herstellerspezifischer Standard der Firma Alesis, der auch von anderen Marktmitbewerbern übernommen wurde und dient der optischen Übertragung von 8 Audiokanälen (mit max. 24-Bit Auflösung) über Lichtwellenleiter mit TOSLINK Steckverbindungen bzw. bei manchen Herstellern auch über das Firewire Interface.
Dieses digitale Audiointerface wird manchmal fälschlich auch als ADAT bezeichnet, was eigentlich einen Standard zur Aufzeichnung von digitalen Audiodaten auf Tonband bezeichnet, der von der gleichen Firma entwickelt wurde.
Ein ADAT Lightpipe Frame folgt den Gesetzmäßigkeiten in Tabelle 1.

Bit	Funktion / Belegung
0-4	Sync / Logisch 0
5	Sync / Logisch 1
6	User Bit 3 / Logisch 0
7	User Bit 2 / Logisch 0: fs = 48 kHz; Logisch 1: fs = 96 kHz
8	User Bit 1 / Zur freien Verfügung
9	User Bit 0 / ADAT Timecode
10	Sync / Logisch 1
11-14	Audiokanal 1, BIT 0-3
15	Sync / Logisch 1
16-19	Audiokanal 1, BIT 4-7
20	Sync / Logisch 1
21-24	Audiokanal 1, BIT 8-11
25	Sync / Logisch 1
26-29	Audiokanal 1, BIT 12-15
30	Sync / Logisch 1
31-34	Audiokanal 1, BIT 16-19
35	Sync / Logisch 1
36-39	Audiokanal 1, BIT 20-24
40-69	Abfolge 10-39 für Audiokanal 2
70-99	Abfolge 10-39 für Audiokanal 3
100-129	Abfolge 10-39 für Audiokanal 4
130-159	Abfolge 10-39 für Audiokanal 5
160-189	Abfolge 10-39 für Audiokanal 6
190-219	Abfolge 10-39 für Audiokanal 7
220-249	Abfolge 10-39 für Audiokanal 8
250	Sync / Logisch 1
251-255	Sync / Logisch 0

Tabelle 1: Aufbau eines ADAT Lightpipe Frames

Bei Verwendung von Audiodaten mit einer niedrigeren Auflösung als 24-Bit werden die überflüssigen LSBs einfach ignoriert.
Die Taktrückgewinnung erfolgt mithilfe der im ADAT Lightpipe Frame verstreuten Sync Information.
Zur Leitungscodierung wird ein einfaches NRZ-I / NRZ-Mark Schema (Kapitel 2.1.4) herangezogen. Die Diskrepanz der Zuordnung der Sync Bits 0-4 und 251-255 in der Literatur (vgl. [Alb10], [Gra06], [Kos05]) ist auf diesen Leitungscode zurückzuführen, da nur ein Pegelwechsel im Signal zu detektieren ist, wenn eine logische 1 codiert wurde.
Relevant ist, daß zwischen zwei ADAT Lightpipe Frames 10 logische Nullen vorhanden sind, aber nicht deren Zugehörigkeit zu den jeweiligen Frames.

2.2.5 TDIF

Das TASCAM Digital Interface Format (TDIF) übernimmt die Vorgaben des AES-3 Data Layers und kombiniert 8 bidirektionale Audiokanäle (8 in / 8 out) zu einem Leitungsstrang. Die elektrische Übertragung erfolgt unsymmetrisch und über parallele gebündelte Datenleitungen mit D-Sub Steckverbindungen (25 Pin).
Zur Leitungscodierung kommt ebenso wie bei ADAT Lightpipe der NRZ-I / NRZ-Mark Code zum Einsatz. Im Unterschied zu ADAT Lightpipe und MADI werden aber die Taktsignale und einige andere Steuerparameter (vgl. Tabelle 2) im gebündelten Kabel selbst mitübertragen.
Diese Steuerparameter sind:

L/R Clock: Zur Kennzeichnung der AES-3 Subframes A oder B des entsprechenden Audiokanals

FS0, FS1: Zusätzliche Indikatoren zur Signalisierung der verwendeten Samplingrate

Emphasis: Information über die eventuelle Vorverzerrung des codierten Audiosignals

Pin	Funktion
1	OUT: AES-3 Channels 1/2
2	OUT: AES-3 Channels 3/4
3	OUT: AES-3 Channels 5/6
4	OUT: AES-3 Channels 7/8
5	OUT: L/R Clock
6	OUT: FS0
7	GND
8	IN: FS0
9	IN: Left/Right Clock
10	IN: AES-3 Channels 7/8
11	IN: AES-3 Channels 5/6
12	IN: AES-3 Channels 3/4
13	IN: AES-3 Channels 1/2
14	GND
15	GND
16	GND
17	GND
18	OUT: Emphasis
19	OUT: FS1
20	IN: FS1
21	IN: Emphasis
22	GND
23	GND
24	GND
25	GND

Tabelle 2: Pinbelegung der D-Sub Verbindung lt. TDIF-1 Version 1.1

2.3 Taktrückgewinnung

Um ein empfangenes Datensignal richtig interpretieren zu können, ist es unter anderem auch nötig, daß der genaue Systemtakt des Senders bekannt ist.
Eine Variante wäre, daß man zu den entsprechenden Daten einen separaten Pilotton mitsendet, um somit die frequenz- und phasenrichtige Abtastung zu garantieren.
Durch entsprechende Codierung des Datensignals (vgl. Kapitel 2.1.2) läßt sich andererseits erreichen, den Takt direkt aus der empfangenen Sequenz abzuleiten und somit auf die dezidierte Übermittlung des Pilottones zu verzichten. Diese Vorgehensweisen, auch Taktrückgewinnung genannt, soll des Weiteren für den Spezialfall der digitalen, zweiwertigen, linearen Modulationen beleuchtet werden.
Hierzu stehen im Wesentlichen zwei Methoden zur Verfügung, die in den Unterabschnitten 2.3.1 und 2.3.2 näher erläutert werden.

2.3.1 Nichtlineare Verzerrung des Signals

Die erste Möglichkeit zeichnet sich durch eine nichtlineare Verzerrung des Signals aus und läßt sich wiederum in zwei Bereiche aufspalten, die in den Kapiteln 2.3.1.1 und 2.3.1.2 beschrieben sind.

2.3.1.1 Generieren taktrelevanter Spektralkomponenten

Die Modellierung einer mit linearer Modulation, z.B. Amplitude Shift Keying (ASK), Phase Shift Keying (PSK), Quadrature Amplitude Modulation (QAM), versehene, gescrambelten Signalquelle sei in Abbildung 18 dargestellt.

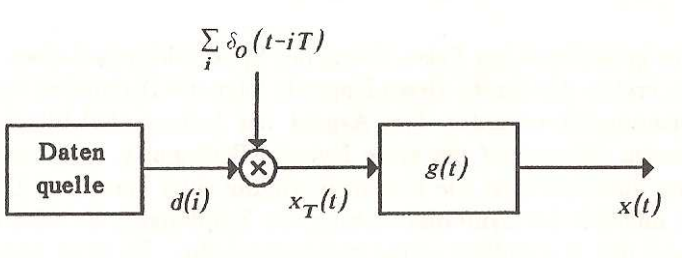


Abbildung 18: Erzeugung der Pulsfolge $x(t)$[Kam92]

Die mathematische Ausformung der zu übertragenden Pulsfolge $x(t)$ aus Abbildung 18 lautet:

$$x(t) = \left(\sum_{i=-\infty}^{\infty} d(i)\, \delta_0(t - iT) \right) * g(t) \quad (1)$$

$$= \sum_{i=-\infty}^{\infty} d(i)\, g(t - iT) \quad (2)$$

$d(i)$ ..... zufälliges, zeitdiskretes, redundanzfreies Datensymbol mit $m_d = E\{d(i)\}$ und $\sigma_d^2 = E\left\{(d(i) - m_d)^2\right\} = 1$
(zweiwertige Modulation$\rightarrow d(i) \in \mathbb{C}^2$, z.B.$\{-1,\, 1\}, \left\{i,\, (1-i)\,\frac{1}{\sqrt{2}}\right\}, \ldots$)

$g(t)$ ..... Impulsantwort des Impulsformers, abhängig von der verwendeten Modulationsart

T ..... zeitliche Periode zwischen zwei Abtastvorgängen

Die spektrale Leistungsdichte die daraus folgt (genaue Herleitung in [Kam92]) ist:

$$\begin{aligned} S_{x\,x}(j\omega) &= \mathcal{F}\{AKF(x(t))\} = \\ &= \frac{\sigma_d^2}{T}\,|G(j\omega)|^2 \;+\; 2\pi\left(\frac{m_d}{T}\right)^2 \sum_{\lambda=-\infty}^{\infty} \left|G(j\frac{2\pi}{T}\lambda)\right|^2 \delta_0(\omega + \frac{2\pi}{T}\lambda) \end{aligned} \tag{3}$$

Im zweiten Summanden in Gleichung 3 wird ersichtlich, daß wenn $d(i)$ nicht mittelwertfrei ist, unterschiedlich gewichtete Frequenzenlinien bei $\frac{\lambda}{T}$ auftreten, und so eine Relation zur Abtastperiode besteht - vorausgesetzt $G(j\frac{2\pi}{T}\lambda)$ ist an diesen Stellen nicht null.

Allerdings können diese nützlichen, spektralen Indizien verschwinden oder gedämpft werden durch

- verwendete Empfangsfilter
- die Impulsformung selbst, wenn dessen Bandbreite sich auf max. $\frac{1}{T}$ beschränkt bzw. die Pulsform zeitlich länger als T ausgedehnt ist
- Verwendung von Rechteckpulsen der Dauer T (Nullstellen im Spektrum an Vielfachen von $\frac{1}{T}$)
- Mittelwertfreiheit der Datenfolge $d(i)$

Mittels nichtlinearer Verzerrung, z.B. Quadrierung von $x(t)$

$$y(t) = x^2(t) = \sum_{i=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} d(i)\; d(l)\; g(t - iT)\; g(t - lT) \tag{4}$$

und anschließender Berechnung des Leistungsdichtespektrums ([Kam92])

$$S_{y\,y}(j\omega) = \frac{2\pi}{T^2} \sum_{\lambda=-\infty}^{\infty} S_{g^2\,g^2}(j\frac{2\pi}{T}\lambda)\; \delta_0(\omega - \frac{2\pi}{T}\lambda) \;+\; \frac{2}{T} \sum_{\substack{\lambda=-\infty \\ \lambda\neq 0}}^{\infty} S_{g_\lambda\,g_\lambda}(j\omega) \tag{5}$$

$S_{g^2\,g^2}(j\frac{2\pi}{T}\lambda)$ ... Leistungsdichtespektrum von $g^2(t)$ ausgewertet an dem Stelle $f = \frac{\lambda}{T}$

$g_\lambda(t)$ ..... $g(t)\; g(t - \lambda T)$

$S_{g_\lambda\,g_\lambda}(j\omega)$ ..... Leistungsdichtespektrum von $g_\lambda(t)$

ergeben sich im ersten Summanden in Gleichung 5 die bereits bekannten, diskreten Linienspektren an den Frequenzen $\frac{\lambda}{T}$, nun aber mit dem Leistungsdichtespektrum $S_{g^2\,g^2}(j\frac{2\pi}{T}\lambda)$ gewichtet.
Als zweiter Summand trägt noch ein kontinuierlicher Verlauf abhängig von $S_{g_\lambda\,g_\lambda}(j\omega)$ zum Verlauf des gesamten Leistungsdichtespektrums bei. Dieser Beitrag wird umso geringer, je kleiner die Überlappung von $g(t)$ mit $g(t-\lambda T)$ ist und verschwindet vollkommen bei Impulsformen die zeitlich kürzer als T sind.

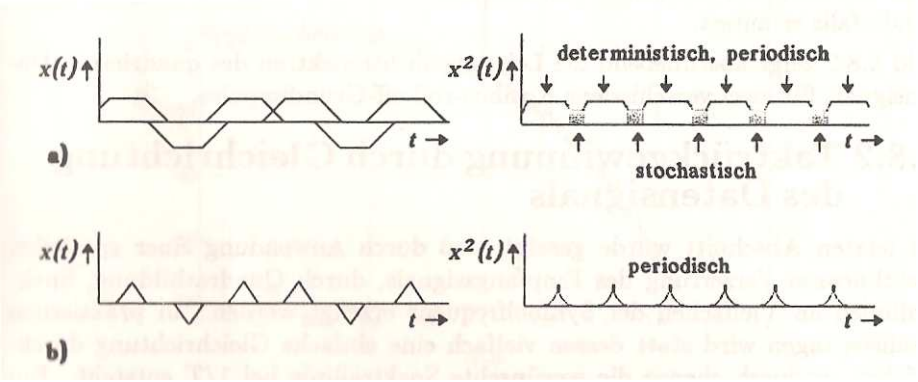


Abbildung 19: Quadrieren von $x(t)$ bei Verwendung unterschiedlicher Impulsformen $g(t)$ [Kam92]

Im Unterschied zu den spektralen Regelmäßigkeiten im linearen Fallbeispiel (Gleichung 3) sind diesselbigen in Gleichung 5 nur mehr von den Gegebenheiten der Autokorrelationsfunktion von $g^2(t)$ abhängig, und nicht mehr zusätzlich vom Mittelwert der Datenfolge von $d(i)$.
Allerdings bleibt das Problem, welches bei Verwendung von Rechteckpulsen mit der Länge T auftritt. In der mathematischen Ausformung in Gleichung 5 werden alle Beiträge $S_{g^2\,g^2}(j\omega)$ an den Stellen $f=\frac{\lambda}{T}$ Nullstellen aufweisen und somit nicht zur Ausprägung von Spektrallinien beitragen.
Dieser Umstand ist auch graphisch einfach zu veranschaulichen (vgl. Abb. 20 Fall a und b). Durch Quadrierung oder Betragsbildung des empfangenen Signals bleibt nur mehr eine konstante Einhüllende übrig und jegliche Taktinformation geht verloren.

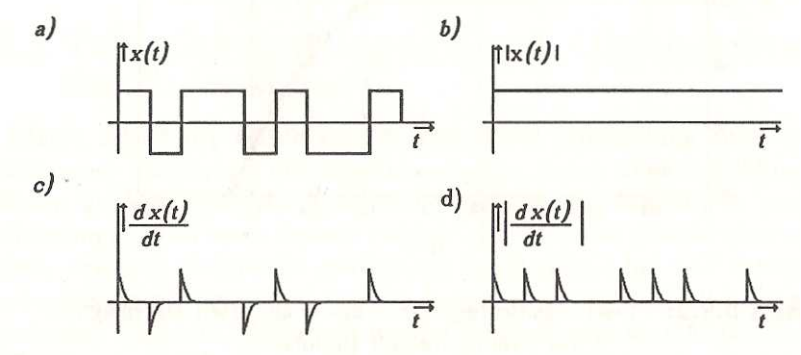


Abbildung 20: Gewinnung taktrelevanter Spektralkomponenten bei Impulsen der Länge T [Kam92]

Um dieses Szenario auszuschließen, wird vor der nichtlinearen Verzerrung eine zeitliche Ableitung des Signals durchgeführt und auf diese Weise die taktrelevanten Signalanteile beibehalten (siehe Abb. 20 Fall c und d).

2.3.1.2 Verarbeitung taktrelevanter Spektralkomponenten

Eine Möglichkeit das Taktsignal zu rekonstruieren ergibt sich durch Verwendung eines Bandpaßfilters, dessen Mittenfrequenz bei der Frequenz des Systemtaktes f_0 liegt (passive Taktrückgewinnung). Hierzu werden meist akustische Oberflächenwellenfilter (SAW Filter) verwendet, die eine sehr hohe Güte ($Q = f_0/3\,dB\;Bandbreite$) aufweisen und mit Mittenfrequenzen bis in den GHz-Bereich herstellbar sind.
Der Takt, der auf diese Weise zurückgewonnen wird, ist allerdings nicht phasenstarr zum ursprünglichen Sendetakt, was bedeutet, daß diese Methode nicht zur Taktversorgung für synchronisierte Einzelbausteine geeignet ist.

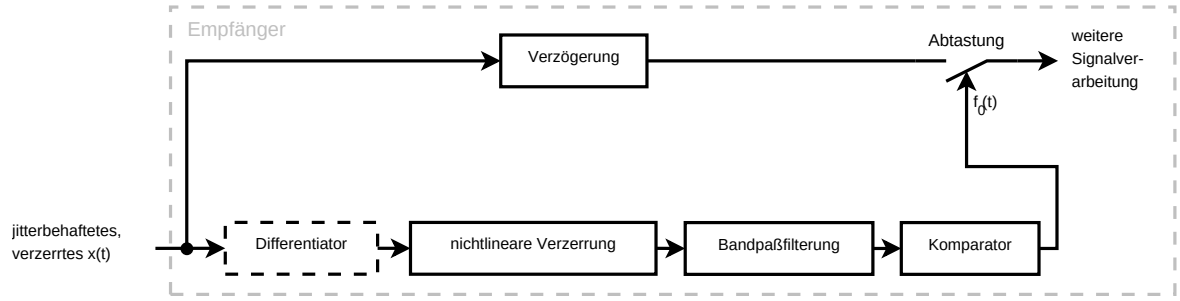


Abbildung 21: Passive Taktrückgewinnung

Eine zweite Möglichkeit besteht im Einsatz einer Phase-Locked Loop (PLL) welche in Abbildung 22 dargestellt ist.

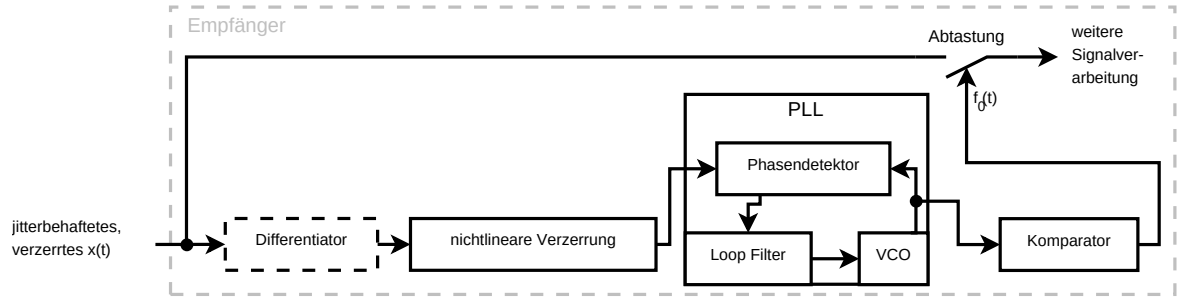


Abbildung 22: Aktive Taktrückgewinnung

Die entscheidenden Unterschiede zur passiven Variante sind, daß die Triggerung auf die Pulsmitte nicht durch ein konstantes Verzögerungsglied garantiert werden muß, und weitaus wichtiger, daß durch den Einsatz der PLL die Phasenänderungen des ursprünglichen Taktsignals mit denen des, im Voltage Controlled Oscillators (VCO) generierten, Systemtaktes - bis auf das zusätzliche Phasenrauschen des VCOs - übereinstimmen.
Allerdings bleibt bei Verwendung von zweiwertigen, linearen Modulationen, die absolute Phasenlage des Taktes durch die Quadrierung des Signals mit einer Unsicherheit von $\pm\pi$ behaftet.
Aus diesem Grund werden zur eindeutigen Identifikation des Systemtaktes die abgewandelten Differenzcodierungsverfahren DQAM, DBPSK verwendet, welche ohne die Kenntnis der absoluten Phasenlage korrekt arbeiten.

2.3.2 Entscheidungsrückgekoppelte Taktregelung

Diese Art der Taktrückgewinnung basiert auf der ersten Nyquistbedingung, d.h. auf der Annahme, daß bei ungestörter Datenübertragung keine Intersymbolinterferenz (ISI) auftritt.
Bebildert wird dieses erwünschte Verhalten in Abbildung 23, die den idealen Zeitverlauf eines einzelnen Datenpulses veranschaulicht, dessen Nullstellen sich an Vielfachen von $t = T$ befinden und dessen Information am Pulsmaximum ($t = 0$) extrahierbar ist.

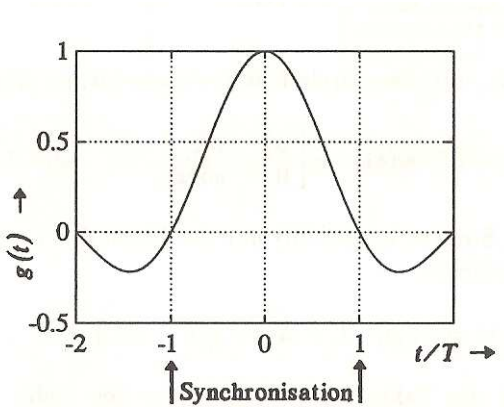


Abbildung 23: ISI freier Datenpuls [Kam92]

Auch bei Übertragung von aufeinanderfolgenden Datenpulsen bleibt dieses Pulsmaximum ungestört, da die folgenden Datenpulse ebenso an Vielfachen von $t = T$ Nullstellen besitzen.
Nach Übersendung der Datenfolge über einen entsprechenden Kanal sieht das Empfangssignal aber keineswegs wie in Abbildung 23 aus, und muß erst mittels adaptiver Entzerrung in diesen Idealzustand übergeführt werden. Erst dann kann mit der entscheidungsrückgekoppelten Taktrückgewinnung begonnen werden, deren Grundlage es ist, aus der Lage der Nullstellen bei $t = \pm T$ den Systemtakt abzuleiten. Das Prinzipschaltbild ist in Abbildung 24 veranschaulicht.

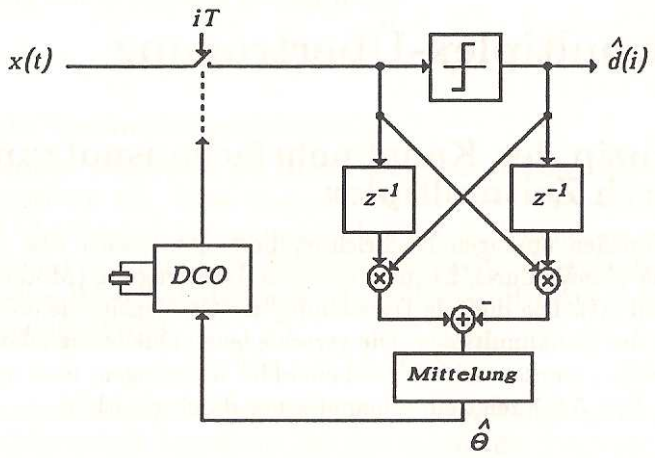


Abbildung 24: Prinzipschaltbild zur entscheidungsrückgekoppelten Taktregelung [Kam92]

Die mathematischen Ausformungen dazu sind:

$$\Theta \;=\; E\left\{\left(x\left(i-1\right)T\right)\hat{d}(i) - x(iT)\,\hat{d}(i-1)\right\} \tag{6}$$

Θ ..... Regelgröße zur Ansteuerung des Oszillators (DCO, VCO, etc.)

$x(iT)$ ..... Momentanwert des übertragenen Pulsverlaufes zum aktuellen Abtastzeitpunkt iT

$x((i-1)\,T)$ ... Momentanwert des übertragenen Pulsverlaufes zum vorigen Abtastzeitpunkt $(i-1)T$

$\hat{d}(i)$ ..... Datensymbol am Ausgang des Entscheiders während der Abtastperiode i

$\hat{d}(i-1)$ ..... Datensymbol am Ausgang des Entscheiders während der vorigen Abtastperiode $i-1$

$$x(iT) \;=\; \sum_{l=-\infty}^{\infty} d(l)\; g(iT + \Delta t - lT) \tag{7}$$

$d(l)$ ..... tatsächliches, zum Zeitpunkt l vom Sender übertragenes, redundanzfreies Datensymbol mit $m_d = E\left\{d(i)\right\} = 0$ und $\sigma_d^2 = E\left\{d(i)^2\right\}$

$g(...)$ ..... Impulsantwort des Impulsformers im Sender, abhängig von der verwendeten Modulationsart

Δt ..... zeitlicher Fehlbetrag zum wahren Abtastzeitpunkt iT

Bei korrekter Abtastung zum Zeitpunkt $i = l$ ergibt sich das Datensymbol $d(l) = d(i) = \hat{d}(i)$.

Durch Einsetzen von Ausdruck 7 in Gleichung 6 und linearer Näherung um den Arbeitspunkt $\pm T$ (siehe [Kam92]) ergibt sich folgender Ausdruck für Θ

$$\Theta = \sigma_d^2 \left(\frac{dg(t)}{dt}|_{-T} \;-\; \frac{dg(t)}{dt}|_{T}\right)\; \Delta t \tag{8}$$

Das heißt die Regelgröße für den Oszillator verhält sich proportional zum zeitlichen Fehlbetrag des wahren Abtastzeitpunktes. Auf dieser Basis läßt sich der, der Datenübertragung zugrundeliegende, Systemtakt phasenrichtig rekonstruieren.
Allerdings erfolgt durch die Notwendigkeit zur Mittelung in Abbildung 24 (der wahre Erwartungswert in Formel 6 kann nur durch unendlich viele Datenpulse realisiert werden) eine Verschmierung des Jitterverhaltens des ursprünglichen Taktsignals.

2.4 Serial Peripheral Interface

Ursprünglich verwendete Motorola erstmals das Serial Peripheral Inferface, das auf dem Master/Slave Übertragungsschema basiert, zur synchronen seriellen Kommunikation zwischen ihren selbst entwickelten ICs. Dieses Interface wurde aber nach und nach seiner Unkompliziertheit und Lizenzfreiheit wegen auch von anderen Herstellern aufgegriffen.
Die einzig vorherrschende Spezifikation betrifft den Electrical Layer des Protokolls, und auch dieser ist sehr offen formuliert, sodaß es notwendig ist die jeweiligen Datenblätter der zu kombinierenden Bausteine auf ihre Kompatibilität hin zu vergleichen.
Abbildung 25 illustriert die grundsätzliche Zusammenschaltung zweier Systeme via SPI, wobei anzumerken ist, daß es keine Konventionen über die Länge der Schieberegister oder deren LSB/MSB Anordnung gibt.

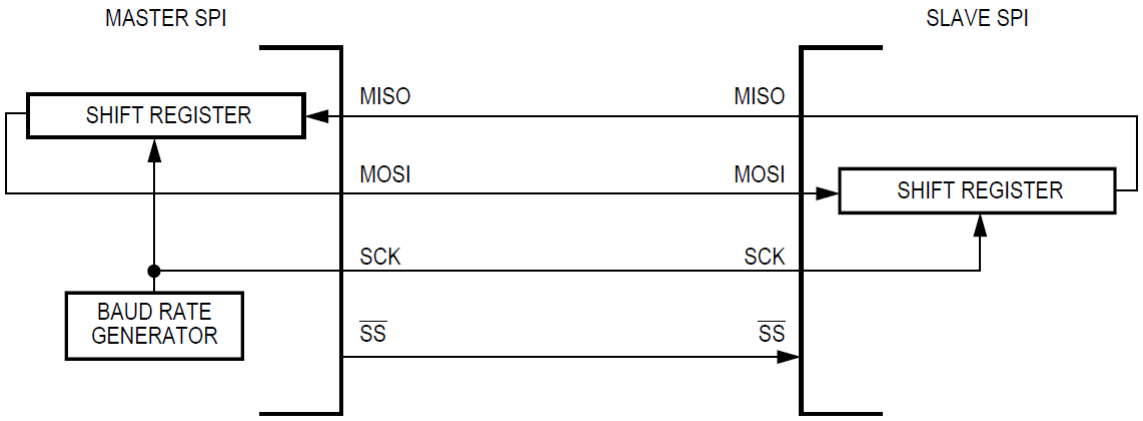


Abbildung 25: Prinzip der SPI Datenübertragung [Fre03]

2.4.1 Signalleitungen

Bei der Kommunikation zwischen zwei Systemen über das Serial Peripheral Interface werden die nachstehenden Signale benötigt:

- Serial Clock ($SCLK$, CLK, SCK)

  Das für die digitale Nachrichtenübertragung zur Synchronisation bzw. zur Abtastung des seriellen Datenstroms benötige Taktsignal.

- Serial Master Out/Slave In (SDI, DI, SI, $MOSI$, $SIMO$)

  Signalleitung die entsprechend dem Namen, den Schieberegisterausgang des Masters und den Dateneingang des Slaves miteinander verbindet.

- Serial Master In/Slave Out (SDO, DO, SO, $MISO$, $SOMI$)

  Rückleitung die an den Dateneingang des Masterbausteins und den Datenausgang des Slaves gelegt wird und für die Weiterleitung dessen Schieberegisterinhaltes sorgt.

- Chip Select ($\overline{CS}$, CS, CSB, SS, $\overline{SS}$, STE)

  Steuerleitung des Masters zur Aktivierung des Slavebausteins, dessen $MISO$ Port andernfalls in den HI-Z Zustand versetzt ist (der Nutzen des HI-Z Verhaltens wird in Kapitel 2.4.3 offensichtlich).
  Die Pegel der Chip Select Leitung besitzen ebenfalls keine eindeutige Bestimmung ob ihrer Bedeutung. Beide Zustände 'active high' oder 'active low' sind je nach verwendetem Baustein legitim.

Aus der Anzahl der Signalleitungen leitet sich auch die Bezeichnung '4 wire serial bus' als Synonym für SPI ab.

2.4.2 SPI Modi

Da weder die Polarität des Taktsignals bei Inaktivität noch dessen Flankenorientierung für den Abtastzeitpunkt (Bitzellenmitte) des Datenstroms definiert sind, ergeben sich folgende Optionen für die Clock Polarity ($CPOL$, $CKPOL$) und die Clock Phase ($CPHA$, $CKPHA$):

- Das Taktsignal ist bei Inaktivität logisch 0 ($CPOL$=0) bzw. logisch 1 ($CPOL$=1)
- Der logische Pegel des Taktes ist in der ersten Hälfte der Bitzelle gleich der Clock Polarity ($CPHA$=0) bzw. das logische Gegenstück der Clock Polarity ($CPHA$=1)

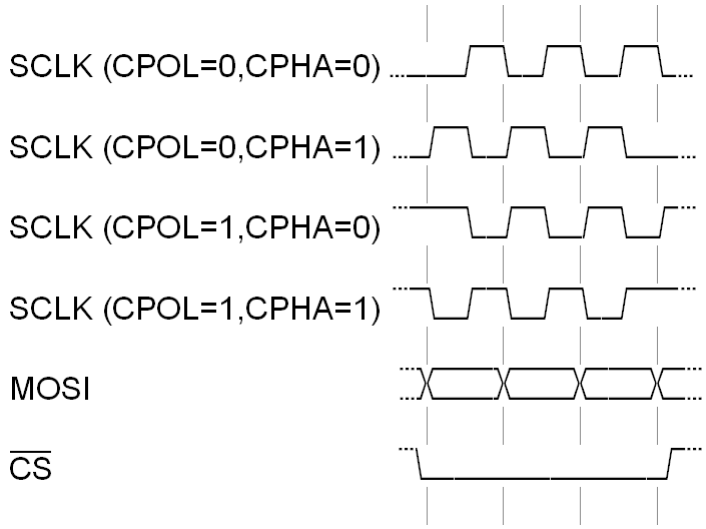


Abbildung 26: Signalverläufe bei gültiger Datenübertragung über SPI

Aus Abbildung 26, die eine gültige Übertragung von 3 Datenbits im entsprechenden Modus darstellt, leiten sich die 4 Betriebsmodi (Tabelle 3) ab, die bei den meisten aktuellen Mikrokontrollern anwählbar sind.

MODUS	CPOL	CPHA
0	0	0
1	0	1
2	1	0
3	1	1

Tabelle 3: SPI Betriebsmodi

2.4.3 Kaskadierung von mehreren Slave Bausteinen

Weiters besteht die Möglichkeit bei Bedarf über einen SPI Master mehrere Slave Geräte zu konfigurieren und zu verwalten. Die Unterschiede, die dadurch in der Kommunikationsstruktur auftreten, werden nachfolgend geklärt.

2.4.3.1 Autonome Verschaltung

Die erste Variante entspricht einer sternförmigen Struktur (Abb 27). Dazu ist für jeden der angeschlossenen, unabhängig voneinander operierenden, Slaves eine dezidierte Steuerleitung ausgeführt ($\overline{SS1}$, $\overline{SS2}$, $\overline{SS3}$) und deren $MOSI$ wie auch deren $SCLK$ Ports werden gemeinsam mit dem entsprechenden Master Ausgang verbunden .
In dieser Vernetzung wird auch die HI-Z Fähigkeit der $MISO$ Slaveports ersichtlich, da diese Ausgänge zusammengeschlossen und gemeinsam an den Master zurückgeführt werden.

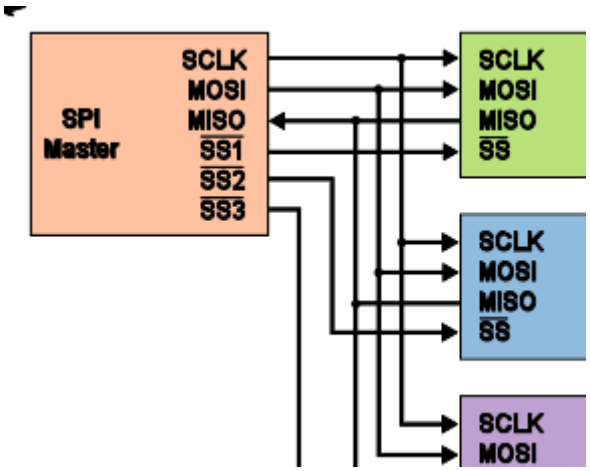


Abbildung 27: Single Master Multiple Slaves Konfiguration [Cbu06]

2.4.3.2 Daisy Chaining

In der zweiten Form bilden mehrere gleichartige Slaves (sofern durch die verwendeten Module vorgesehen) einen gemeinsamen Verband (Abb. 28) und werden dadurch auch nur mehr über eine einzige Chip Select Steuerleitung ($\overline{SS}$) dirigiert.
Weiters sind die $MISO$ Ports der Slaves mit den jeweiligen $MOSI$ Ports der nachfolgenden Slaves

verbunden, wobei der letzte direkt dem Master rückgeführt wird.
Vergleicht man nun Abbildung 28 mit Abbildung 25 so ist in beiden Bebilderungen diesselbe Kommunikationsstruktur ersichtlich, bis auf den Unterschied, daß sich die Länge des internen Slave-Schieberegisters mit deren Anzahl multipliziert hat. Dementsprechend länger müssen auch die $SCLK$ bzw. $\overline{SS}$ Signale aktiviert bleiben.

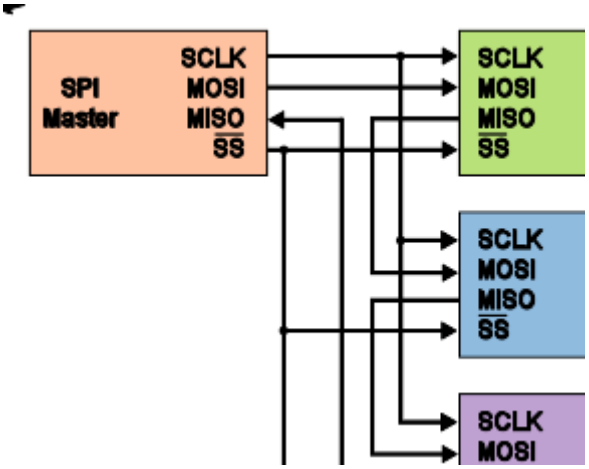


Abbildung 28: Single Master Multiple Slaves Daisy Chaining Konfiguration [Cbu06]

2.4.4 Verwendung mehrerer Master

Bei einem eventuellen Einsatz mehrerer Masterbausteine (z.B. zwei oder mehr Mikrocontroller) ist zu beachten, daß keinerlei Kollisionsabfragen und Error Handling durch SPI zur Verfügung gestellt werden und dadurch Gewissheit nur durch das Studium und den Vergleich der Datenblätter der verschiedenen Hersteller einkehrt.

3 Praktische Realisierung

Mit dem, in Abschnitt 2 erläuterten, theoretischen Fundament soll unter Berücksichtigung der Zielsetzungen des Kapitels 3.1 die entsprechende Realisierung eines Konzepts zur Behebung der geschilderten Probleme aus Kapitel 1.1.2 erreicht werden. Die Erklärung dieser Grundsatzidee erfolgt in Sektion 3.2.
Außerdem gibt es in Kapitel 3.3 eine Überprüfung der am Markt befindlichen kommerziellen Lösungen zur Bewältigung von Teilen dieser Problemstellung.
Dominiert wird dieser Hauptabschnitt jedoch vor allem von der Verwaltungssoftware des Gesamtsystems bzw. von den entworfenen Hardwarebausteinen (vgl. Abschnitt 3.4 bzw. 3.5).

3.1 Prioritäten der Umsetzung

Die Realisierung der Erweiterung des AKUT Systems wurde unter Berücksichtigung der nachfolgenden Punkte durchgeführt:

- Kompatibilität zum bestehenden System

  Der existierende und in Kapitel 1.1 präsentierte Aufbau überzeugte im praktischen Betrieb. Deshalb wurde nach einem Weg gesucht die zusätzlichen geforderten Funktionen (siehe Abschnitt 1.2) mittels Erweiterung des bestehenden Systems umszusetzen.

- Taktreinheit

  Unabdingbar für das korrekte Funktionieren des verwendeten $\Sigma\Delta$ A/D Wandlers ist unter anderem ein sauberes Taktsignal.
  Die aus dem zugehörigen Datenblatt entnommenen Vorgaben sind in folgender Abbildung dargestellt.

CIRRUS LOGIC® CS5340

SWITCHING CHARACTERISTICS - SERIAL AUDIO PORT (Logic "0" = GND = 0 V; Logic "1" = VL, C_L = 20 pF)

Parameter	Symbol	Min	Typ	Max	Unit
MCLK Specifications					
MCLK Period	t_{clkw}	36	-	45	ns
		72	-	1953	ns
MCLK Pulse Width High	tclkh	15	-	-	ns
MCLK Pulse Width Low	tclkl	15	-	-	ns

Abbildung 29: Spezifikationen zur Taktversorgung des verwendeten $\Sigma\Delta$ A/D Konverters

Überdies soll auch das Jitterverhalten des Taktes minimalst gehalten werden, um den Performanceerwartungen des A/D Wandlers (Klirrfaktor, Frequenzverlauf, SNR, etc.) zu entsprechen. Näheres dazu im technischen Datenblatt des Bausteins.

- Synchronität der Basissysteme

  Die Eigenschaft der Synchronität aller Basissysteme garantiert die samplegenaue Abtastung jedes einzelnen Mikrophonsignals. Bei Erfüllung dieser Voraussetzung leitet sich daraus die Möglichkeit ab, auch auf kompliziertere Signalverarbeitungsverfahren wie Korrelationssberechnungen zurückgreifen zu können. Das bedeutet aber für die schaltungstechnische Realisierung, daß jegliches asynchrones Verhalten im Pfad der Datenübertragung hervorgerufen durch autonome Oszillatoren o.ä. zu vermeiden ist.

- autonome, flexible Module

  Unter der Annahme, daß es am Markt kein System gibt, welches auf alle Festlegungen dieses Abschnitts 3.1 zielt, wird eine weitgehende Modularisierung des Systemaufbaus angestrebt. Der Vorteil dieser Vorgabe zeigt sich vor allem bei einer eventuellen Nichtverfügbarkeit von verwendeten Bauelementen, Änderungen bei der Verwaltung oder Sensorik des AKUT Gesamtsystems, aber auch bei Vereinfachungen oder Verbesserungen die durch den technologischen Fortschritt entstehen können.

- Kosten

  Nicht zu unterschätzen ist der entstehende finanzielle Aufwand. Da für den Betrieb eines AKUT Systems Dutzende von Basissystemen von Nöten sind, kann durch das Verwenden günstiger Bausteine die multiplikative Kostenabhängigkeit zu einem Vorteil umgekehrt werden.

- Portierbarkeit der Software

  Zu guter Letzt soll garantiert sein, daß die Anwendung des AKUT Systems keine Frage des verwendeten Betriebssystems ist und somit Verwaltung wie auch Steuerung der Mikrophonkanäle an die vorherrschende Rechnerarchitektur und Infrastruktur in der Tunnelzentrale adaptiert werden kann.

3.2 Grundsätzliche Idee der Realisierung

Eine der beiden Zielsetzungen dieser Erweiterung (siehe Abschnitt 1.2) betraf die Steuerung der Verstärkung der Mikrophonsignale aus den Räumlichkeiten der Tunnelzentrale heraus. Hierzu stehen vorerst noch 2 Varianten zur Auswahl.
Variante 1 (Abb. 30) zeigt den Ansatz über die Manipulation des Mikrophonvorverstärkers. Zu diesem Zweck soll das zur Steuerung vorhandene mechanische Potentiometer durch ein Digitales mit SPI-Ansteuerung (vgl. Abschnitt 2.4) ersetzt werden.

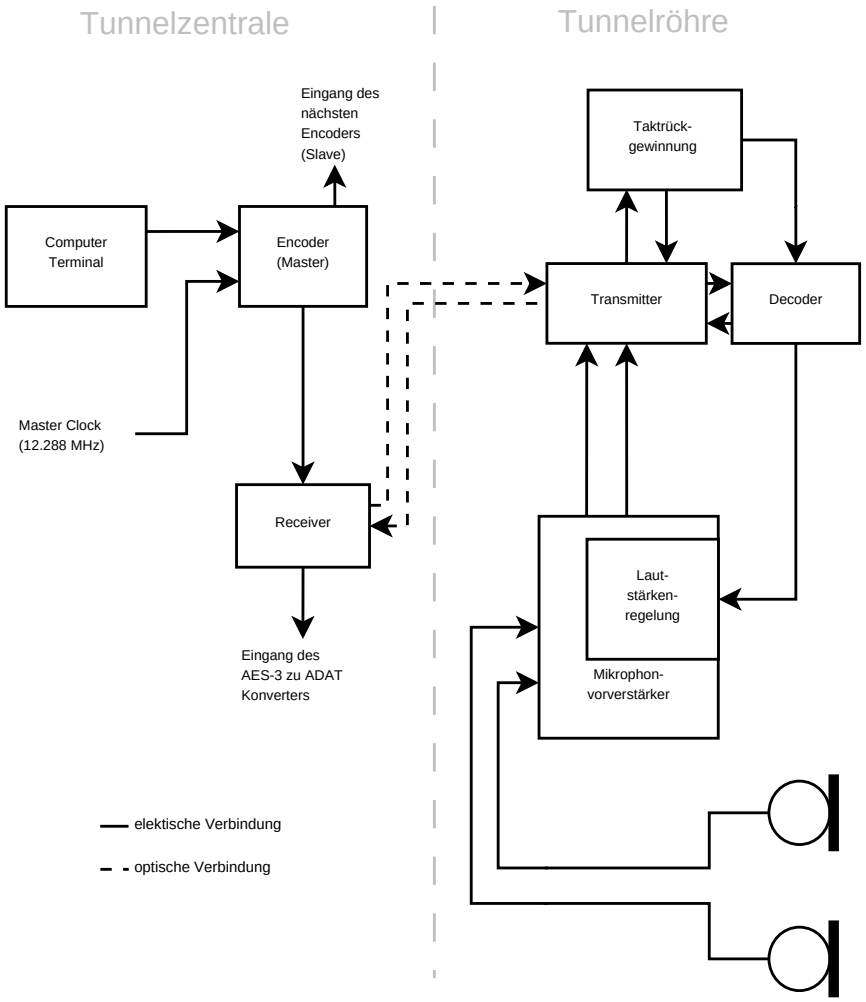


Abbildung 30: Erweitertes Basissystem in Variante 1

Die zweite Variante (Abb. 31) läßt den Mikrophonvorverstärker unangetastet, welcher nun mit einem fixen Verstärkungsfaktor betrieben wird. Dafür ist aber unmittelbar an dessen Ausgang eine autonome Lautstärkenregelung die ebenso, wie das vorher erwähnte Digitalpotentiometer, über SPI programmiert wird.
Die entgültige Auswahl und deren Entscheidungskriterien sind in Kapitel 3.3.3 dargelegt.
Alle weiteren Ausführungen über das erweiterte Basissystem betreffen trotz des geschilderten Unterschiedes zwischen Variante 1 und 2 beide abgebildeten Realisierungsoptionen.

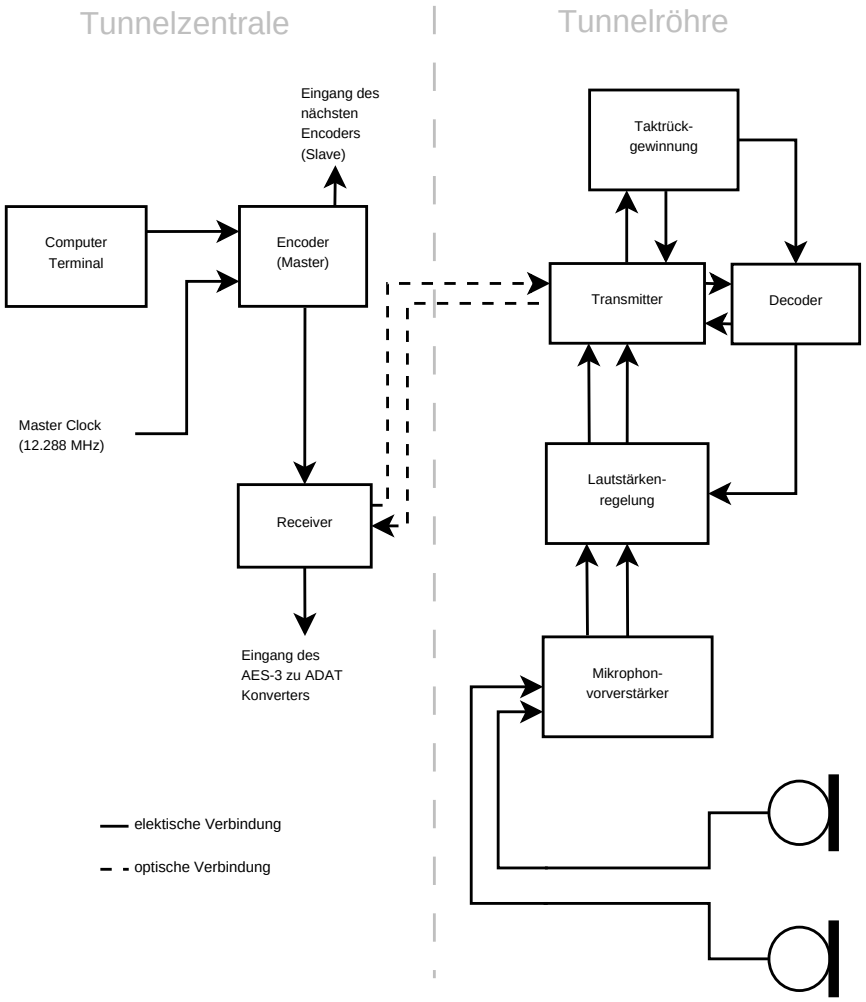


Abbildung 31: Erweitertes Basissystem in Variante 2

Um die Steuerdaten für die Lautstärkenregelung von der Tunnelzentrale über den bestehenden Lichtwellenleiter zum Ort des Geschehens zu übermitteln, werden zuerst die gewünschten Parameter in der Verwaltungssoftware (Abschnitt 3.4, Anhang B) des Computer Terminals gewählt und dem Master-Encoder zugeführt.
Neben besagtem Master-Encoder gibt es für jedes weitere erweiterte Basissystem einen Slave-Encoder, die allesamt keinen Anschluß zum Terminal besitzen. Verbunden sind die Encoder untereinander mittels Linientopologie (Abb. 32) , um den Verkabelungsaufwand im Schaltschrank der Tunnelzentrale möglichst überschaubar zu halten - bei Dutzenden bis Hunderten Basissystemen ein nicht zu unterschätzender Faktor.
Dies impliziert allerdings auch, daß an allen beteiligten Encodern diesselben Steuerdaten anliegen, und diese in der Lage sind, die Daten auf ihre Zugehörigkeit zu den angeschlossenen Mikrophonkanälen zu überprüfen. Im Falle einer gewünschten Veränderung des aktuellen Verstärkungsfaktors eines Mikrophonkanals werden im zugehörigen Encoder auf das zu jeder Zeit übertragene Master Clock Signal mittels Manchester Coding (Abschnitt 2.1.4) die erforderlichen Steuerdaten aufmoduliert und dem Receiver zugeschanzt, welcher wiederum mit dem Lichtwellenleiter verbunden ist.

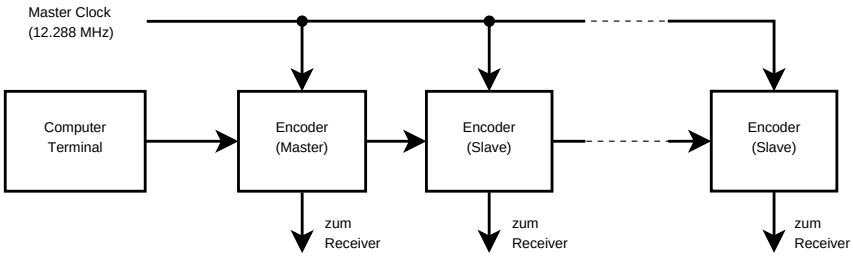


Abbildung 32: Verbindungstopologie der Encoderbausteine

Im bestehenden Transmitter am anderen Ende des Lichtwellenleiters erfolgt die Auskopplung des frühestvorliegenden elektrischen Signals zur Taktrückgewinnung.
Das vollständig rekonstruierte Taktsignal dient einerseits dem Betrieb des $\Sigma\Delta$ A/D Wandlers und des AES-3 Encoders im Transmitter, andererseits auch der Dekodierung der, noch im Manchester Code gepackten, Steuersignale die vom Transmitterbaustein in den Decoder weitergeleitet wurden.
Zusätzlich befindet sich am Decoder noch eine Schaltung zur Aufbereitung der Steuerdaten, um sie konform den Vorgaben des Serial Peripheral Interfaces der Lautstärkenregelung übermitteln zu können.
Die zweite Aufgabe bestand in der Realisierung der Resetfunktion aus der Tunnelzentrale heraus.
Die Unterschiede zu Ersterer sind marginal. Bis zum Decoder vollziehen die Daten denselben Weg wie die Steuerdaten. Von dort aus löst die entsprechende Bitkombination elektronisch den Reset des $\Sigma\Delta$ A/D Wandlers und des AES-3 Encoders am Transmitter aus. Zusätzlich wurde an der Möglichkeit des manuellen Rücksetzens vor Ort nichts verändert und ist weiterhin möglich.

Dies beschreibt die Funktionsweise der Schaltung im Groben und Ganzen. Zur näheren Erläuterung der einzelnen Bausteine verweise ich auf die Kapitel 3.4 und 3.5.

3.3 Recherche über bereits existierende Bausteine zur Lösung von Teilen der Problemstellung

In diesem Abschnitt werden die Ergebnisse der Markterhebung über bereits erhältliche Komponenten zur Realisierung der Erweiterung der AKUT Basissysteme dargelegt.

3.3.1 Schnittstelle Computer Terminal - Encoder (Master)

Im Kapitel 3.2 wurde die Notwendigkeit der Anbindung des Master-Encoders an das Computer Terminal erwähnt. Noch nicht festgelegt ist jedoch die Entscheidung über den zu verwendenden Schnittstellenstandard, welche nachfolgend geschildert wird.

- IO-Warrior

  Die Firma Code Mercenaries Hard- und Software GmbH stellt mit der IO-Warrior Serie eine unkomplizierte und universelle Lösungsmöglichkeit für die Realisierung dieser Schnittstelle zur Verfügung.

Diese integrierten Schaltkreise glänzen mit folgender Ausstattung:

- zukunftssichere USB Anbindung zum Computer
- abhängig vom ausgewählten Produkt 12 bis 50 verfügbare I/O Pins
- Bibliotheken zur Einbindung in eigene Software (unterstützt Windows, Linux und Mac-OS)
- Spezialfunktionen wie z.B. LCD Display, SPI, I²C, RC5 Infrared, LED Matrix, Key/Switch Matrix Interfaces
- Vielfältige Auswahl an IC Packages und Starter Kits (siehe Tabelle 5)

Die Anzahl der auf dem IO-Warrior vorhandenen I/O Pins und die genaue Zuordnung der Spezialfunktionen sind in Tabelle 4 illustriert.

Typ	I/O Pins	LCD	SPI	I²C	RC5 IR	LED	KEY/ SWITCH
IO-Warrior 40	32	✓	✗	✓	✗	✓	✓
IO-Warrior 24	16	✓	✓	✓	✓	✓	✗
IO-Warrior 24 PV	12	✗	✓	✓	✗	✗	✗
IO-Warrior 56	50	✓	✓	✓	✓	✓	✓

Tabelle 4: Funktionsumfang der IO-Warrior Serie

Die Preise bewegen sich pro Einheit in der Größenordnung von 11,75 € (IO-Warrior 24) bis 34.90 € (IO-Warrior 56), die der Starter Kits von 49 € (IOW40-Kit) bis 69 € (IOW56-Kit).[1] Weitere Informationen über das Angebot der Code Mercenaries Hard- und Software GmbH sind auf der firmeneigenen Homepage http://www.codemercs.com abrufbar.
Aufgrund der angebotenen Vielfalt wurde nach Kenntniserlangung dieser Chipserie die Recherche über die weiteren Möglichkeiten der Schnittstellenanbindung beendet. Für die Realisierung der Schnittstelle zwischen Computer Terminal und Encoder (Master) genügt die Version mit 32 I/O Pins (näheres in den Abschnitten 3.4 und 3.5).

[1] Preisniveau 20. März 2010

Typ	SSOP48	DIL24	SOIC24	MLFP56	MODUL	STARTER KIT
IO-Warrior 40	✓	✗	✗	✗	✓	✓
IO-Warrior 24	✗	✓	✓	✗	✗	✓
IO-Warrior 24 PV	✗	✓	✓	✗	✗	✗
IO-Warrior 56	✗	✗	✗	✓	✓	✓

Tabelle 5: Angebotene Packages der IO-Warrior Serie

3.3.2 Manchester Co- und Decodierung

Da der Manchester Code ein sehr verbreiteter Leitungscode ist, und es deshalb naheliegt, daß bereits vorgefertigte Bausteine zur Co- und Decodierung existieren, wurde auch in diesem Fall eine Feststellung über die angebotenen Bausteine durchgeführt.

- SEEQ 8020 MCC

  Der erste Kandidat zur Erledigung der Leitungscodierung und Decodierung ist dieser eben genannte integrierte Schaltkreis der Firma LSI Logic.

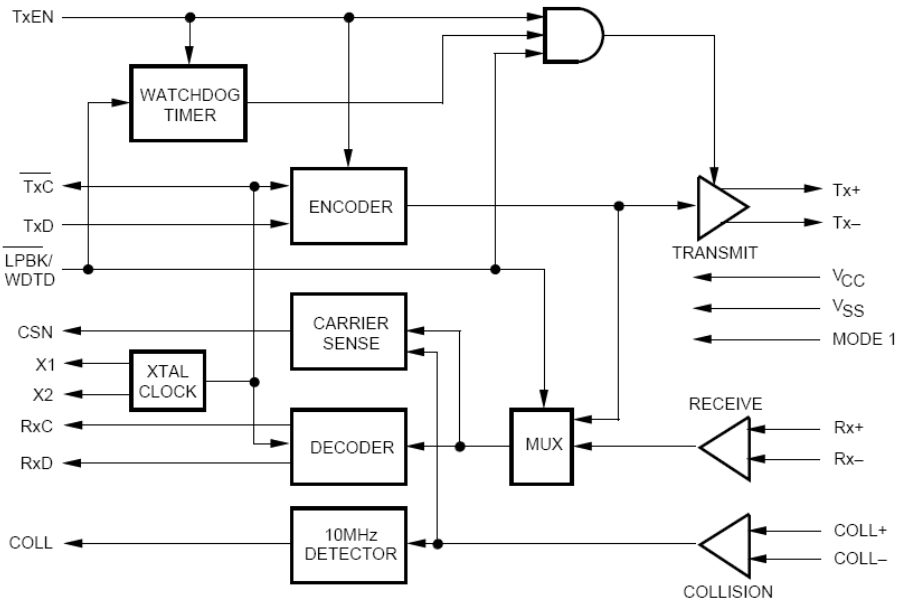


Abbildung 33: Blockdiagramm des SEEQ 8020 MCC

Features: Modulation des Master Clock Signals mit dem ankommenden NRZ Daten $(TxC,\ TxD \Rightarrow Tx+,\ Tx-)$

Dekodierung der empfangenen, manchestercodierter Information $(Rx+,\ Rx- \Rightarrow RxD)$

Taktrückgewinnung mittels integrierter PLL $(Rx+,\ Rx- \Rightarrow RxC)$

integrierte Treiber zur differentiellen Datenübertragung über den angeschlossenen Kommunikationskanal $(Rx+,\ Rx-,\ Tx+,\ Tx-)$

automatische Kollisionsabfrage $(Coll+,\ Coll- \Rightarrow Coll)$

deaktivierbarer Watchdog zur Verhinderung kontinuierlicher Datenübertragung

Auf den ersten Blick erfüllt der Baustein von LSI Logic die Vorgaben bzw. übertrifft diese sogar.
Bei genauerem Betrachten gibt es allerdings eine Eigenschaft des Mikrochips, welche seinen Einsatz für die Erweiterung der AKUT Basissysteme nicht zuläßt. Das Taktsignal TxC ist lt. Datenblatt nur für eine Frequenz von 20 MHz spezifiziert. Zurückzuführen ist dieser Umstand auf die hauptsächliche Verwendungsart des Produktes zur Ethernet-basierenden Informationsübertragung. Da aber der $\Sigma\Delta$ A/D Wandler ein integrales Vielfaches von 48 kHz zur Verarbeitung der Audiosignale benötigt stellt dieser IC keine kompatible Lösung dar.

- HD-6409

  Als zweiten, in Frage kommenden, Schaltkreis wurde der HD-6409 der Intersil Corporation unter die Lupe genommen.

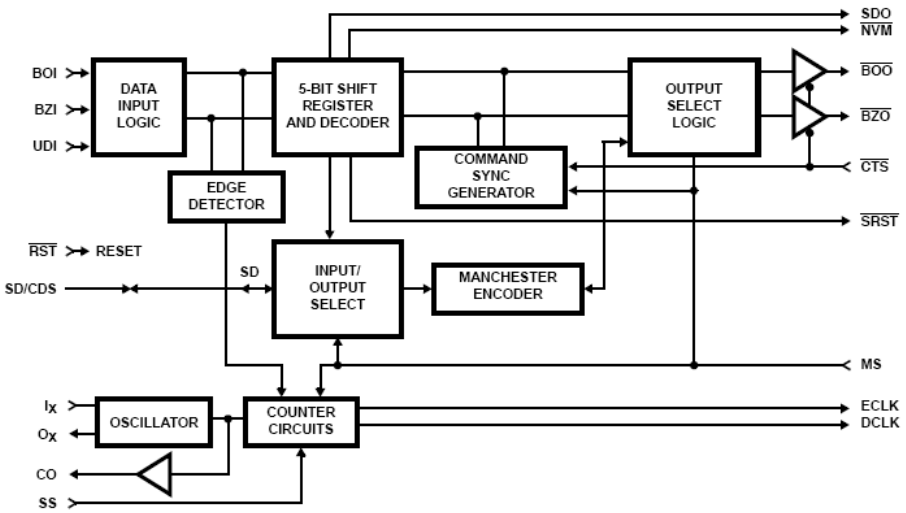


Abbildung 34: Blockdiagramm des HD-6409

Features: Repeater Modus: Codierte Daten werden aufgefrischt/wiederhergestellt $(BOI,\ BZI \Rightarrow \overline{BOO},\ \overline{BZO})$

Converter Modus: Codierung der NRZ Daten $(UDI \Rightarrow \overline{BOO},\ \overline{BZO})$

decodieren des Manchester Codes $(BOI,\ BZI/UDI \Rightarrow SDO)$

Taktrückgewinnung mittels digitaler PLL $(BOI,\ BZI/UDI \Rightarrow DCLK)$

Auch diese Schaltung könnte theoretisch zum Management des manchestercodierten Signals herangezogen werden. Die maximal erlaubte Frequenz des Taktes beträgt 16 MHz - ohne Einschränkungen ganz im Gegensatz zum SEEQ 8020 MCC. Jedoch bleibt die Problematik des Jitterverhaltens der verbauten digitalen PLL ein Stolperstein.

Kurz zusammengefaßt kann man feststellen, daß das eigentliche Problem dieser markttechnischen Erhebung dieses ist, daß die Prioritäten der Hersteller auf 'wirkliche', durchsatzstarke Datenübertragung gelegt wurde und nicht wie in der vorliegenden Aufgabenstellung die quasiperfekte, dauerhafte Verteilung des Master Clocks.
Dies stellt auch das ausschlaggebende Moment dar, die Codierung, Decodierung und Taktrückgewinnung in die eigene Hand zu nehmen und einen diskreten Aufbau zu wagen.

3.3.3 Möglichkeiten zur Lautstärkenregelung

Wie aus den Abbildungen 30 und 31 ersichtlich ist, unterscheiden sich die zwei durchgedachten Varianten zur Erweiterung des Basissystems nur in der Implementierung der Lautstärkenregelung.
Die minimale Anforderung dafür entspricht der Abdeckung der 60 dB Tag/Nacht Differenz im Schalldruckpegel in maximal 0,5 dB Schritten ($\Rightarrow$ mindestens 120 Einstellungen).

- Austauschen des vorhandenen Potentiometers

  Als ersten Ansatz wurde die Idee verfolgt das mechanische Potentiometer des Mikrophonvorverstärkers durch ein digitales Potentiometers für Audioanwendungen zu ersetzen. Dies bringt jedoch folgende Nachteile mit sich:

  - zusätzlicher digitaler Schaltkreis auf der Platine des analogen Vorverstärkers
  - überproportional große Toleranzen der Widerstandskennlinie verglichen mit linearen Potentiometern
  - unbekanntes Verhalten beim Umschaltzeitpunkt $\Rightarrow$ eventuelle Glitches im Audiosignal

  Hersteller dieser Bauteile sind z.B. ON Semiconductors, Maxim IC oder Analog Devices, aber leider waren zur Zeit der Projektplanung keine '128 tap Audio Digipots' verfügbar weswegen diese Realisierungsmöglichkeit fallen gelassen wurde.

- Einsetzen einer separaten Lautstärkenregelung

  Hier scheint das Marktangebot keine Wünsche offen zu lassen. Sowohl der PGA-2311 von Burr Brown/Texas Instruments als auch der CS3310 von Cirrus Logic schaffen es mit Leichtigkeit die geforderten Voraussetzungen zu erfüllen sowie auch die Nachteile der digitalen Potentiometer zu umgehen.
  Verwunderlich ist allerdings die augenscheinliche Ähnlichkeit bezüglich der Kenndaten der

beiden Mikrochips, sodaß es genügt im Weiteren nur den PGA-2311 detaillierter vorzustellen, der schlußendlich auch den Vorzug gegenüber dem CS3310 bekam aufgrund marginal besserer Daten, aber vor allem durch die leichtere Verfügbarkeit.

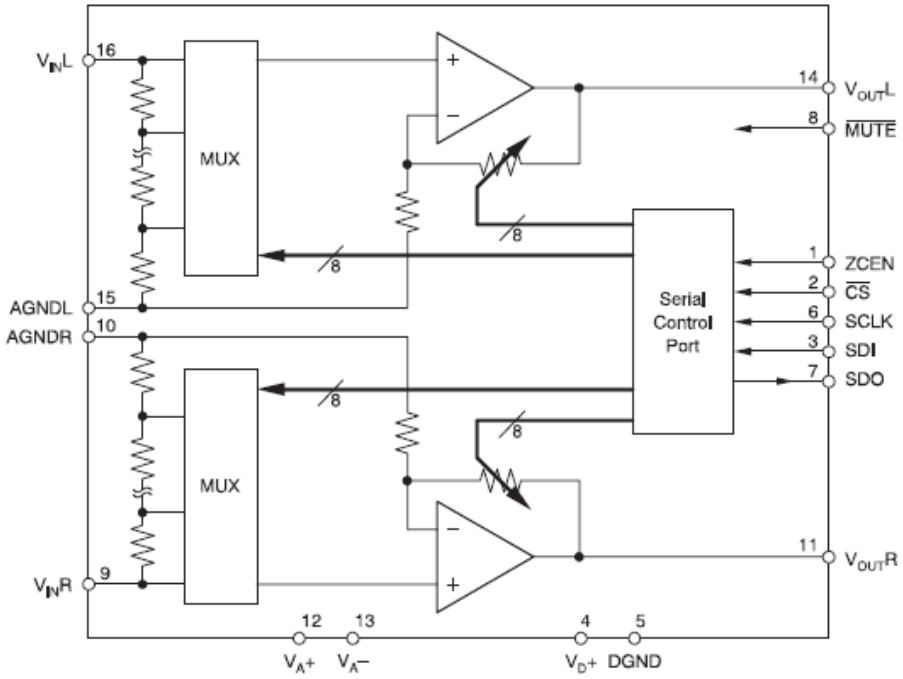


Abbildung 35: Blockschaltbild des PGA-2311

Features: Verstärkung im Bereich von +31,5 dB bis -95,5 dB in 0,5 dB Schritten

unterstützt zwei unabhängige analoge Audiokanäle $(V_{in}L/AGNDL, V_{in}R/AGNDR)$

Ansteuerung via SPI $(\overline{CS},\ SCLK,\ SDI,\ SDO)$

zuschaltbare Nulldurchgangsdetektion des Audiosignals zur Verhinderung von Signalartefakten während des Umschaltzeitpunkts $(ZCEN)$

120 dB Dynamikumfang

Kanalübersprechen < -130 dB

Hinzu kommt noch die Möglichkeit digitale und analoge Domäne des Bausteins vollständig zu separieren (vgl. Abbildung 36). Diesen störtechnischen Vorteil erkauft man sich allerdings mit der Notwendigkeit einer zusätzlichen bipolaren Spannungsversorgung für den Analogteil.
Weiters stehen von der gleichen Firma die Bausteine PGA-2310 und PGA-2320 zur Verfügung, die mit ±15 V Betriebsspannung versorgt werden, aber ansonsten vollkommen pin- und funktionskompatibel mit dem PGA-2311 sind.

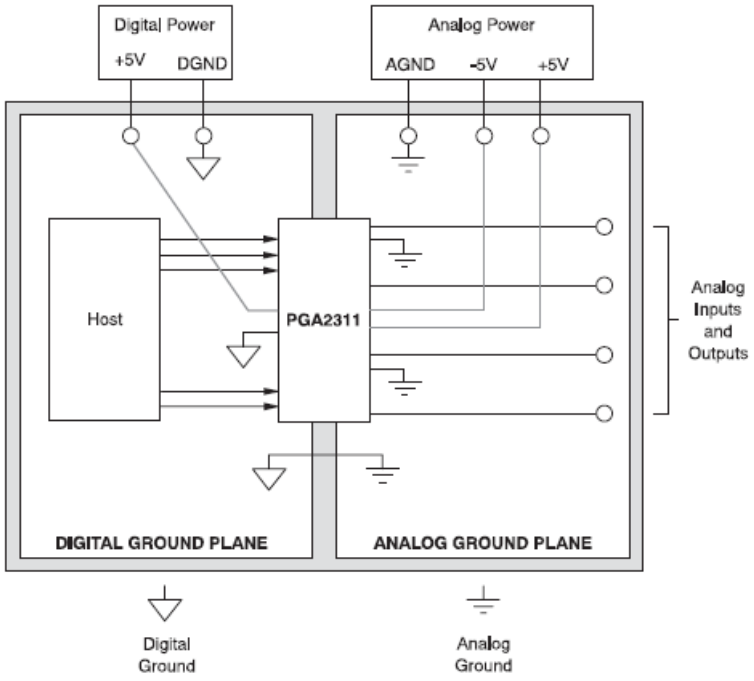


Abbildung 36: Layoutempfehlung für den PGA-2311

3.4 Resultierende Software

Die Applikation zur Steuerung der Hardware in Abschnitt 3.5, genauer gesagt des Master-Encoders, wurde in C++ als plattformunabhängige Konsolenapplikation verwirklicht, deren Funktionsweise in den nächsten Kapiteln beleuchtet werden soll. Zum Einsatz kamen ausschließlich Elemente aus der C++ Standard Library, um die Portierbarkeit so einfach und unkompliziert wie möglich zu gestalten. Zuvor erfolgt aber noch eine Übersicht wie die benötigten Daten zur Steuerung der Mikrophonsignalverstärkung, der Taktrückgewinnung und des Transmitters strukturiert sind.

3.4.1 Struktur der Steuerdaten

Wie in Abschnitt 3.3.1 schon erwähnt, fiel die Auswahl des ensprechenden IO-Warriors auf die Version mit 32 I/O-Pins, um am Master-Encoder die 32-Bit Steuerdaten parallel verarbeiten zu können.

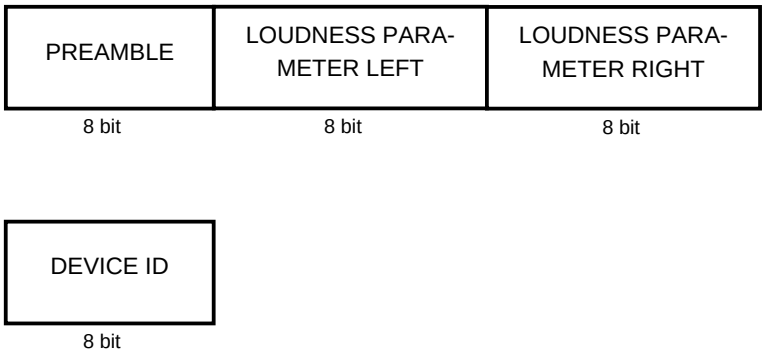


Abbildung 37: Struktur des Steuerdaten

3.4.1.1 Preamble

Diese Bitsequenz bestimmt die Art der auszuführenden Operation.
Folgende drei gültige Varianten sind implementiert, können aber unter Berücksichtigung der Arbeitsweise des Präambel Decoders (Kapitel 3.5.3.3) in der Datei 'defines.h' (siehe Anhang B) individuell angepaßt werden:

- Control Preamble:

  Wird gewählt, um eine Manipulation des Verstärkungsfaktors der Lautstärkenregelung zu beabsichtigen. Diese ist ohne Einschränkungen wählbar, da die Schieberegister im Decoder (vgl. Abschnitt 3.5.3) nach einem Control Befehl ohnehin gelöscht werden.

  Standard Controlpräambel = '11111111'

- Sync Preamble:

  Synchronisiert die Taktrückgewinnung (siehe Kapitel 3.5.2.2), um sicherzustellen, daß auf die Bitzellenmitte der manchestercodierten Daten getriggert wird. Es gilt bei der Auswahl der Syncpräambel zu berücksichtigen, daß sich durch folgende Taktzyklen in den Schieberegistern des Decoders nicht irrtümlich die Signatur einer Controlpräambel ergibt und somit die Lautstärkenregelung der angeschlossenen Kanäle ungewollt dejustiert. Die Illustration dieses Problems erfolgt in Abbildung 38.

  Standard Syncpräambel = '00011000'

- Reset Preamble:

  Kündigt einen Reset des $\Sigma\Delta$ A/D Wandlers und des AES-3 Encoders am Transmitter an. Bei Wahl der Resetpräambel sind die gleichen Überlegungen zu tätigen, die auch für die Syncpräambel gelten.

  Standard Resetpräambel = '11110111'

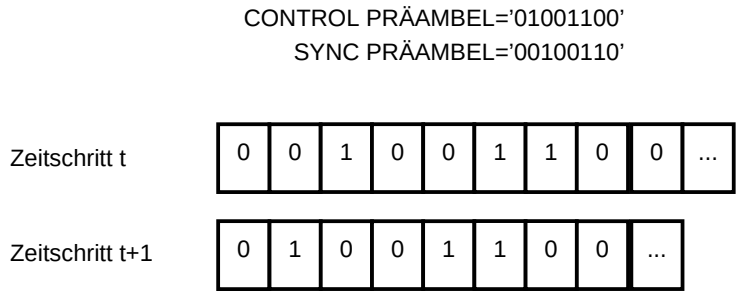


Abbildung 38: Beispiel für eine ungültige Wahl der Präambelsequenzen

3.4.1.2 Loudness Parameter

Dieser Parameter entspricht dem binären Verstärkungswert für die Lautstärkenregelung PGA-2311 des jeweiligen Mikrophonsignals. Der gültige Bereich erstreckt sich von '00000000' $\rightarrow$ -95,5 dB bis '11111111' $\rightarrow$ +31,5 dB (genauere Angaben dazu in Kapitel 3.5.4)

3.4.1.3 Device ID

Jeder Encoder besitzt einen 8-Bit DIL Schalter über den seine Device ID eingestellt werden kann. Die Device ID Logik im Encoder (Abschnitt 3.5.1) vergleicht diese mit den 8-Bit Device ID Steuerdaten und lädt bei Übereinstimmung die restlichen Steuerdaten (Preamble, Loudness Parameter Left und Loudness Parameter Right) in die Schieberegister des Encoders.
Weiters ist zu beachten, daß in der Datei 'defines.h' (siehe Anhang B) eine beliebig wählbare Device ID für den Shifting-Mode der Schieberegister zu reservieren ist (→ Device ID for shifting op) und diese dadurch nicht mehr zur Device ID Kennung per DIL Schalter zur Verfügung steht.

3.4.2 Struktur der Datenspeicherung

Alle notwendigen Daten zur Verwaltung eines AKUT Systems werden in einem 'vector<vector<string>>' Array der selbstdefinierten Klasse 'lut' gespeichert. Die maximale Länge jedes Strings beträgt 10 Zeichen. Nähere Informationen über diese Klasse sind im Anhang B angeführt. Der Aufbau des soeben beschriebenen Arrays ist in Tabelle 6 veranschaulicht:

Channel	Device ID	Byte Alignment	Loudness Parameter
'0'	'00000001'	'left'	'22'
'1'	'01000100'	'right'	'255'
'2'	'01001100'	'right'	'0'
...	...	...	...

Tabelle 6: Illustration eines gültigen Lookup Table Arrays

3.4.2.1 Channel

Der Parameter Channel entspricht der zugeordneten Nummer eines einzelnen Mikrophonkanals. Diese muß - als einzige formale Gesetzmäßigkeit des Arrays - von 0 bis zur Maximalanzahl der Audiokanäle-1 ansteigend definiert sein.

3.4.2.2 Device ID

Begriffserklärung siehe Abschnitt 3.4.1. Diese Bitfolge dient der Information welche Device ID und damit auch welcher Encoder zum jeweiligen Mikrophonkanal gehört.

3.4.2.3 Byte Alignment

Da jedes AKUT Basissystem in der Lage ist zwei Mikrophonkanäle anzusprechen, muß für die eindeutige Zuordnung neben der Device ID noch eine weitere Information, das Byte Alignment, dargelegt werden. Gültige Werte für diesen Eintrag sind 'left' bzw. 'right'.

3.4.2.4 Loudness Parameter

Siehe Abschnitt 3.4.1. Allerdings ist die Darstellung und Speicherung dieser Größe im Array aus Gründen des Bedienkomforts dezimal dargestellt. Die gewünschten zulässigen Minimal- und Maximalwerte sind in der Datei 'defines.h' (siehe Anhang B) einstellbar.

3.4.3 Befehlsübersicht

Die Steuerung der Software erfolgt über den Aufruf der implementierten Befehle und den zugehörigen Parametern, die nachfolgend erläutert werden.

- **INIT**

  Bereitet alle AKUT Basissysteme auf den Betrieb vor, indem der am Terminal angeschlossene IO-Warrior initialisiert, die Taktrückgewinnungsblöcke aller Kanäle synchronisiert, ein Standard Array zur Verwaltung generiert, und jeder Mikrophonkanal auf vordefinierte Werte gestellt wird.

- **LOAD** [filename]

  Lädt ein gespeichertes Verwaltungsarray aus der CSV-Datei [filename] und testet es auf seine Gültigkeit. Ist diese gegeben, so werden die entsprechenden Loudness Parameter auf die AKUT Basissysteme übertragen - ist es aber ungültig, wird das geladene Array verworfen, und der Zustand, der vor dem LOAD Befehl vorherrschte, bleibt unverändert.

- **SAVE** [filename]

  Unter der Bedingung, daß [filename] noch nicht existiert, wird das aktuelle Array in der CSV-Datei [filename] gespeichert. Ansonsten erfolgt ein Dialog das Überschreiben der existierenden Datei [filename] betreffend.

- **ABS** [channel_begin-channel_end] [loudness_parameter]

  Setzt die, durch 'channel_begin' und 'channel_end' definierten, Mikrophonkanäle auf den angegebenen Verstärkungsfaktor [loudness_parameter] und vermerkt diese Veränderung im Verwaltungsarray, wenn die Operation erfolgreich ausgeführt wurde.

- **INC** [channel_begin-channel_end]

  Erhöht den Verstärkungsfaktor jedes Mikrophonkanals, des durch 'channel_begin' und 'channel_end' definierten Bereiches, um +0,5 dB und vermerkt diese Veränderung im Verwaltungsarray, wenn die Operation erfolgreich ausgeführt wurde.

- **DEC** [channel_begin-channel_end]

  Ändert den Verstärkungsfaktor jedes Mikrophonkanals, des durch 'channel_begin' und 'channel_end' definierten Bereiches, um -0,5 dB und vermerkt diese Veränderung im Verwaltungsarray, wenn die Operation erfolgreich ausgeführt wurde.

- **RESET** [channel_begin-channel_end]

  Führt einen Reset der A/D Wandler und der AES-3 Encoder der Audiokanäle 'channel_begin'-'channel_end' sowie die durch diesselbe Device ID ebenso betroffenen Mikrophonkanäle durch.

- **PRINT** [channel_begin-channel_end]

  Dient der Bildschirmausgabe eines gewünschten Ausschnitts (durch [channel_begin-channel_end] definiert) des Verwaltungsarrays.

- **QUIT**

  Speichert den aktuellen Zustand des Verwaltungsarrays in der CSV-Datei 'autosave.csv' und beendet die AKUT Control Software.

3.5 Resultierende Hardware

Dieses Kapitel bildet einen funktionalen Überblick über die schaltungstechnischen Erweiterungen des endgültig realisierten, erweiterten Basissystems aus Abbildung 30. Die genauen Baupläne der elektronischen Schaltungen sind den Schematics in Anhang A zu entnehmen.

3.5.1 Encoder

Das Blockschaltbild des entworfenen Encoders ist in Abbildung 39 dargestellt, dessen einzelne Komponenten in den Kapiteln 3.5.1.1 bis 3.5.1.4 näher erläutert werden.

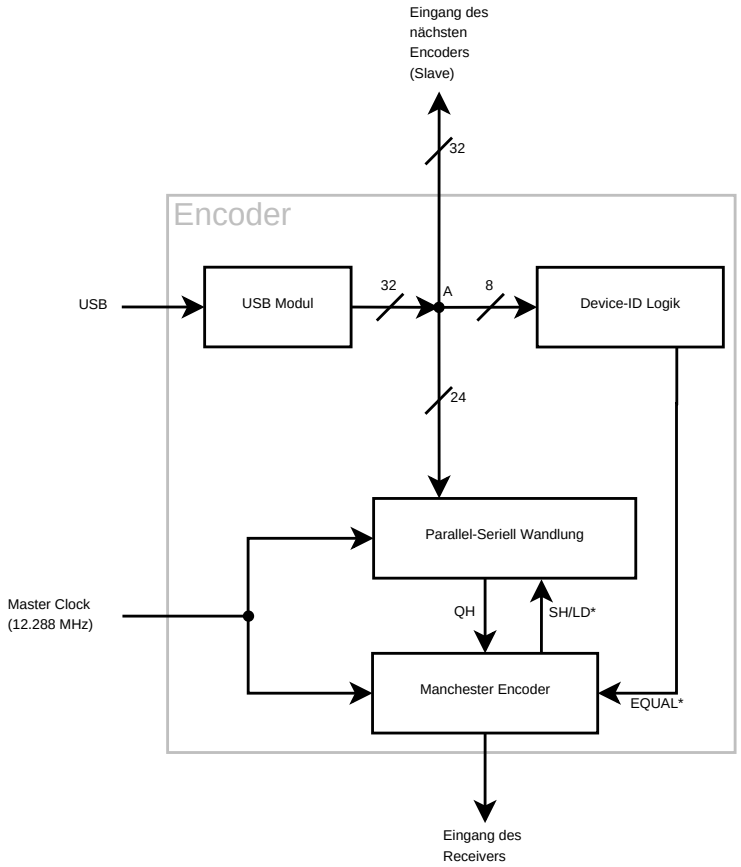


Abbildung 39: Blockschaltbild des Master-Encoders

3.5.1.1 USB Modul

Hauptteil dieses Blocks ist der in Abschnitt 3.3.1 beschriebene IO-Warrior Chip. Dieser Part ist ein Unikum des Master-Encoders selbst. Statt dieses Blocks erfolgt bei den Slave Varianten die Einspeisung der 32 Steuerbits an Knotenpunkt A wie in Abbildung 39 eingezeichnet.

3.5.1.2 Parallel-Seriell Wandlung

Da die Manchester Codierung für die synchrone serielle Datenübertragung ausgelegt ist, müssen die vom USB Modul gelieferten parallelen Steuerdaten (Preamble, Loudness Parameter Left, Loudness Parameter Right) zuerst in einen Bitstream umgewandelt werden.

Hierzu ist das Signal SH/LD^* (vgl. Abb. 39) des Manchester Encoders von Bedeutung:

- SH/LD^* ist logisch 0 $\rightarrow$ 24-Bit der parallelen Daten werden in die Schieberegister eingelesen
- SH/LD^* ist logisch 1 $\rightarrow$ Ausgabe des im Schieberegister enthaltenen Bitstreams (siehe Signal QH in Abb. 39) an den Manchester Encoder, taktsynchron mit der Frequenz des Master Clocks.

3.5.1.3 Device-ID Logik

Jeder Encoder besitzt, wie in Kapitel 3.4.1 beschrieben, eine über DIL-Switches einstellbare 8-Bit Kennung. Daraus generiert dieser Block ein entsprechendes Informationssignal $EQUAL^*$ für den nachfolgenden Manchester Encoder:

- Schalterstellung entspricht den 8-Bit Device ID Steuerdaten (vgl. Abb. 39) $\rightarrow$ $EQUAL^*$ ist logisch 0
- Schalterstellung entspricht nicht den 8-Bit Device ID Steuerdaten (vgl. Abb. 39) $\rightarrow$ $EQUAL^*$ ist logisch 1

3.5.1.4 Manchester Encoder

In diesem Baustein laufen nun alle wichtigen Daten zusammen. Der Ausgang des Schieberegisters der Parallel-Seriell Wandlung QH und das Taktsignal selbst werden intern einem EX-OR Gatter zugeführt, was den eigentlichen Codiervorgang darstellt.
Das $EQUAL^*$ Signal der Device-ID Logik wird in diesem Systemblock zum taktsynchronisierten SH/LD^* Indikator gewandelt, der wiederum in der Parallel-Seriell Wandlung die Austaktung der Steuerdaten aus dessen Schieberegistern veranlaßt:

- positive Flanke von $EQUAL^*$ $\rightarrow$ positive Flanke von SH/LD^* $\rightarrow$ Austakten der Steuerdaten über QH $\rightarrow$ Codieren der Steuerdaten im EX-OR Gatter des Manchester Encoders
- in allen anderen kombinatorischen Fällen von $EQUAL^*$ $\rightarrow$ QH ist logisch 0 $\rightarrow$ Taktsignal wird im EX-OR Gatter mit logisch 0 verknüpft, was wiederum den Systemtakt selbst ergibt.

Durch die synchrone Taktung der Parallel-Seriell Wandlung und des Manchester Encoders erfolgt die Umschaltung zwischen diesen beiden Funktionszuständen, ohne evt. Glitches, Jitter und sonstige Signalunreinheiten, d.h. ohne Störeinflüsse auf den, in der Signalkette ganz am Ende stehenden, $\Sigma\Delta$ A/D Wandler auszuüben.

3.5.2 Taktrückgewinnung

Aufgabe dieses Blocks ist die Rückgewinnung des Taktes des Master-Clock Generators inklusive seines Phasenrauschens (entspricht nach Eigendefinition einer absoluten Phasenstarrheit). Hauptaugenmerk soll auf die Erfüllung dieser Spezifikation gelegt werden, wenn keine Steuerdatenübertragung stattfindet, da es zu den Zeitpunkten von Resetbefehlen oder der Veränderung des Verstärkungsfaktors des Mikrophonkanals unweigerlich zur Störung der Arbeitsweise des betroffenen $\Sigma\Delta$ A/D Konverters kommt.

Folgende Annahmen sollen bei der theoretischen Herleitung der Taktrückgewinnung in diesem Kapitel Gültigkeit haben:

- alle verwendeten Flip Flops sind positiv flankengetriggert
- die Flip Flops triggern ausschließlich auf die Flanke in der Bitzellenmitte (vgl. Abbildungen 41 und 42). Dies wird durch die Synchronisation sichergestellt deren Problematik in Abschnitt 3.5.2.2 näher erläutert ist.
- keine Signallaufzeiten $\rightarrow$ instantane Umsetzung der Eingangszustände in die entsprechenden Ausgangszustände

Die an die Taktrückgewinnung aus manchestercodierten Daten gestellten Anforderungen konkretisieren sich zu:

1. Generation eines Pulses der Dauer T_P ausgelöst zu den Zeitpunkten $t = n\,T_{Bitzelle} - T_{Bitzelle/2}$; $(n \in \mathbb{N})$ durch Pegelwechsel im Signal $DATA\ IN$
2. Verhindern einer weiteren Pulstriggerung für die Zeitdauer T_S, die auch die jeweils nachfolgende Flanke zu den Zeitpunkten $t = n\,T_{Bitzelle}$; $(n \in \mathbb{N})$ in $DATA\ IN$ unterdrücken soll $\rightarrow T_S > T_{Bitzelle/2}$
3. Um die Taktinformation jeder Bitzelle des Manchester Codes auszunutzen, muß die Bedingung $T_\mathrm{S} < T_{Bitzelle}$ erfüllt sein
4. Die Anforderungen 2 und 3 lassen sich zur Bedingung $T_{Bitzelle/2} < T_S < T_{Bitzelle}$ subsumieren

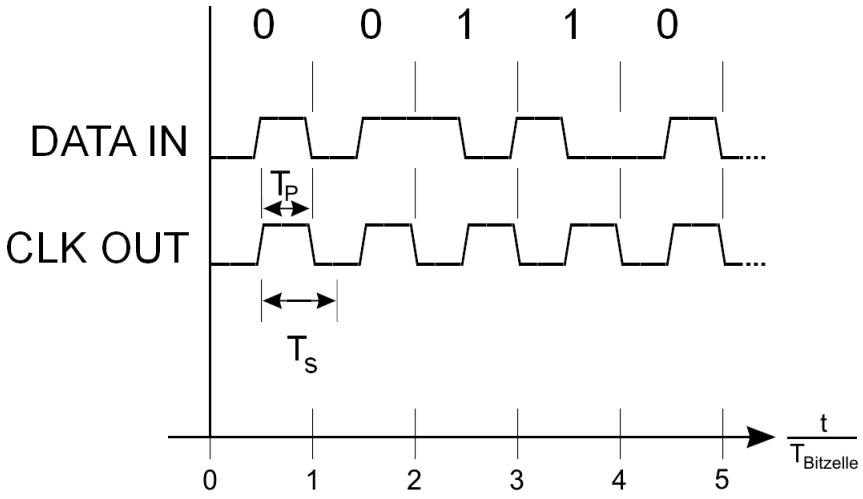


Abbildung 40: Illustration der Anforderungen an die Taktrückgewinnung

3.5.2.1 Funktionsweise

Das Prinzip der aus den Anforderungen an die Taktrückgewinnung abgeleiteten und realisierten Schaltung ist in Abbildung 41 gezeigt.

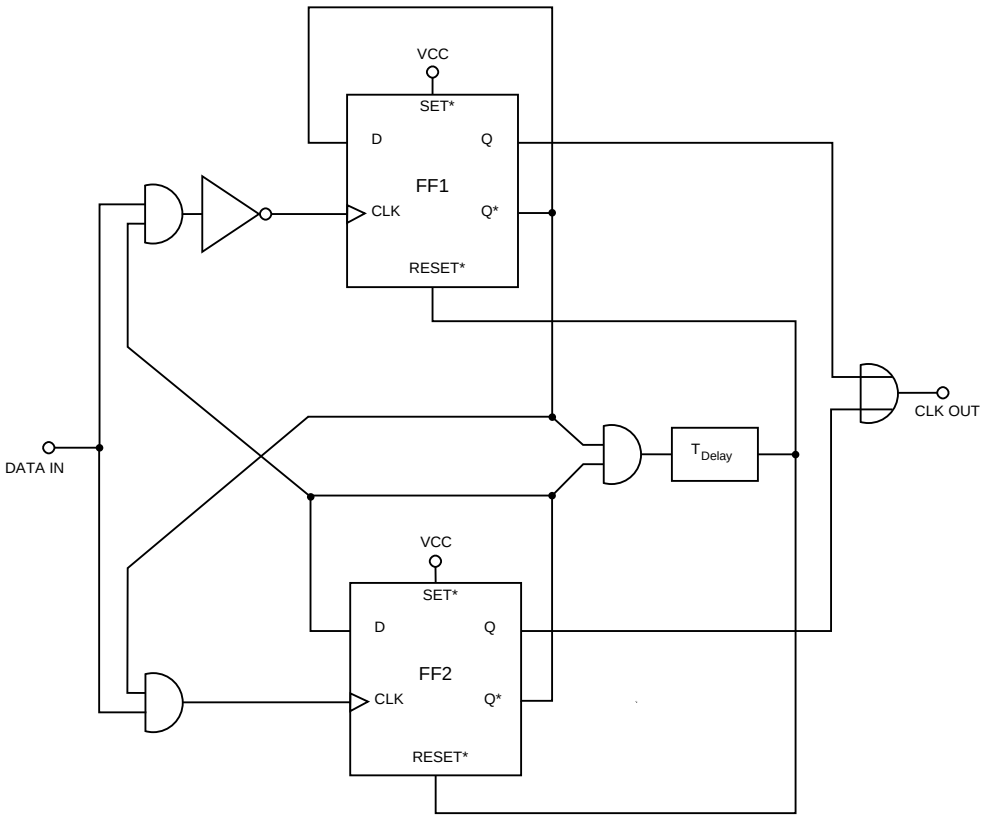


Abbildung 41: Prinzipbild der Schaltung zur Taktrückgewinnung

Die dieser Schaltung zugrundeliegende Funktions- und Arbeitsweise ergibt sich wie folgt:

- FF1 übernimmt das Generieren eines Pulses bei negativer Flanke von $DATA\,IN$ in der Bitzellenmitte, FF2 bei positivem Flankenevent
- Die Ausgänge Q^* der beiden Flip Flops verhindern sich gegenseitig mithilfe der UND Gatter an den CLK Eingängen weitere Pulstriggerungen während der aktiven Pulszeit
- Mit dem Parameter T_{Delay} kann das $RESET^*$ Signal und somit auch die Pulsbreite T_P festgelegt werden. Für Letztere folgt $T_P = T_{Delay}$ und schließlich als zu erfüllende Bedingung:

$$\begin{aligned} & T_{Bitzelle/2} < T_S < T_{Bitzelle} \\ \rightarrow \quad & T_{Bitzelle/2} < T_P + T_{Delay} < T_{Bitzelle} \\ \rightarrow \quad & T_{Bitzelle/2} < 2T_{Delay} < T_{Bitzelle} \\ \rightarrow \quad & T_{Bitzelle/4} < T_{Delay} < T_{Bitzelle/2} \end{aligned}$$

  Damit ist, wie schon bei den Anforderungen beschrieben, gewährleistet, daß es keine unerwünschten Fehltriggerungen am Bitzellenanfang gibt, aber auch, daß die Schaltung bereit ist die Orientierung der nächstfolgenden Flanke in der Bitzellenmitte auswerten zu können.

- Durch EX-OR Verknüpfung der beiden Q Ausgänge ist somit der wiederhergestellte Takt am Ausgang $CLK\,OUT$ der Schaltung abzugreifen.

Die Taktrückgewinnung verhält sich also wie ein logisches Gatter, falls keine Steuerdatenübertragung erfolgt ergo im normalen Betrieb des $\Sigma\Delta$ A/D Wandlers. Es treten dank dieses Gatterverhaltens keine zusätzlichen Jittereffekte oder zusätzliches Phasenrauschen im rekonstruierten Taktsignal auf - ganz im Gegensatz zu einer eventuellen Lösung der Aufgabe mittels PLL.
Nachteil dieser Schaltung ist, daß der Duty Cycle des Taktsignals keine 50% beträgt. Dies ist in den Datenblättern des verwendeten $\Sigma\Delta$ A/D Wandlers auch nicht gefordert, sondern nur minimale Pulsbreiten für beide Logikzustände von jeweils 15 ns (vgl. Abb. 29).
In der Praxis ergeben sich natürlich weitere Überlegungen durch die endlichen Signallaufzeiten der verwendeten Flip Flops und Gatter. Realisiert wurde aber schlußendlich dieses zeitkritischste aller Module mit der 74HCT Bauteilserie, die die Taktrekonstruktion einwandfrei erledigte. Es war nicht notwendig die 74F Serie (pinkompatibel zu 74HCT) heranzuziehen, die abermals nur Bruchteile der Signalverzögerungen der verwendeten Bauelemente besitzen und somit auch zur Taktrückgewinnung bis zu 100 MHz verwendbar wären.
Beim eigentlichen Aufbau der Hardware wurde darauf geachtet, daß in beiden Signalpfaden der Taktrückgewinnung diesselben zeitlichen Verzögerungen vorherrschen. Darum sind auch Logikbausteine von Texas Instruments (TI) gewählt worden, welche die Eigenschaft der 'Balanced Propagation Delays' aufweisen. Diese Bausteine sind in der Bauteilliste für das erweiterte Basissystem der Tunnelröhre in Anhang A eigens gekennzeichnet.

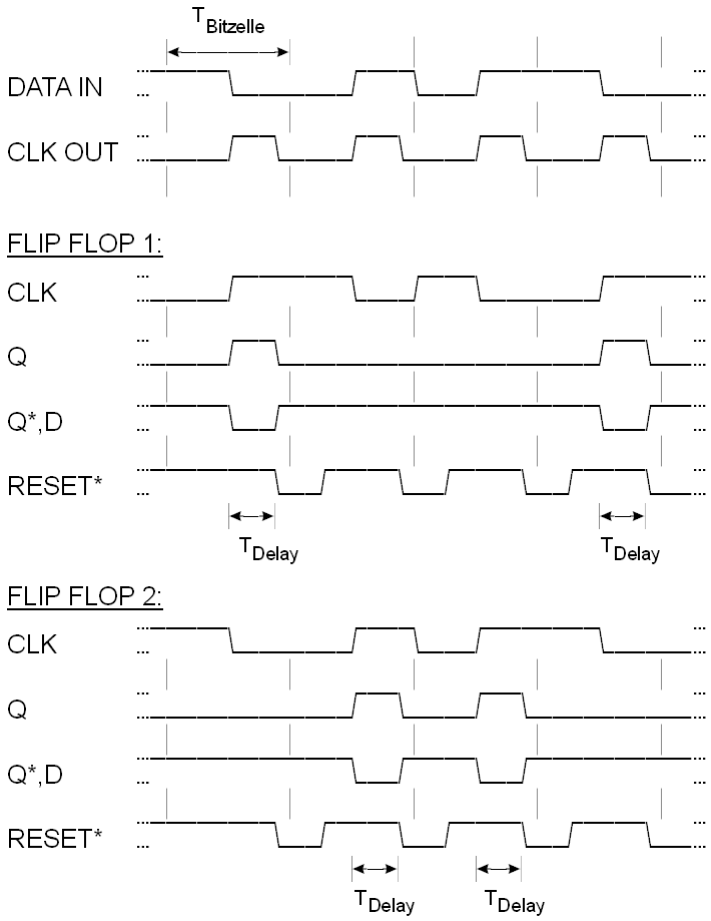


Abbildung 42: Timingdiagramm zur Schaltung in Abbildung 41

3.5.2.2 Synchronisation

Abbildung 43 soll das Auftreten einer Fehltriggerung der Taktrückgewinnung visualisieren. Charakteristisch für dieses Fehlverhalten ist das Auslösen der Pulse am Bitzellenanfang.

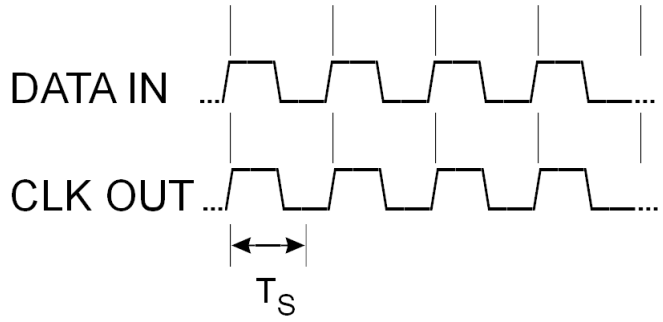


Abbildung 43: Fehltriggerung der Taktrückgewinnung

Das gleiche $CLK\ OUT$ Muster aus Abbildung 43 ließe sich auch mit dem um $T_{Bitzelle/2}$ verschobenen Signal $DATA\ IN$ erreichen mit dem Unterschied, daß permanent auf die negative Flanke am Bitzellenanfang getriggert würde d.h. das andere Flip Flop aus Abbildung 41 aktiv wäre.

Um diese Zustände auszuschließen, die nach dem Einschalten des Basissystems auftreten können, reicht es zur Synchronisierung der Taktrückgewinnung mindestens ein Steuerbit über den Lichtwellenleiter zu schicken, was durch die Übermittlung der Sync Preamble erreicht wird.

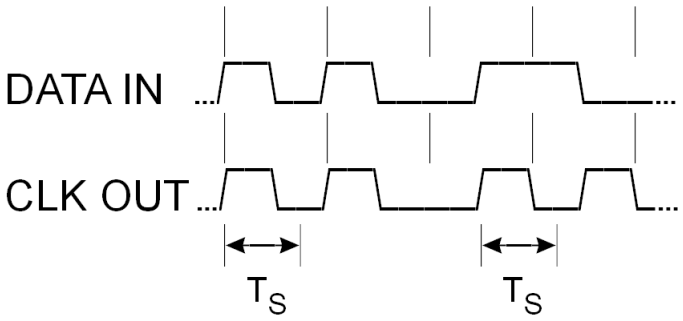


Abbildung 44: Vorgang der Synchronisation der Taktrückgewinnung

In Abbildung 44 ist der Verlauf der Synchronisation dargestellt. Anfänglich erfolgt die Pulstriggerung noch am Bitzellenanfang, jedoch synchronisiert sich die Schaltung durch eine fehlende Signalflanke am Beginn der Bitzelle (entspricht einem Pegelwechsel) von selbst und bleibt durch die Definition des Manchester Codes (vgl. Kapitel 2.1.4.6) synchronisiert.

3.5.3 Decoder

Das Blockschaltbild des entworfenen Decoders ist in Abbildung 45 dargestellt, dessen einzelne Komponenten in den Kapiteln 3.5.3.1 bis 3.5.3.5 näher erläutert werden.

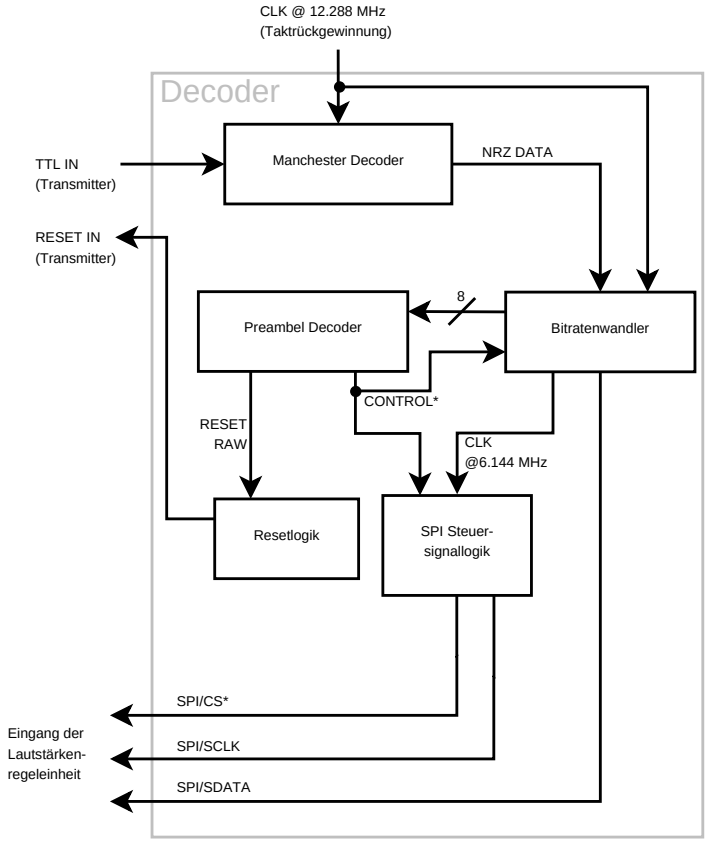


Abbildung 45: Blockschaltbild des Decoders

3.5.3.1 Manchester Decoder

Zuerst wird das eventuell verschliffene $TTL\,IN$ Signal des Transmitters digital aufgefrischt und dann der Leitungsdecodierung zugeführt, die in der selben Weise wie die Codierung - Eine EX-OR Verknüpfung mit dem wiederhergestellten und laufzeitkorrigierten Taktsignal $CLK@12.288\,MHz$ - abläuft. Anschließend erfolgt eine Abtastung der wiedergewonnenen NRZ Steuerdaten mit dem rückgewonnenen Systemtakt, um eventuelle Glitches im Informationsmuster $NRZ\,DATA$ zu eliminieren.

3.5.3.2 Bitratenwandler

Hier wird aus dem seriellen Bitstrom $NRZ\,DATA$ die 8-Bit Präambel (vgl. Kapitel 3.4.1) vom eigentlichen 16-Bit Lautstärkeparameter getrennt. Letzteren benötigt der PGA-2311 zur Justierung des Verstärkungsfaktors der angeschlossenen Mikrophonkanäle (siehe Abschnitt 3.5.4).
Bei entsprechender Signalisierung mittels $CONTROL^*$ (genauere Erklärung dazu erfolgt in Sektion

3.5.3.3) wird der 16-Bit Lautstärkeparameter mit einer Datenrate von 12.288 Mbit/s auf 6.144 Mbit/s konvertiert. Dies ist erforderlich, da das Serial Peripheral Interface des PGA-2311 laut Datenblatt nur eine maximale Datenrate von 6.25 Mbit/s verarbeiten kann.
Weiters wird in diesem Block aus dem $CLK@12.288\,MHz$ Signal der $CLK@6.144\,MHz$ Takt für die SPI Steuersignallogik abgeleitet.

3.5.3.3 Präambel Decoder

Die vom Bitratenwandler separierte Präambel wird in dieser Sektion ausgewertet und je nach Erkennung einer der gültigen Codewörter (Definitionen siehe Abschnitt 3.4.1.1) wird eine der nachstehenden Aktionen ausgeführt:

- Sync Preamble

  Passiert diese Sektion ohne Folgewirkungen für den Decoder, synchronisiert jedoch die Taktrückgewinnung (vgl. Sektion 3.5.2.2)

- Reset Preamble

  Das Signal $RESET\,RAW$ wird logisch 0 und löst in der zuständigen Reset Logik einen Puls aus, der ein Zurücksetzen des $\Sigma\Delta$ A/D Wandlers und des AES-3 Encoders des Transmitterbausteins bewirkt.

- Control Preamble

  Bei Erkennung einer Controlpräambel wird das Signal $CONTROL^*$ logisch 0 gesetzt und dadurch im Bitratenwandler veranlaßt, den 16-Bit Läutstärkeparameter in den Schieberegistern auf eine Datenrate von 6.144 Mbit/s zu wandeln und als $SPI/SDATA$ an die Lautstärkenregelung weiterzuleiten.
  Der SPI Steuerlogik wird durch die negative Flanke von $CONTROL^*$ ebenso signalisiert, die zum Datentransfer zusätzlich benötigten Signale $SPI/SCLK$ und SPI/CS^* für die PGA-2311 Lautstärkenregelung bereitzustellen.
  Anschließend erfolgt eine Löschung der Bufferschieberegister im Bitratenwandler, um bei weiteren Verschiebungen Bitkombinationen, die irrtümlich als Prämbel interpretiert werden könnten, vorzubeugen.

Die voreingestellten logischen Bitkombinationen der jeweiligen Präambeln sind in der Datei 'defines.h' in Anhang B, die strukturelle Notwendigkeit und Verwendung der Präambeln in Kapitel 3.4.1.1 ausführlich erläutert.

3.5.3.4 Reset Logik

Dieses Modul besteht aus einer monostabilen Kippstufe, die allein dazu dient, aus der negativen Flanke von $RESET\,RAW$ einen Resetpuls $RESET\,IN$ von definierter Zeitdauer (min. 200 μs laut Datenblatt des Cirrus Logic CS 8406) an den Transmitter zu senden und somit den $\Sigma\Delta$ A/D Wandler und den AES-3 Encoder zurückzusetzen.

3.5.3.5 SPI Steuersignallogik

Generiert, im Falle einer negativen Flanke von $CONTROL^*$ die Signale $SPI/SCLK$ und SPI/CS^*, die mit $SPI/SDATA$ des Bitratenwandlers die Spezifikation zur SPI Datenübertragung an die Lautstärkenregelung (Abb. 46) vervollständigen.

3.5.4 Lautstärkenregelung

Diese ist zusammengesetzt aus dem in Kapitel 3.3.3 beschriebenen Chip PGA-2311 und einer separaten, bipolaren Spannungsversorgung (±5 V) für dessen Analogteil.

Der Verstärkungsfaktor eines Mikrophonkanals läßt sich mit folgender Formel berechnen

$$Gain(dB) \quad = \quad \begin{cases} 31.5 \; - \; [0.5\,(255 - N)] & f\ddot{u}r\; 1 \leq N \leq 255;\; N \in \mathbb{N} \\ -\infty \rightarrow Software\; Mute & f\ddot{u}r\; N = 0 \end{cases} \tag{9}$$

N ..... Dezimale Darstellung des 8-Bit Lautstärkeparameters eines Kanals

Die SPI Spezifikationen zum Einstellen des in Gleichung 9 berechneten Verstärkungsfaktors sind in Abbildung 46 illustriert.

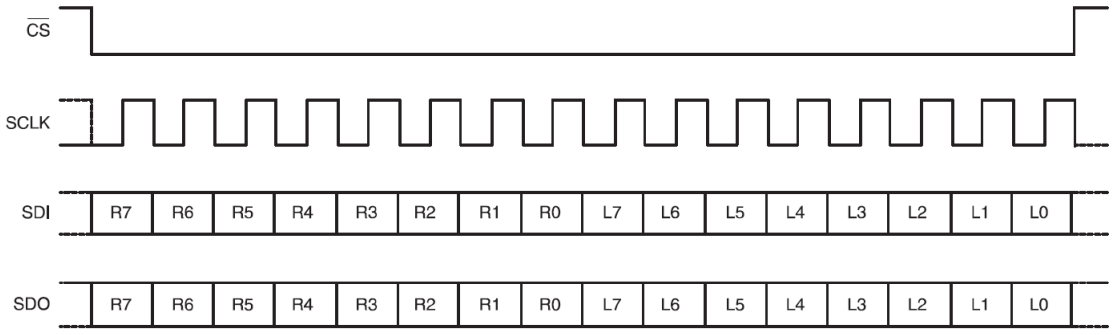


Abbildung 46: SPI Übertragungsformat der PGA-2311 Lautstärkenregelung

4 Frequenzgang der Lautstärkenregelung

Um die Abhängigkeit der Amplitude von der jeweiligen Signalfrequenz zu ermitteln, wurde eine Messung des Frequenzganges der Lautstärkenregelung mithilfe der Software WinMLS der Firma Morset Sound Development durchgeführt.

4.1 Meßaufbau und Einstellungen

Zur Messung selbst wurden folgende Geräte und Software benötigt:

- Meßrechner der Joanneum Research Forschungsgesellschaft mbH
- M-Audio Delta 1010LT Soundkarte (in: XLR3, out: XLR3)
- Lautstärkenregelung aus Abschnitt 3.5.4 (in: 6,35 mm TS Klinke, out: RCA Buchse)
- WinMLS 2004 Professsional Level 5

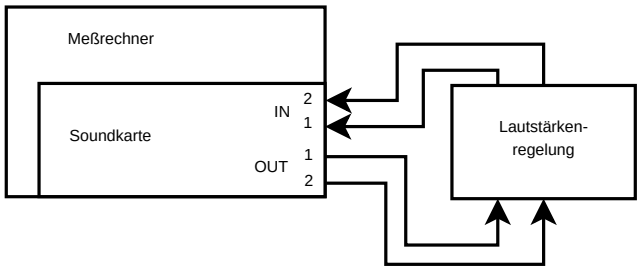


Abbildung 47: Skizze des Meßaufbaus

Die Einstellungen für die einzelnen Meßvorgänge waren, sofern nicht explizit anders erwähnt:

- Samplingfrequenz: 48 kHz
- Sampleauflösung: 16-Bit
- Meßmethode: -10 dBFS Single Sweep 16 Hz - 24 kHz
- Anzahl der Kanäle: 2
- Emphasis: nein

4.2 Meßergebnisse

Die nachfolgenden Ergebnisse wurden um die gemessene Frequenzantwort der verwendeten Soundkarte korrigiert. Ebenso sind die Pegel des Soundkartenausgangs auf 0 dBFS Aussteuerung hochskaliert worden.

4.2.1 Software Mute (Lautstärkeparameter=0dec)

Die ersten dargestellten Meßergebnisse in Abbildung 48 betreffen die Software Mute Funktion der Lautstärkenregelung. Der Rauschteppich im Software Mute Zustand zeigt keinerlei auffällige Spitzen im Frequenzverlauf der Schaltung. Weiters bleiben Störeinflüsse durch die Netzfrequenz und ihrer Harmonischen unter der -90 dB Schwelle.

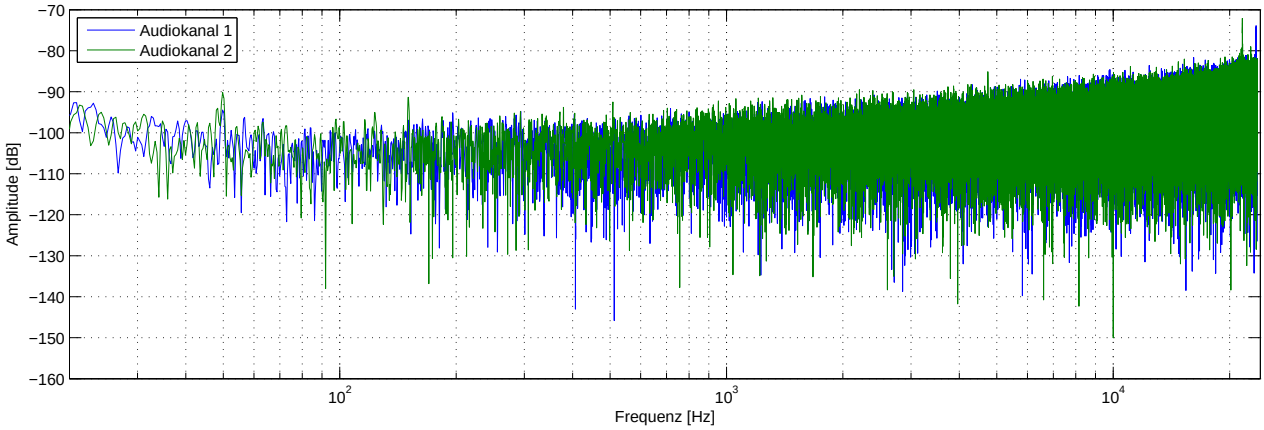


Abbildung 48: Frequenzantwort der Lautstärkenregelung @ 000dec

4.2.2 0 dB Verstärkung (Lautstärkeparameter=192dec)

Der Frequenzgang der Lautstärkenregelung bei 0 dB Verstärkung zeigt die Signalverfälschungen des Loop-Through Modus, welche in Abbildung 49 veranschaulich werden. In der momentanen Konfiguration (vgl. Abbildung 31 in Kapitel 3.2) stellt die 0 dB Einstellung die kleinstmögliche Signalabschwächung der Audiokanäle dar. Auch hier sind die maximalen Abweichungen marginal und auf ca. -0,4 dB bei tiefen Frequenzen bzw. ca. +0,1 dB zu hohen Frequenzen hin beschränkt.

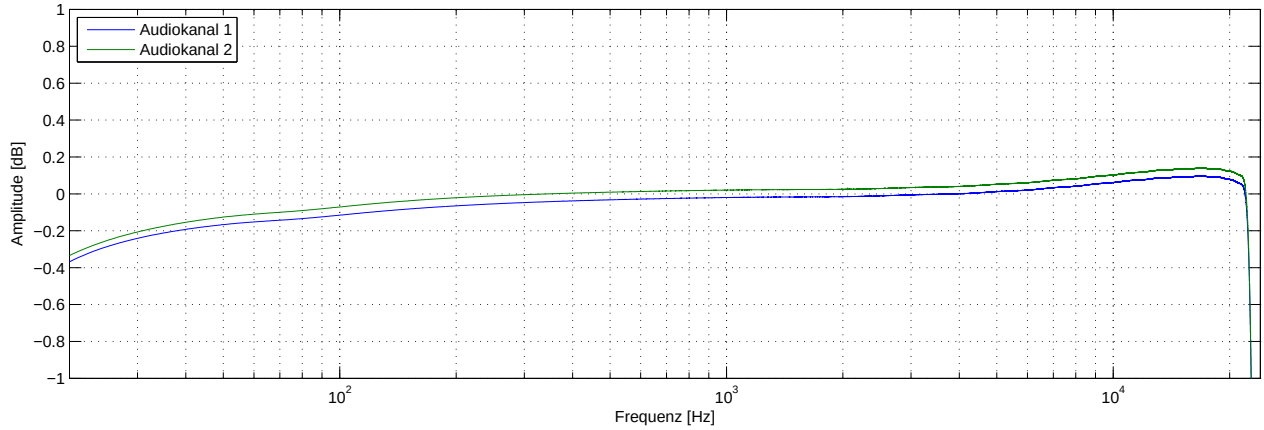


Abbildung 49: Frequenzantwort der Lautstärkenregelung @ 192dec

4.2.3 Verstärkung des Audiosignals mittels der Lautstärkenregelung

Wie in Kapitel 5.2.1 noch erläutert wird, wäre es auch möglich die Lautstärkenregelung zum Mikrophonvorverstärker umzufunktionieren. Aus diesem Grund werden nachfolgend einige Messungen bei höheren Verstärkungsfaktoren illustriert, die mit veränderten Meßeinstellungen für die Eingangspegel der Lautstärkenregelung durchgeführt worden sind.

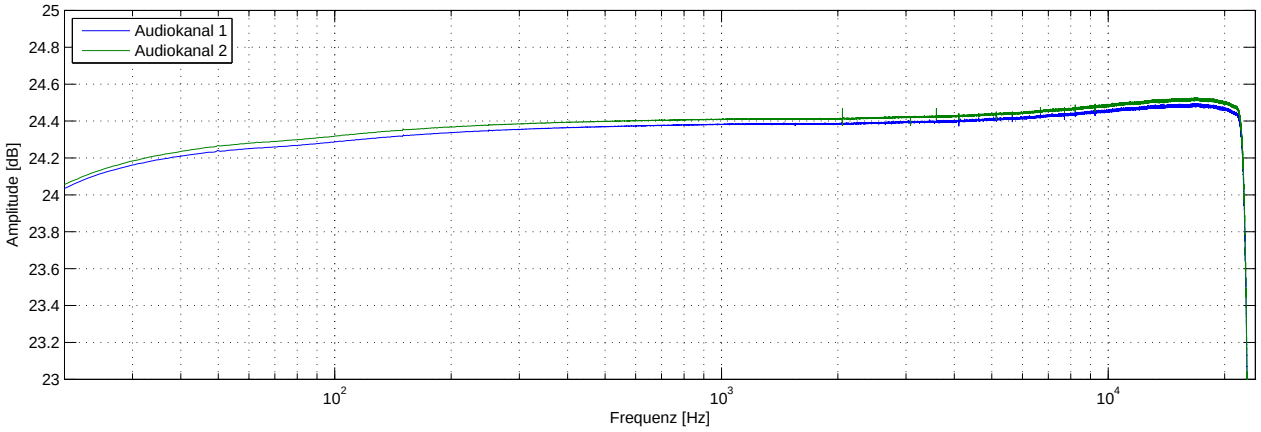


Abbildung 50: Frequenzantwort der Lautstärkenregelung @ 241dec (+24,5 dB)

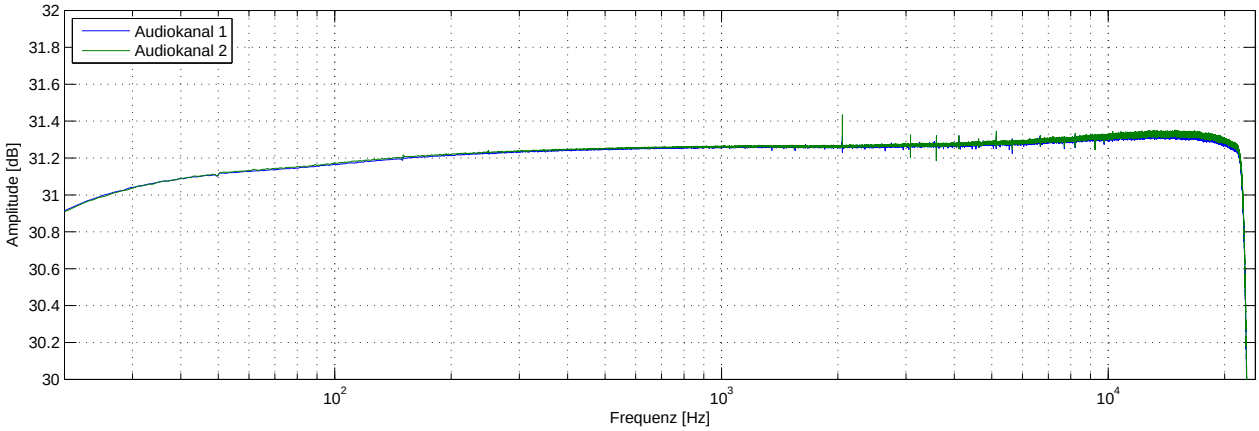


Abbildung 51: Frequenzantwort der Lautstärkenregelung @ 255dec (+31,5 dB)

Aus den Abbildungen 50 und 51 wird ersichtlich, daß bei Verstärkungen jenseits der 20 dB Glitches im Frequenzverlauf sichtbar werden. Diese Artefakte sind aber höchstwahrscheinlich auf die verminderten Eingangspegel von -30 bzw. -37 dBFS an den unsymmetrischen Eingängen der Lautstärkenregelung zurückzuführen die angewandt wurden, um Clipping in der Meßkette zu vermeiden.

5 Zusammenfassung und mögliche Verbesserungen

Dieses Kapitel soll eine kurze chronologische Darstellung der Tätigkeiten und die sich nach der Entwicklung und Test des Prototypen aufdrängenden Verbesserungsmöglichkeiten beinhalten.

5.1 Zusammenfassung

Am Anfang dieser Diplomarbeit stand das Einlesen in die Dokumentation des bestehenden Systems. Durch dessen Verlässlichkeit wurde die Entscheidung getroffen, dieses um die gewünschten Fernwartungsfunktionen zu erweitern.
Die naheliegendste Idee war die einzelnen Basissysteme über den Lichtwellenleiter, der für die Zuführung des Masterclocks verantwortlich ist, fernzusteuern.
Da die Reinheit dieses Taktes für die Funktion des $\Sigma\Delta$ A/D Wandlers und des AES-3 Encoders von essentieller Bedeutung sind, mußte eine Möglichkeit gefunden werden, diese trotz der Übertragung der Steuerdaten beizubehalten. Nach Studium diverser Literatur über Leitungscodierung fiel die Auswahl auf eine Codierung dieser Daten mit dem Manchester Code, der einerseits einen simplen Codier- und Decodiervorgang aufweist wie auch per Definition stattfindende Pegelwechsel zur akribischen Rückgewinnung des Taktes. Der dabei auftretende Nachteil, daß dieser Code nicht spektral effizient ist, war für die Realisierung nicht von Bedeutung, da jedem Basissystem ein eigener Lichtwellenleiter dezidiert für die Übertragung, des mit den Steuerdaten modulierten Taktsignals, zur Verfügung steht.
Mit dieser fundamentalen Festlegung konnte erstmals die Benennung und Ausformung der notwendigen Module (Encoder, Decoder, Taktrückgewinnung und Lautstärkenregelung) zur Erweiterung der Basissysteme gestaltet werden.
Danach folgte eine Recherche über bereits angebotene Teilsysteme, die den Aufbau der vorhergenannten Module erleichtern sollten. Dabei stellte sich die nicht zufriedenstellende Performance der angebotenen digitalen Potentiometer heraus, die den Bauteilaufwand zusätzlich in die Höhe trieb, um die SPI Spezifikationen der schlußendlich implementierten Lautstärkenregelung zu erfüllen.
Weiteres Aneignen von Wissen über das Serial Peripheral Interface und Möglichkeiten der Taktrückgewinnung waren die Folge.
Der theoretische Entwurf der Hardwaremodule erfolgte händisch mithilfe der zugehörigen technischen Datenblätter der einzelnen Bausteine.
Getestet wurde der auf Lochraster gelötete Hardware-Prototyp durch Ansteuerung mittels Software, die dem exzellenten SDK des IO-Warriors beigelegt ist, was eine getrennte Entwicklung von Hard- und Software ermöglichte.
Es folgten der Entwurf und die Dokumentation der Software zur Steuerung des Master Encoders.
Zu guter Letzt wurden mit dem Gesamtpaket von Hard- und Software die Frequenzgangmessungen durchgeführt, und Überlegungen über zukünftige Verbesserungen angestellt, die im nachfolgenden Kapitel ausgeführt sind.

5.2 Zukünftige Verbesserungen

Durch die universelle und modulare Gestaltung der Erweiterung der Basissysteme ist es ohne große Veränderungen möglich zukünftige Verbesserungen zu implementieren, von denen zwei in den Kapiteln 5.2.1 bzw. 5.2.2 vorgestellt werden.

5.2.1 Mikrophonvorverstärker vs. ausgebaute Lautstärkenregelung

In der jetzigen Ausführung des erweiterten Basissystems besteht eine Trennung von Mikrophonvorverstärker und Lautstärkenregelung. Der Mikrophonvorverstärker wird momentan zur konstanten Vorverstärkung des Audiosignals, zur Phantomspeisung der Mikrophone und zum XLR-Interfacing eingesetzt.
Die zukünftige Änderung zielt darauf ab, diese Aufgaben direkt an die Lautstärkenregelung zu übertragen und damit den Mikrophonvorverstärker entbehren zu können (siehe Abbildung 52).

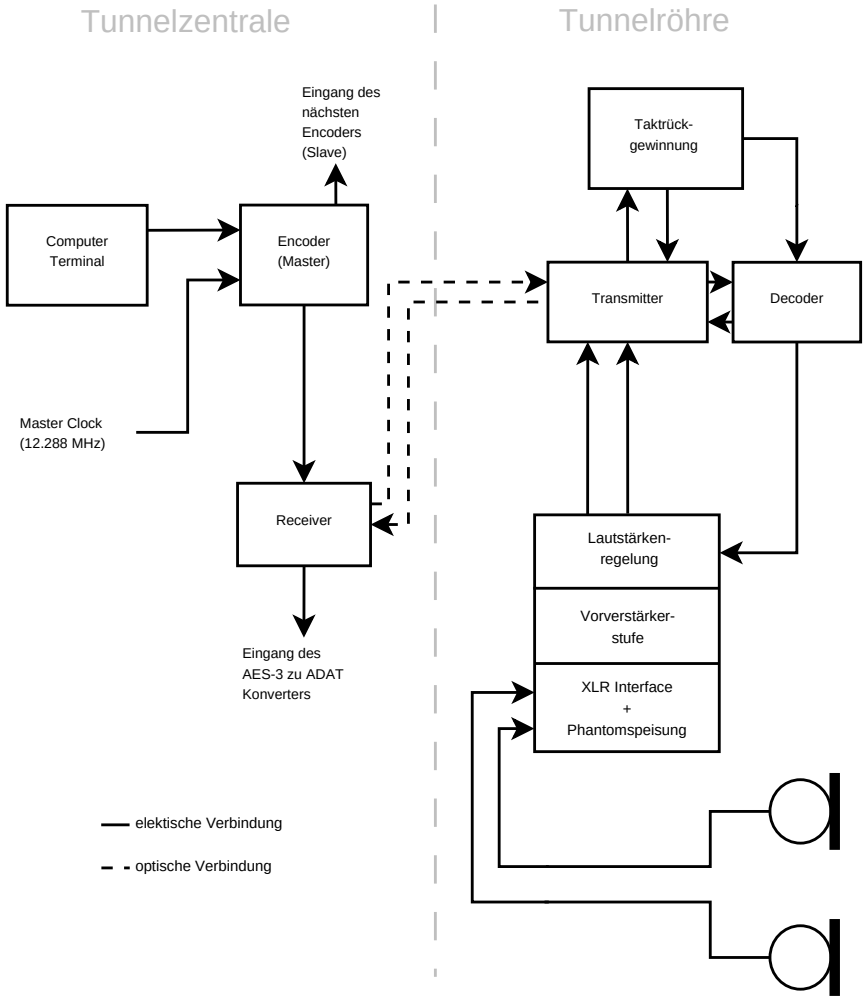


Abbildung 52: Erweitertes Basissystem mit vollständig ausgebauter Lautstärkenregelung

Dazu würde vor die PGA-2311 Lautstärkenregelung eine rauscharme Vorverstärkerstufe implementiert werden, welche die symmetrischen Mikrophonsignale, die nach dem XLR-Interface zur Verfügung stehen, direkt übernimmt.
Weiters müßte noch die Schaltung zur Phantomspeisung der angeschlossenen Mikrophone integriert werden, was an die rauscharme Vorverstärkerstufe noch die zusätzliche Anforderung der hohen Gleichtaktunterdrückung stellt. Da diese Vorverstärkerstufe üblicherweise mit höheren Betriebsspannungen als ±5 V betrieben wird, z.B ±15 V, sollte auch darüber nachgedacht werden, den PGA-2311 durch die vollkommen pin- und funktionskompatiblen Varianten PGA-2310 oder PGA-2320 zu ersetzen, die ebenso mit ±15 V Betriebsspannung versorgt werden. Auf diese Weise wäre es möglich, statt 5

Versorgungsspannungen (±5 V, ±15 V, +5 V_{dig}) zur Verfügung zu stellen, auch mit 3 (±15 V, +5 V_{dig}) ein Auslangen zu finden.

5.2.2 Diskreter Aufbau vs. Mikrocontroller

Die Blöcke des Decoders zur Manchester Decodierung sowie die SPI Logik könnten theoretisch durch die Verwendung eines Mikrocontrollers ersetzt werden, da diese Bausteine, im Vergleich zur Taktrückgewinnung, nicht mehr zu den zeitkritischen Blöcken dieser Schaltung zu zählen sind. Dies würde den Chip Count auf der Transmitterplatine drastisch reduzieren, und so zur Verringerung der Platinengröße beitragen.
Der Einfachheit halber sollte der µC das SPI 'Schnittstellenprotokoll' beherrschen, um direkt mit der Lautstärkenregelung kommunizieren zu können und diese Funktionalität nicht erst eigens über die Programmierung emulieren zu müssen.
Außerdem müßten die A/D Wandler an den µC-Eingängen mit einer Frequenz größer als 24,576 MHz arbeiten, um das mit der Taktfrequenz von 12,288 MHz manchestercodierte Datensignal korrekt einlesen und auf ihre Gültigkeit prüfen zu können. Diese Anforderung an die Eingänge des µCs impliziert aber, daß der Systemtakt selbst, mit dem der Mikrocontroller betrieben werden müßte, auch ein Vielfaches von 24,576 MHz sein sollte, um µC-Eingangsdaten zu garantieren, die wiederum immer an denselben informationstragenden Stellen einer Manchester-Bitzelle abgetastet werden. Eine zufriedenstellende Variante diese geforderte Frequenzvervielfachung zu realisieren wäre eine Frequenzsynthese des Ausgangssignals der Taktrückgewinnung, da in diesem Falle auch die Phasenstarrheit bei der Abtastung der Daten gegeben ist.

Literatur

[Kos05] Markus Koschier, Aufbereitung von Audiosignalen zur Übertragung in optischen Netzen, Graz 2005

[Grf02] Franz Graf, Design and implementation of a system for an automatic classification of acoustic signals, Graz 2002

[Grf04] Franz Graf, System und Verfahren zur automatischen Überwachung eines Verkehrsweges (European Patent EP1376510), 2004

[Rop06] Carsten Roppel, Grundlagen der digitalen Kommunikationstechnik: Übertragungstechnik - Signalverarbeitung - Netze, Hanser Fachbuchverlag; 1. Auflage, (2. März 2006)

[Wer09] Martin Werner, Nachrichtentechnik: Eine Einführung für alle Studiengänge, Viweg+Teubner; 6. Auflage, (15. Januar 2009)

[Rup05] Marcel Rupf, Nachrichtentechnik und Mobilkommunikation Vorlesungsskript, WS 2005/06, https://sites.zhaw.ch/~rur/ntm/unterlagen/ntmkap53leicode.pdf

[Gra06] Gerhard Graber, Digitale Audiotechnik 1 Vorlesungsskript, WS 2006/07

[EBU95] European Broadcasting Union, Engineering guidelines for the EBU/AES digital audio interface, Genf 1995, http://tech.ebu.ch/docs/other/aes-ebu-eg.pdf

[Ale07] Alejo2083, Wikipedia, http://commons.wikimedia.org/wiki/File:Biphase_Mark_Code.svg

[AES03] Audio Engineering Society, AES Recommended Practice for Digital Audio Engineering - Serial Multichannel Audio Digital Interface (MADI), http://www.iis.ee.ethz.ch/~felber/DataSheets/AES-EBU/aes10-2003.pdf

[Alb10] Frank Albert, Das ADAT Optical Interface, http://www.albert-av.de/htm/adat/interface.htm

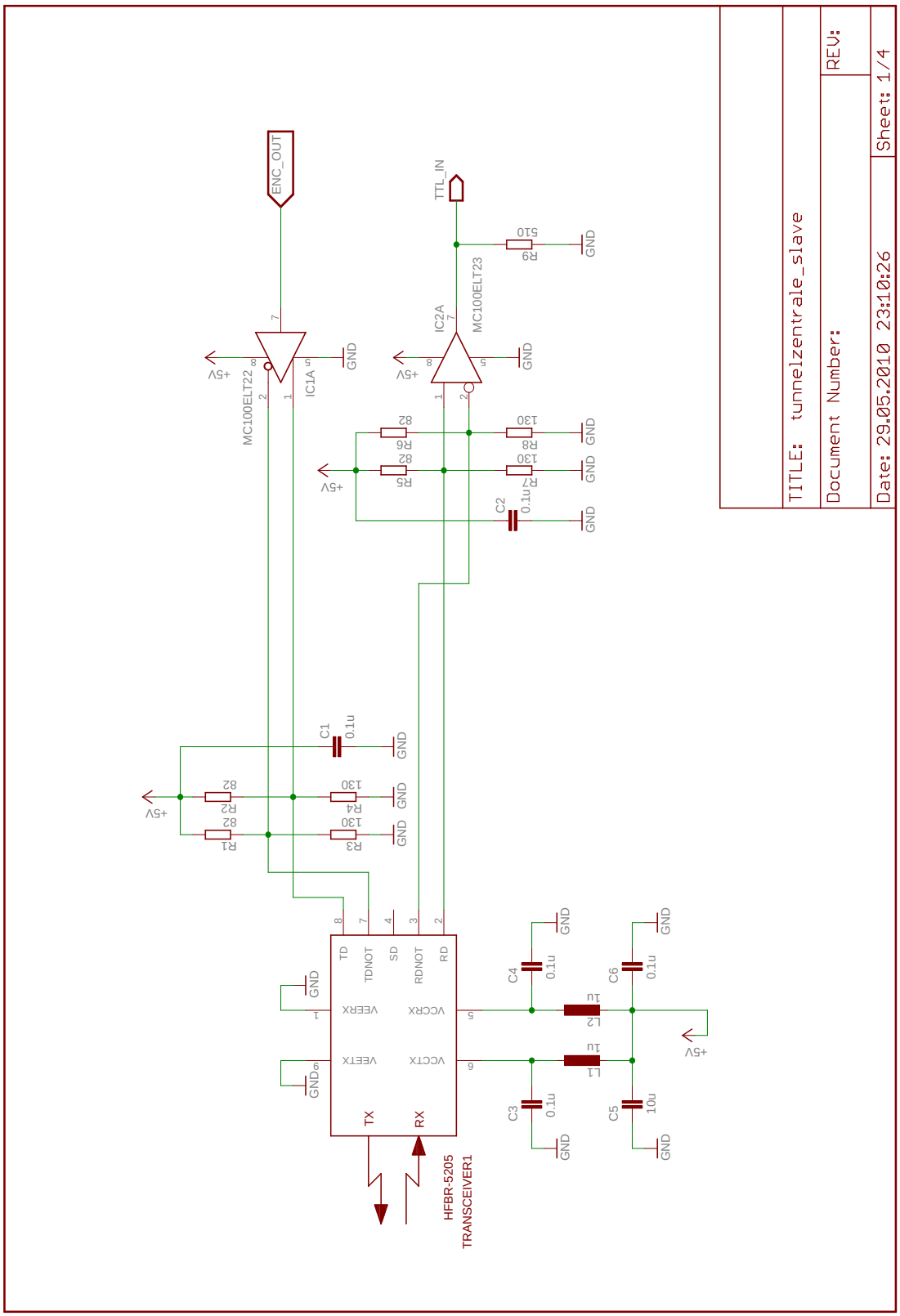
[Cbu06] Cburnett, Wikipedia, http://en.wikipedia.org/wiki/File:SPI_three_slaves.svg, http://en.wikipedia.org/wiki/File:SPI_three_slaves_daisy_chained.svg

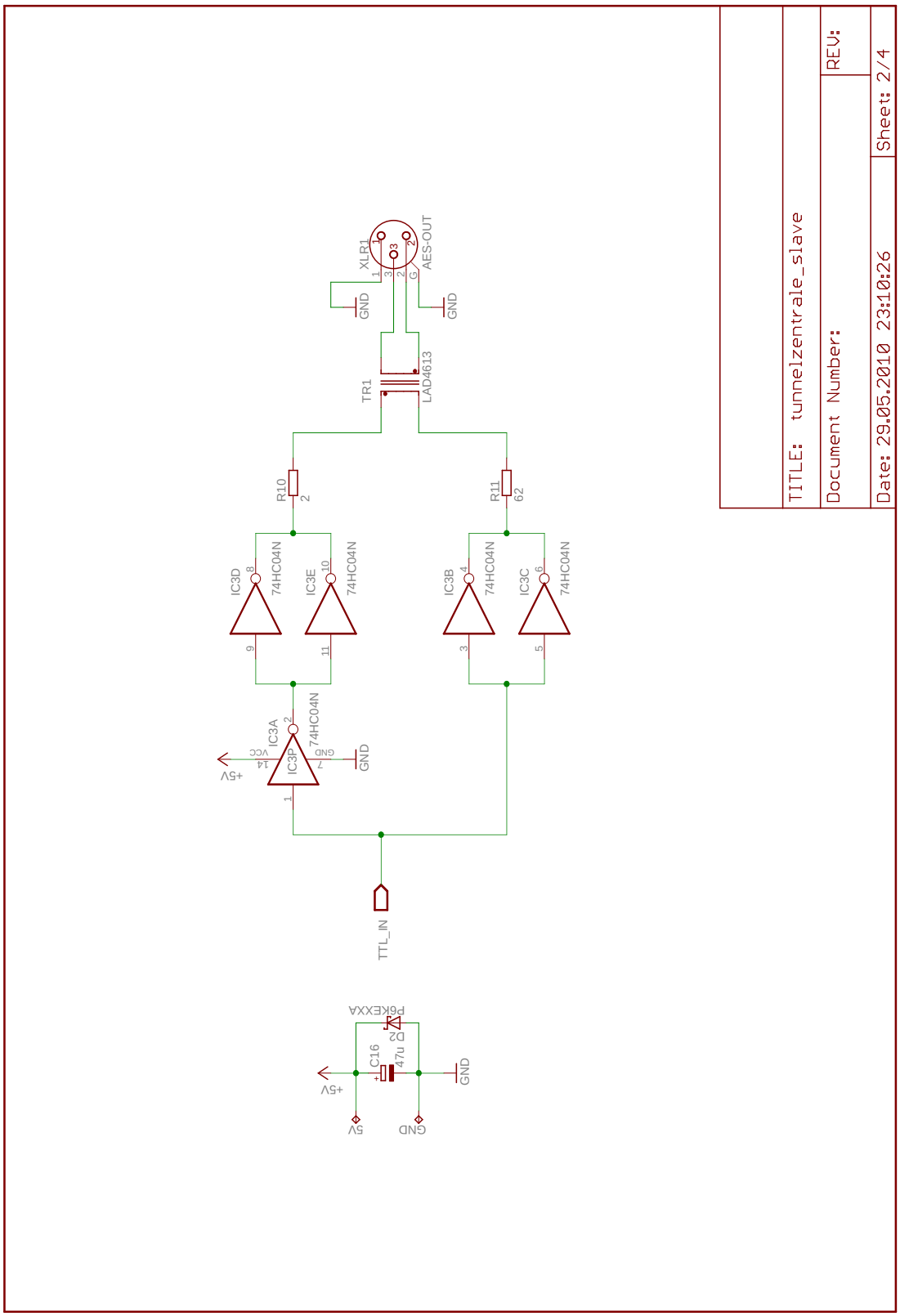
[Fre03] Freescale Semiconductor Inc, SPI Block Guide V03.06. 2003, http://www.datasheetarchive.com/pdf-datasheets/Datasheets-12/DSA-222613.pdf

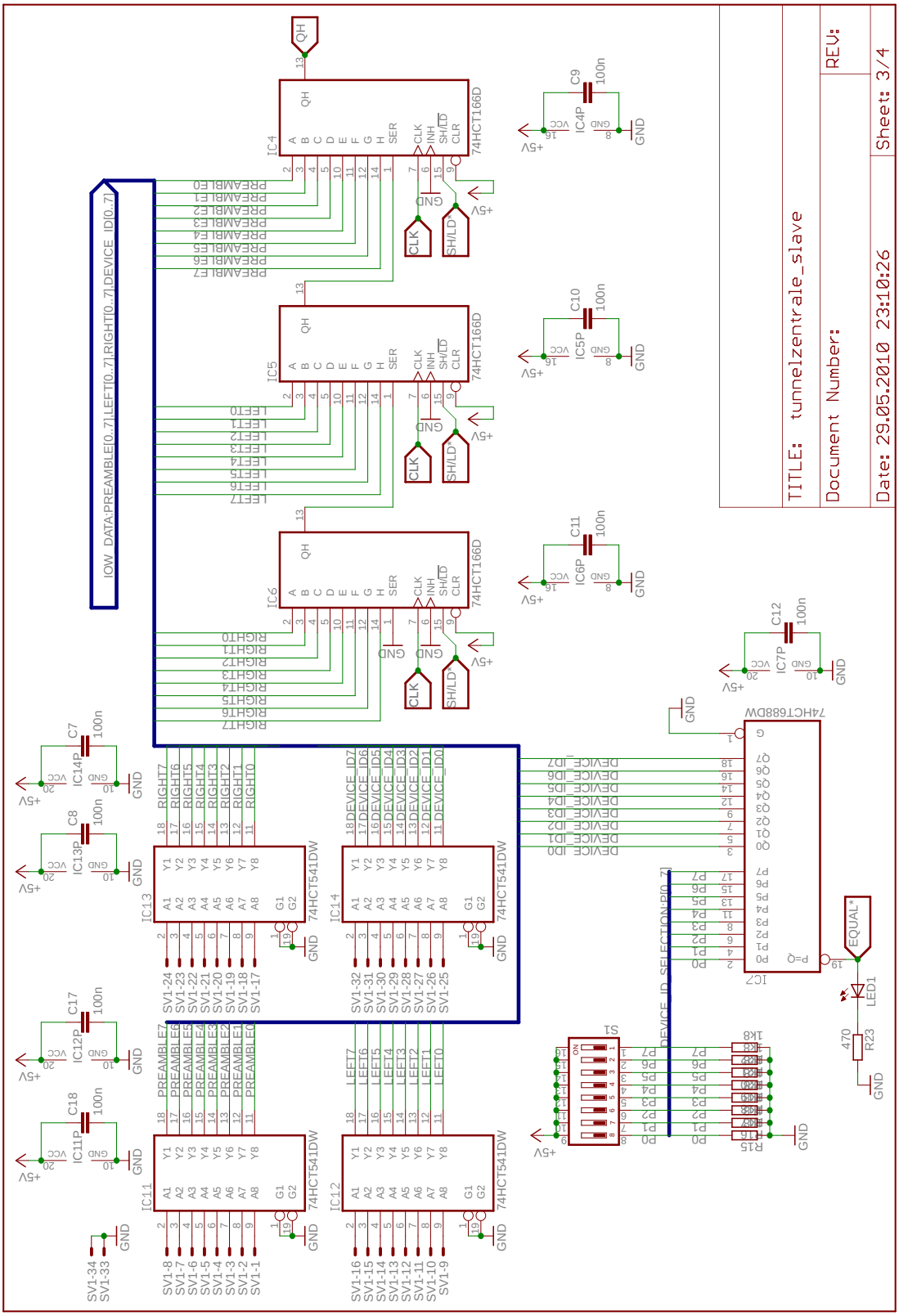
[Kam92] Karl Dirk Kammeyer, Nachrichtenübertragung, B. G. Teubner Stuttgart; 1. Auflage, 1992

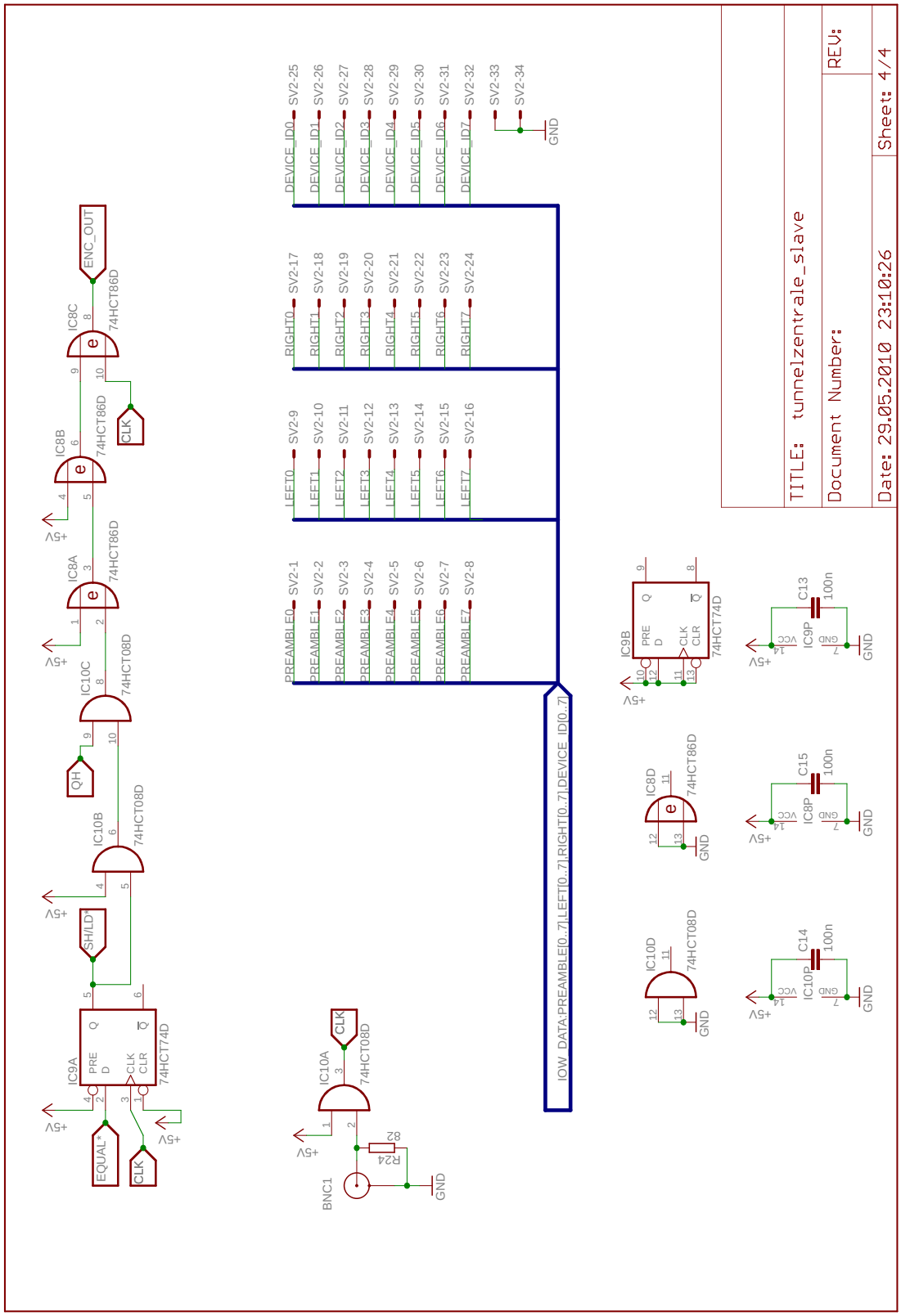
[Wal02] Rick Walker, Clock and Data Recovery for Serial Digital Communication, 2002, http://omnisterra.com/walker/pdfs.talks/ISSCC2002.pdf

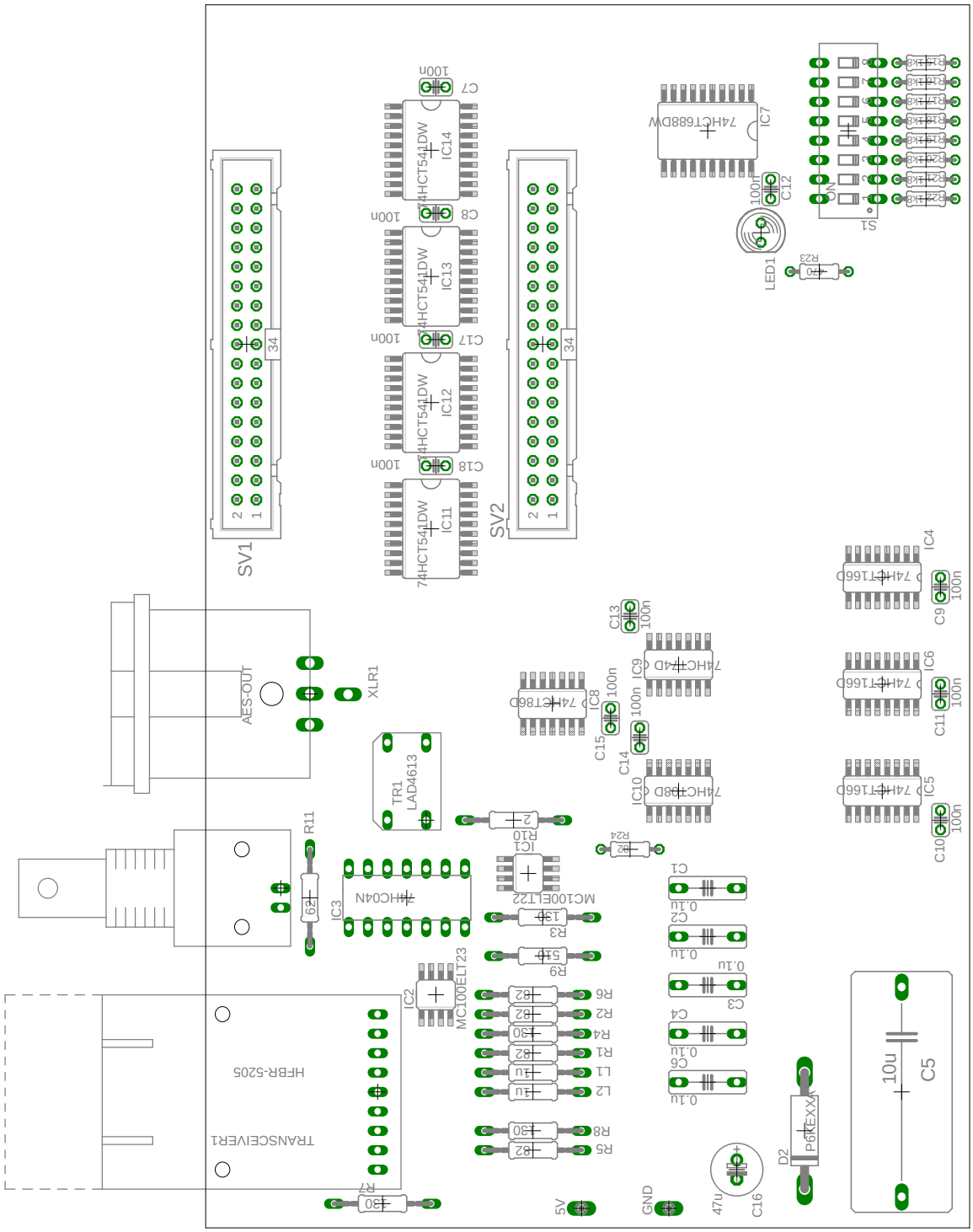
Anhang A: Abbildungen zur Erläuterung und Fertigung der elektronischen Schaltungen (EAGLE - Schematics und Board Layout Dateien)

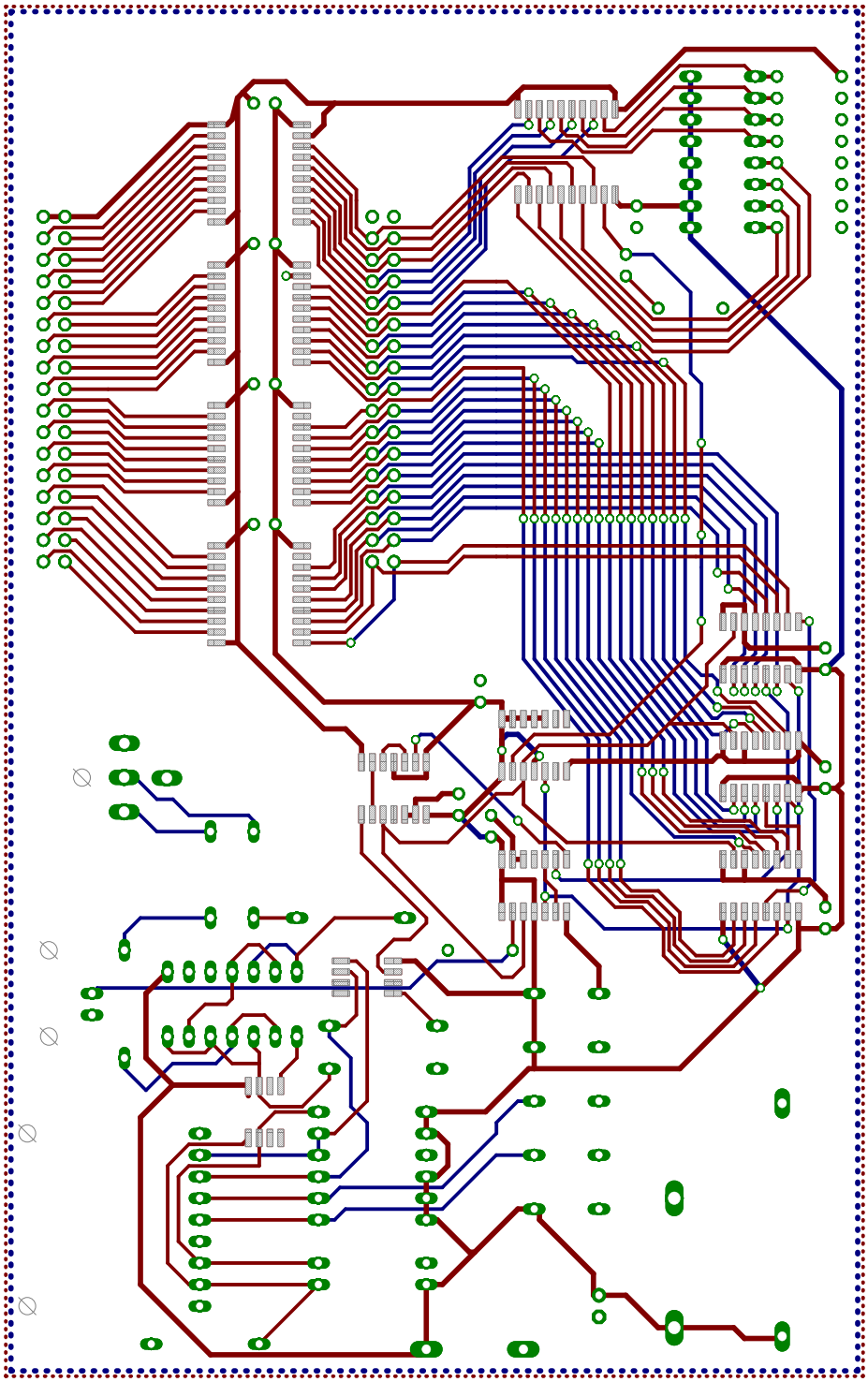

TITLE: tunnelzentrale_slave
Document Number:
REV:
Date: 29.05.2010 23:10:26
Sheet: 1/4
ENC_OUT
TTL_IN
MC100ELT22
MC100ELT23
IC1A
IC2A
+5V
GND
R1
R2
R3
R4
R5
R6
R8
R9
82
130
510
C1
C2
C3
C4
C5
C6
0.1u
2.7n
10u
1u
L1
L2
TD
TDNOT
SD
RDNOT
RD
VEERX
VCCRX
VEETX
VCCTX
TX
RX
HFBR-5205
TRANSCEIVER1


TITLE: tunnelzentrale_slave
Document Number:
REV:
Date: 29.05.2010 23:10:26
Sheet: 2/4
XLR1
AES-OUT
GND
GND
TR1
LAD4613
R10
2
R11
62
IC3D
74HC04N
IC3E
74HC04N
IC3B
74HC04N
IC3C
74HC04N
IC3A
74HC04N
IC3P
VCC
GND
+5V
GND
TTL_IN
P6KEXXA
D2
C16
47u
+5V
GND
5V
GND


TITLE: tunnelzentrale_slave
Document Number:
REV:
Date: 29.05.2010 23:10:26
Sheet: 3/4
IOW DATA:PREAMBLE[0..7],LEFT[0..7],RIGHT[0..7],DEVICE_ID[0..7]
74HCT166D
74HCT541DW
74HCT688DW
100n
1k8
470
LED1
EQUAL*
+5V
GND


TITLE: tunnelzentrale_slave
Document Number:
REV:
Date: 29.05.2010 23:10:26
Sheet: 4/4
IOW_DATA:PREAMBLE[0..7],LEFT[0..7],RIGHT[0..7],DEVICE_ID[0..7]
DEVICE_ID0 DEVICE_ID1 DEVICE_ID2 DEVICE_ID3 DEVICE_ID4 DEVICE_ID5 DEVICE_ID6 DEVICE_ID7
RIGHT0 RIGHT1 RIGHT2 RIGHT3 RIGHT4 RIGHT5 RIGHT6 RIGHT7
LEFT0 LEFT1 LEFT2 LEFT3 LEFT4 LEFT5 LEFT6 LEFT7
PREAMBLE0 PREAMBLE1 PREAMBLE2 PREAMBLE3 PREAMBLE4 PREAMBLE5 PREAMBLE6 PREAMBLE7
SV2-25 SV2-26 SV2-27 SV2-28 SV2-29 SV2-30 SV2-31 SV2-32
SV2-17 SV2-18 SV2-19 SV2-20 SV2-21 SV2-22 SV2-23 SV2-24
SV2-9 SV2-10 SV2-11 SV2-12 SV2-13 SV2-14 SV2-15 SV2-16
SV2-1 SV2-2 SV2-3 SV2-4 SV2-5 SV2-6 SV2-7 SV2-8
SV2-33 SV2-34
GND
IC9A 74HCT74D
IC9B 74HCT74D
IC10A 74HCT08D
IC10B 74HCT08D
IC10C 74HCT08D
IC10D 74HCT08D
IC8A 74HCT86D
IC8B 74HCT86D
IC8C 74HCT86D
IC8D 74HCT86D
IC9P C13 100n
IC8P C15 100n
IC10P C14 100n
EQUAL*
SH/LD*
CLK
QH
ENC_OUT
BNC1
R24 82
+5V
GND

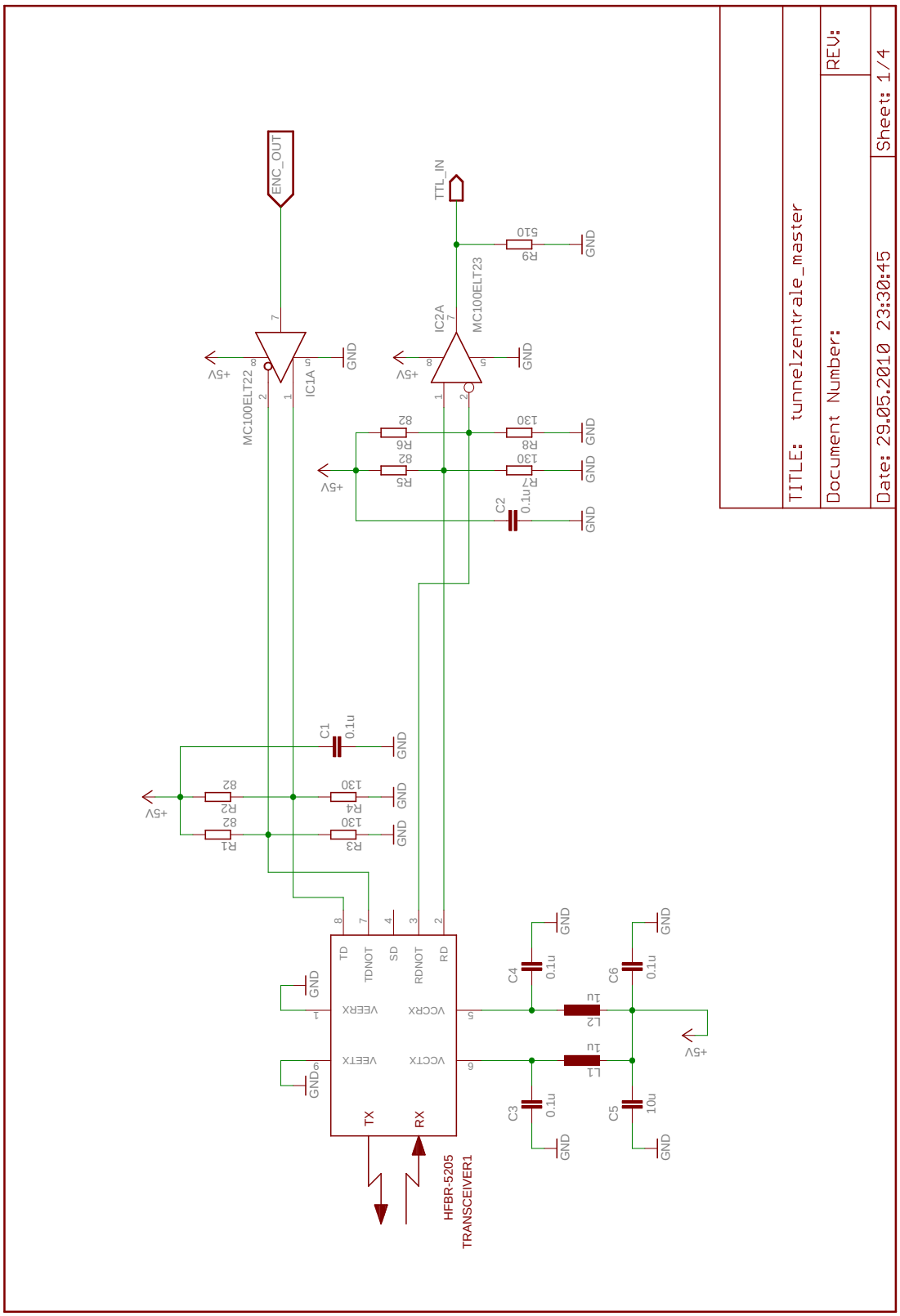

SV1
SV2
34
74HCT541DW
IC14
IC13
IC12
IC11
C7
C8
C17
C18
100n
74HCT688DW
IC7
C12
LED1
R23
S1
AES-OUT
XLR1
TR1
LAD4613
74HCT86D
IC8
C15
C13
74HC74D
IC9
C14
IC10
74HCT166D
IC4
IC6
IC5
C9
C11
C10
R11
IC3
74HC04N
IC1
MC100ELT22
R10
R24
R3
R9
IC2
MC100ELT23
R6
R2
R4
R1
L1
L2
R8
R5
R7
HFBR-5205
TRANSCEIVER1
C1
C2
C3
C4
C6
0.1u
10u
C5
D2
P6KE
C16
47u
5V
GND

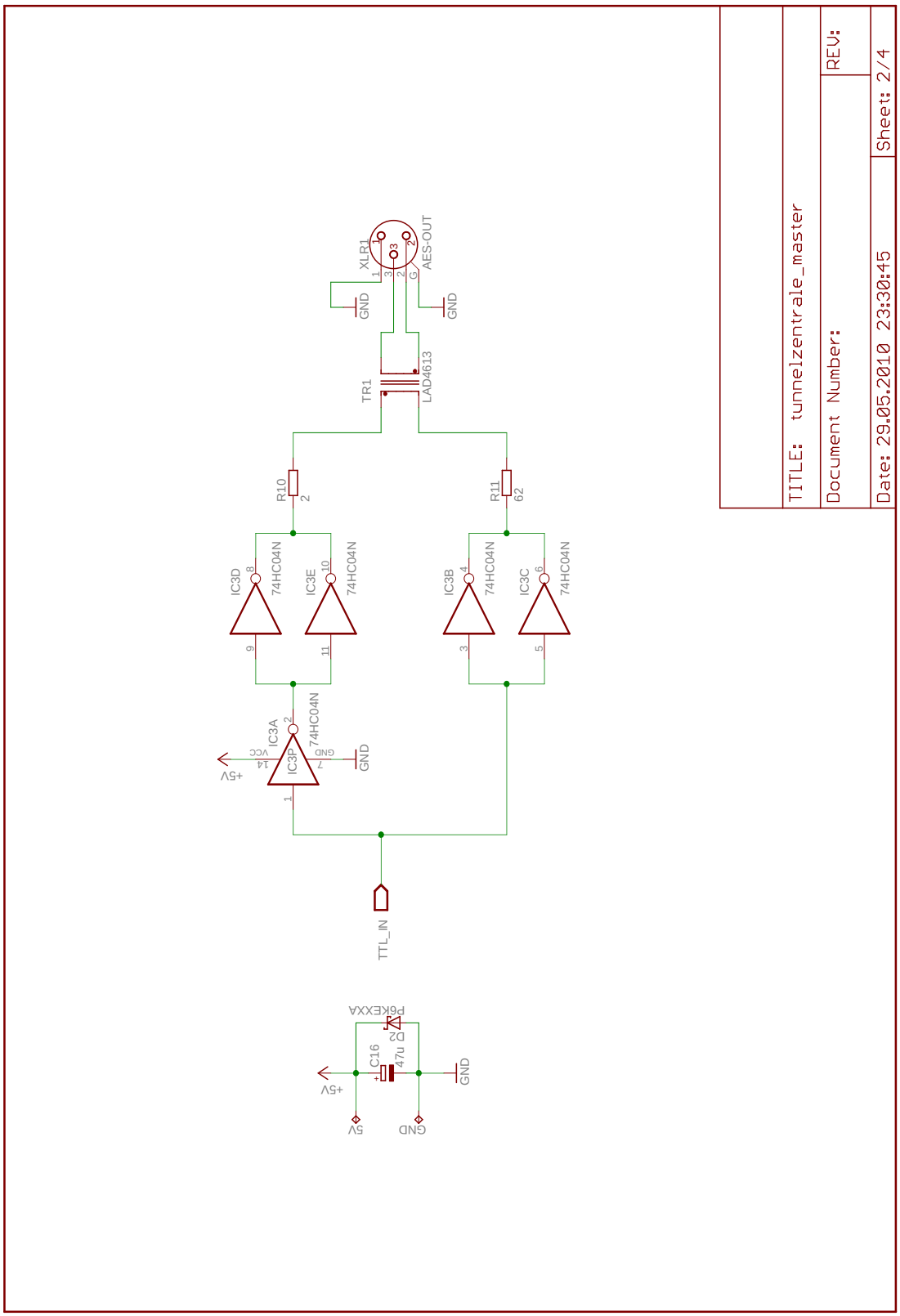


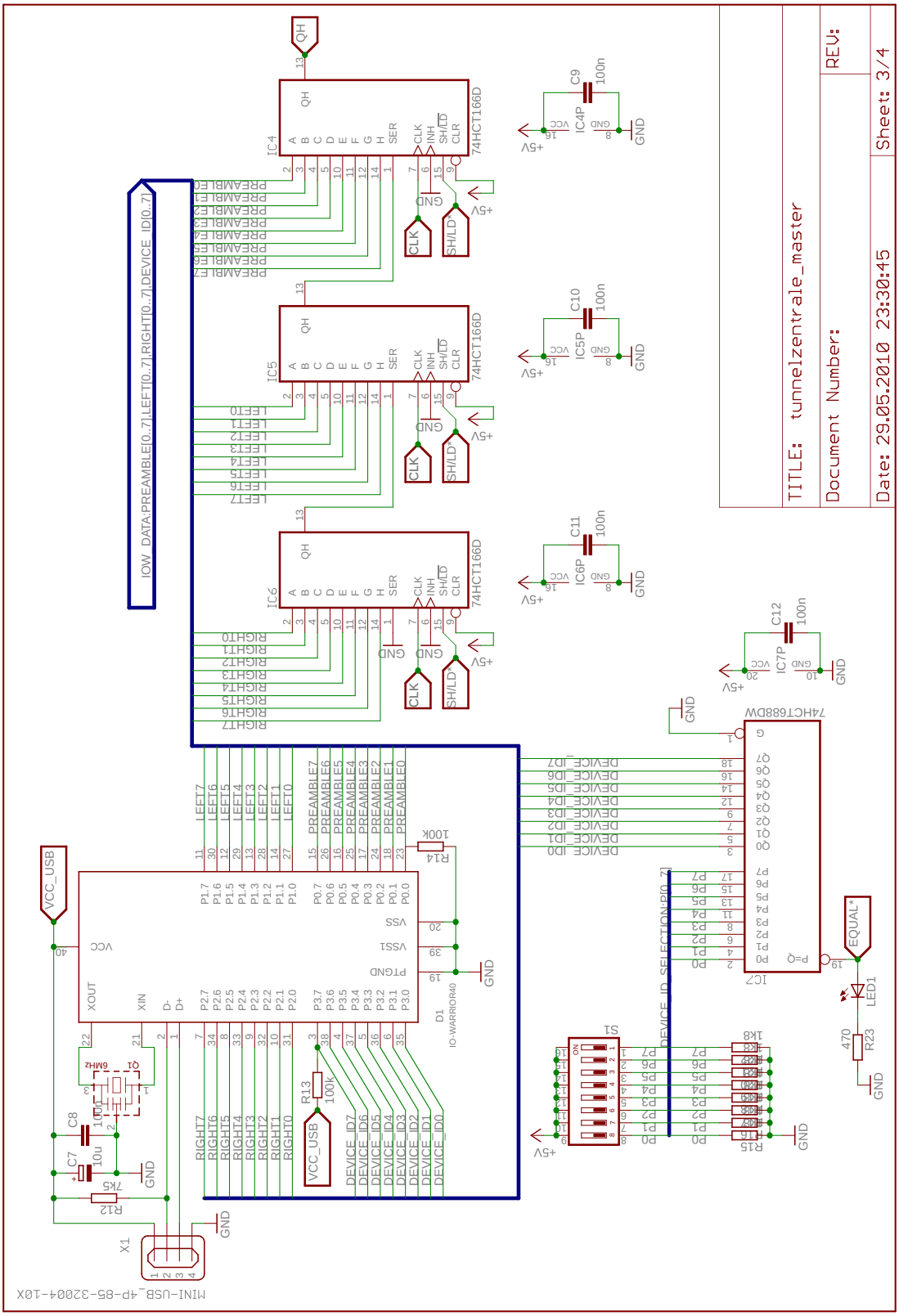
System: TUNNELZENTRALE_SLAVE; Platinendimensionen: 160x100 mm

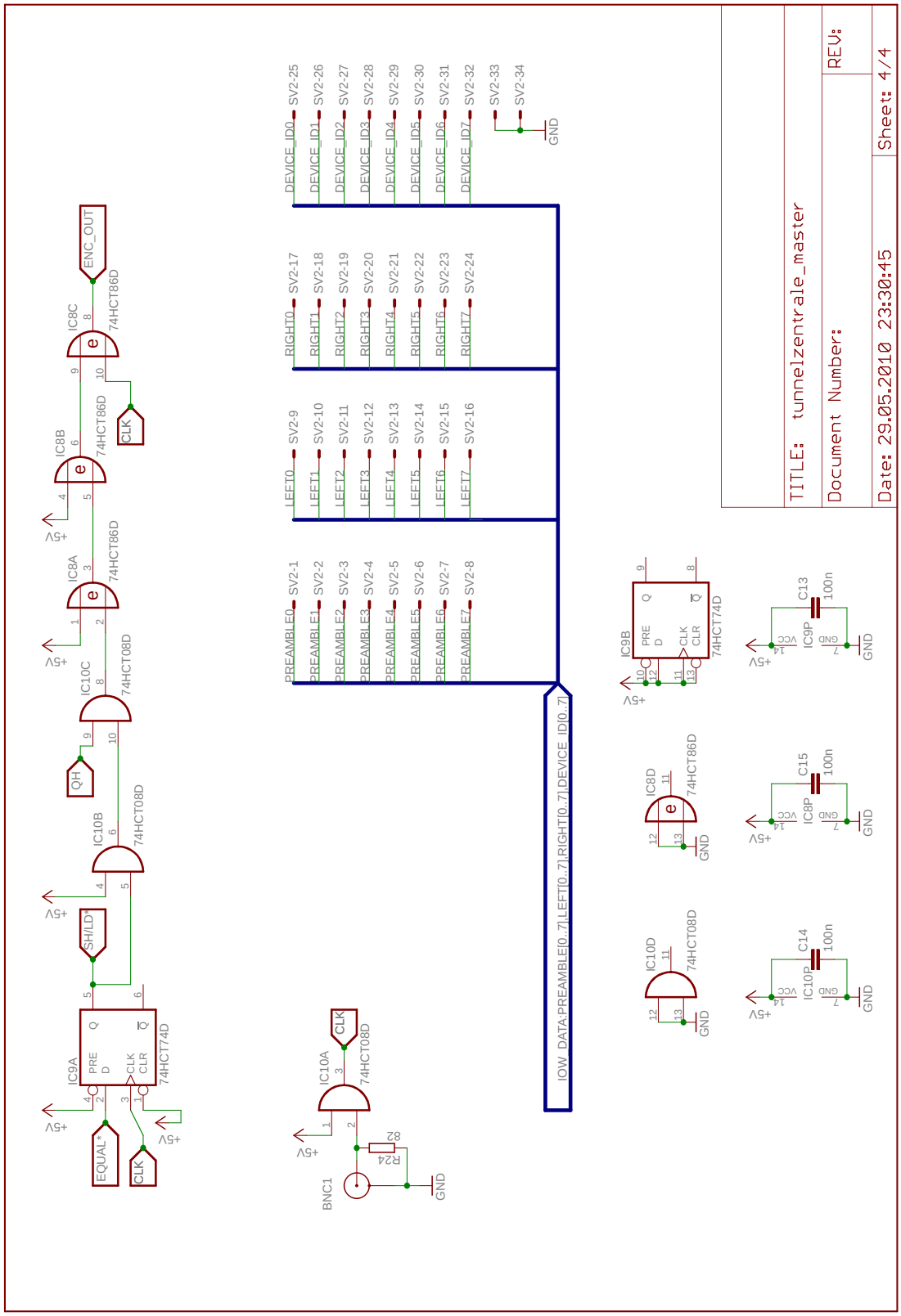
Bauteilliste:

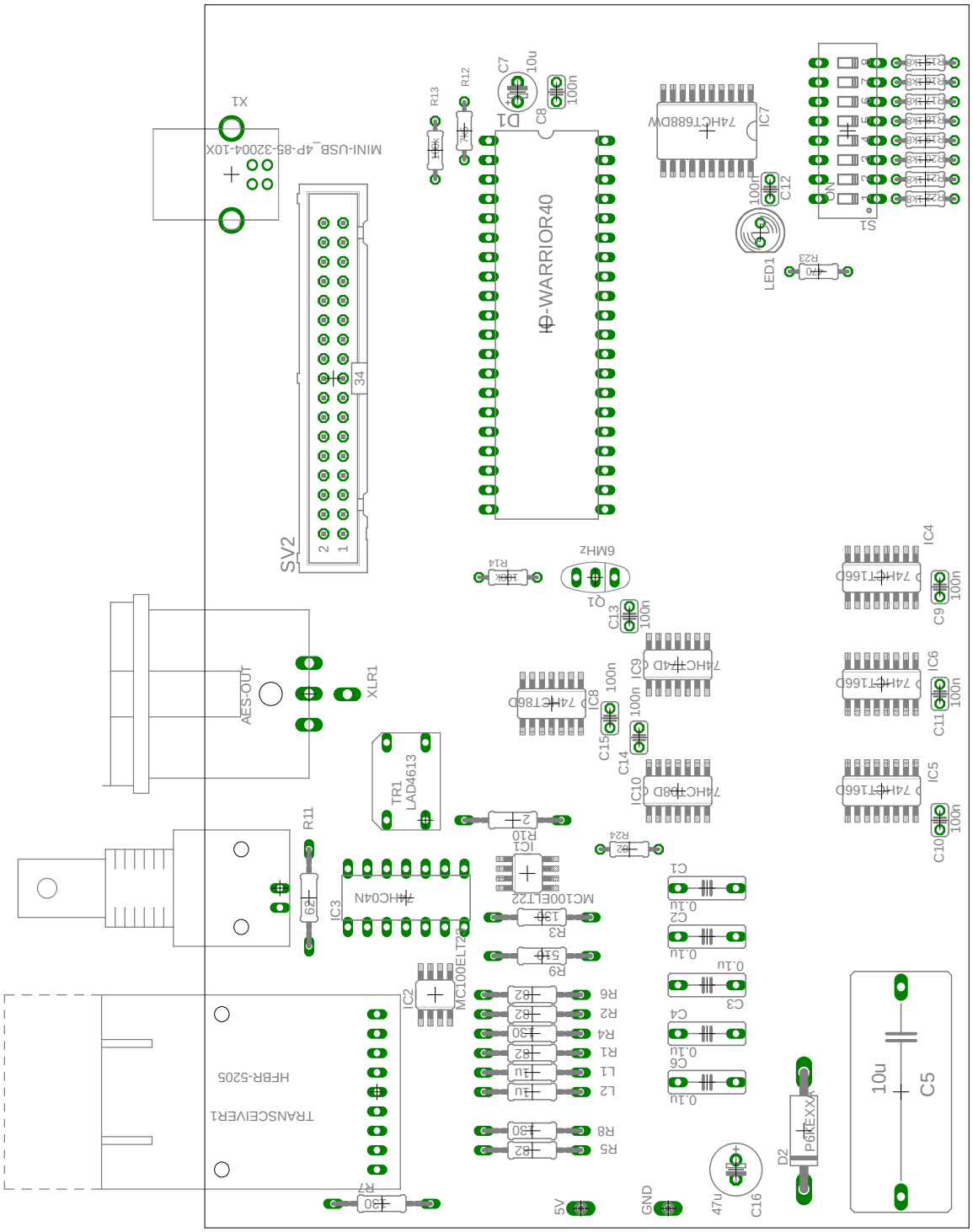
Menge	Wert	Device	Bauteile
1		BNC90	BNC1
1		DIP08YL	S1
1		LED5MM	LED1
2		ML34E	SV1, SV2
5	0.1u	C-EU075-032X103-OVAL	C1, C2, C3, C4, C6
8	1k8	R-EU_0204/7	R15, R16, R17, R18, R19, R20, R21, R22
2	1u	L-EU0207/12-OVAL	L1, L2
1	2	R-EU_0207/12-OVAL	R10
1	10u	C-EU275-134X316-OVAL	C5
1	47u	CPOL-EUE2.5-7	C16
1	62	R-EU_0207/12-OVAL	R11
1	74HC04N	74AC04N	IC3
1	74HCT08D	74HCT08D	IC10
1	74HCT74D	74HCT74D	IC9
1	74HCT86D	74HCT86D	IC8
3	74HCT166D	74HCT166D	IC4, IC5, IC6
4	74HCT541DW	74HCT541DW	IC11, IC12, IC13, IC14
1	74HCT688DW	74HCT688DW	IC7
1	82	R-EU_0204/7	R24
4	82	R-EU_0207/12-OVAL	R1, R2, R5, R6
11	100n	C-EU025-024X044	C7, C8, C9, C10, C11, C12, C13, C14, C15, C17, C18
4	130	R-EU_0207/12-OVAL	R3, R4, R7, R8
1	470	R-EU_0204/7	R23
1	510	R-EU_0207/12-OVAL	R9
1	AES-OUT	NC3MD-H	XLR1
1	HFBR-5205	HFBR-52051X9	TRANSCEIVER1
1	LAD4613	SM-T4	TR1
1	MC100ELT22	MC100ELT22	IC1
1	MC100ELT23	MC100ELT23	IC2
1	P6KEXXA	P6KEXXA	D2
2	TPPAD1-13Y	TPPAD1-13Y	5V, GND

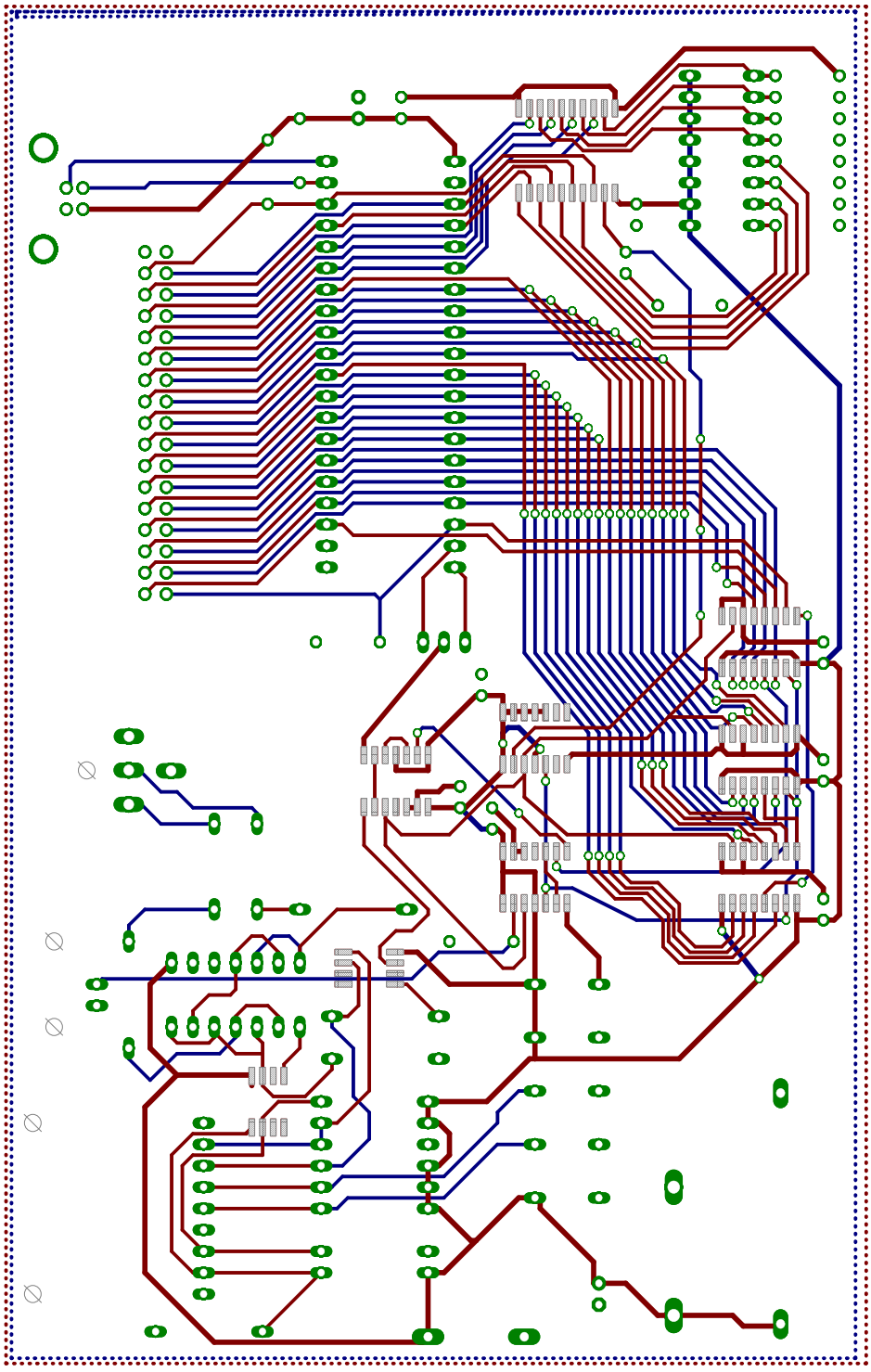

TITLE: tunnelzentrale_master
Document Number:
REV:
Date: 29.05.2010 23:30:45
Sheet: 1/4
ENC_OUT
TTL_IN
MC100ELT22
MC100ELT23
IC1A
IC2A
+5V
GND
R1
R2
R3
R4
R5
R6
R8
R9
82
130
510
C1
C2
C3
C4
C5
C6
0.1u
2.7u
10u
L1
L2
1u
TD
TDNOT
SD
RDNOT
RD
VEERX
VCCRX
VEETX
VCCTX
TX
RX
HFBR-5205
TRANSCEIVER1


TITLE: tunnelzentrale_master
Document Number:
REV:
Date: 29.05.2010 23:30:45
Sheet: 2/4
XLR1
AES-OUT
GND
GND
TR1
LAD4613
R10
2
R11
62
IC3D
74HC04N
IC3E
74HC04N
IC3B
74HC04N
IC3C
74HC04N
IC3A
74HC04N
IC3P
VCC
GND
+5V
GND
TTL_IN
P6KEXXA
D2
C16
47u
+5V
GND
5V
GND


TITLE: tunnelzentrale_master
Document Number:
REV:
Date: 29.05.2010 23:30:45
Sheet: 3/4
IOW_DATA:PREAMBLE[0..7],LEFT[0..7],RIGHT[0..7],DEVICE_ID[0..7]
74HCT166D
74HCT688DW
IO-WARRIOR40
MINI-USB_4P-85-32004-10X
VCC_USB
GND
+5V
CLK
SH/LD


TITLE: tunnelzentrale_master
Document Number:
REV:
Date: 29.05.2010 23:30:45
Sheet: 4/4
IOW_DATA:PREAMBLE[0..7],LEFT[0..7],RIGHT[0..7],DEVICE_ID[0..7]
DEVICE_ID0 SV2-25
DEVICE_ID1 SV2-26
DEVICE_ID2 SV2-27
DEVICE_ID3 SV2-28
DEVICE_ID4 SV2-29
DEVICE_ID5 SV2-30
DEVICE_ID6 SV2-31
DEVICE_ID7 SV2-32
SV2-33
SV2-34
GND
RIGHT0 SV2-17
RIGHT1 SV2-18
RIGHT2 SV2-19
RIGHT3 SV2-20
RIGHT4 SV2-21
RIGHT5 SV2-22
RIGHT6 SV2-23
RIGHT7 SV2-24
LEFT0 SV2-9
LEFT1 SV2-10
LEFT2 SV2-11
LEFT3 SV2-12
LEFT4 SV2-13
LEFT5 SV2-14
LEFT6 SV2-15
LEFT7 SV2-16
PREAMBLE0 SV2-1
PREAMBLE1 SV2-2
PREAMBLE2 SV2-3
PREAMBLE3 SV2-4
PREAMBLE4 SV2-5
PREAMBLE5 SV2-6
PREAMBLE6 SV2-7
PREAMBLE7 SV2-8
ENC_OUT
IC8C
74HCT86D
IC8B
74HCT86D
CLK
IC8A
74HCT86D
IC10C
74HCT08D
QH
IC10B
74HCT08D
SH/LD*
IC9A
PRE
D
CLK
CLR
Q
74HCT74D
EQUAL*
CLK
+5V
GND
IC10A
74HCT08D
R24
82
BNC1
IC9B
74HCT74D
IC8D
74HCT86D
IC10D
74HCT08D
C13
100n
IC9P
C15
100n
IC8P
C14
100n
IC10P
VCC
GND

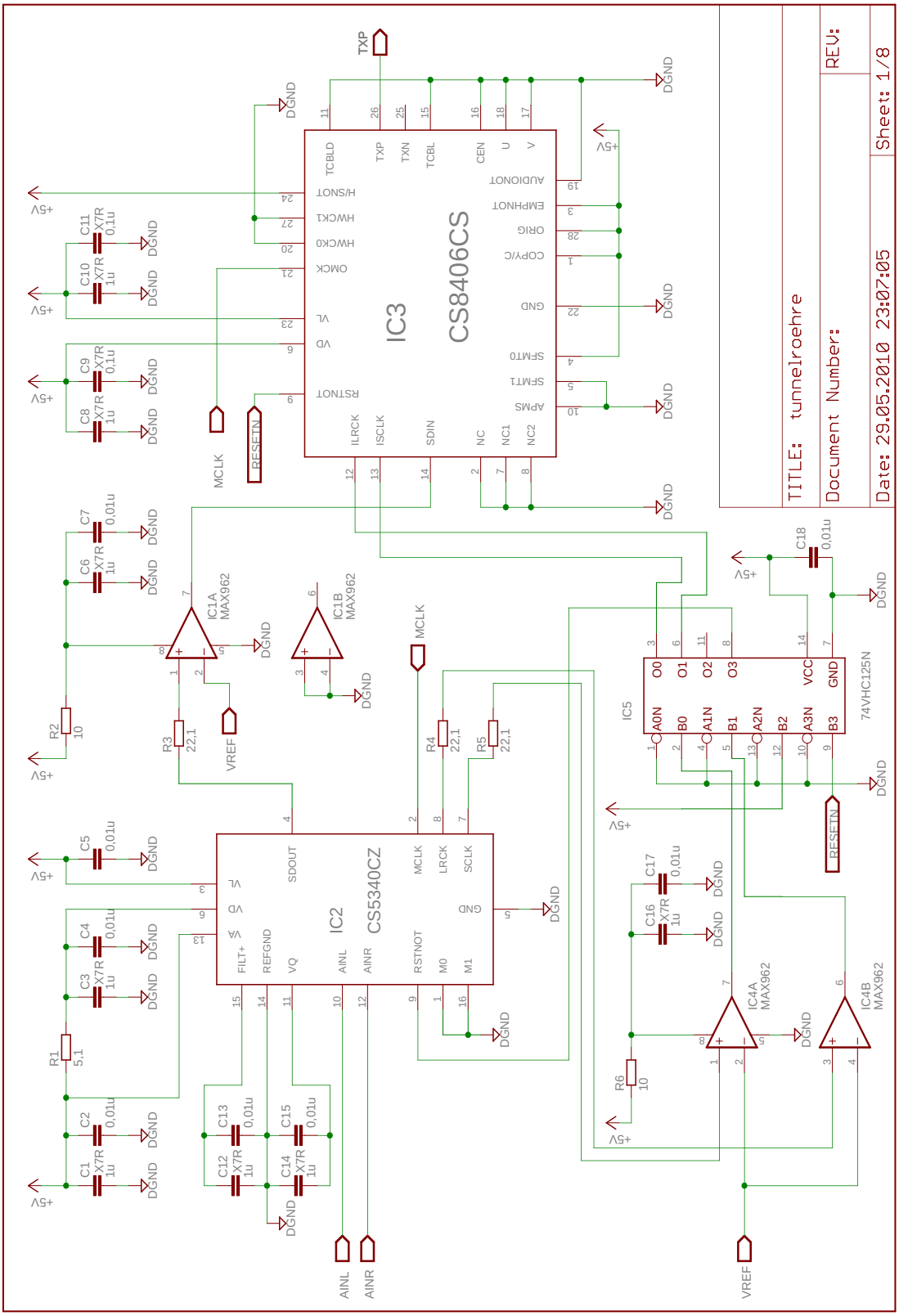

X1
MINI-USB_4P-85-32004-10X
SV2
34
D1
HD-WARRIOR40
R12
R13
C7
10u
C8
100n
IC7
74HCT688DW
C12
100n
LED1
R23
S1
ON
IC4
74HCT166D
C9
100n
IC6
74HCT166D
C11
100n
IC5
74HCT166D
C10
100n
6MHz
Q1
R14
C13
100n
IC9
74HCT4D
IC8
74HCT86D
100n
C15
100n
C14
IC10
74HCT8D
AES-OUT
XLR1
TR1
LAD4613
R11
IC3
74HC04N
R29
R10
IC1
MC100ELT22
R24
R3
51R
R9
IC2
MC100ELT22
R6
R2
R4
R1
L1
L2
R8
R5
R7
HFBR-5205
TRANSCEIVER1
C1
0.1u
C2
0.1u
C3
0.1u
C4
0.1u
C6
0.1u
10u
C5
D2
P6KE
5V
GND
47u
C16

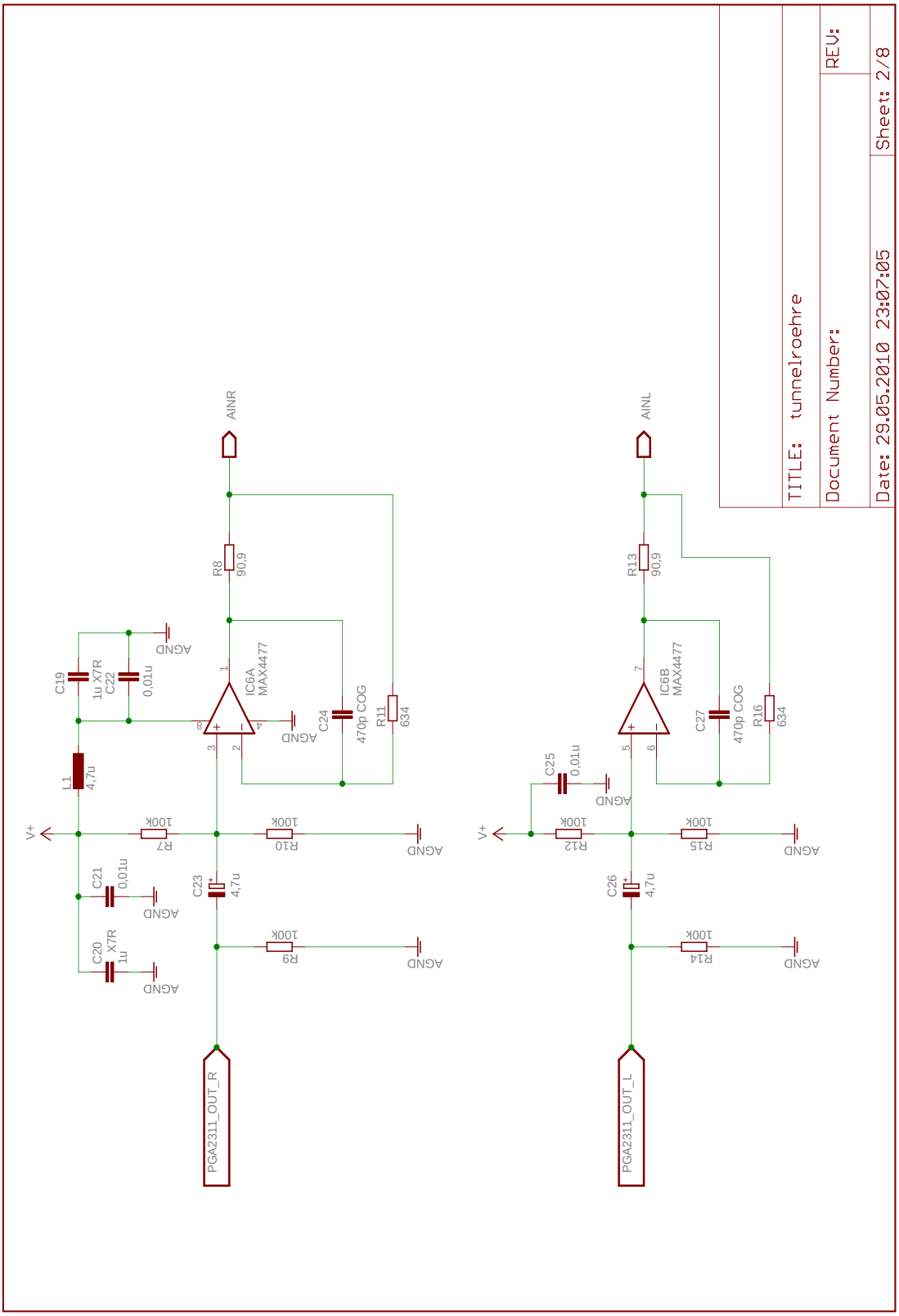


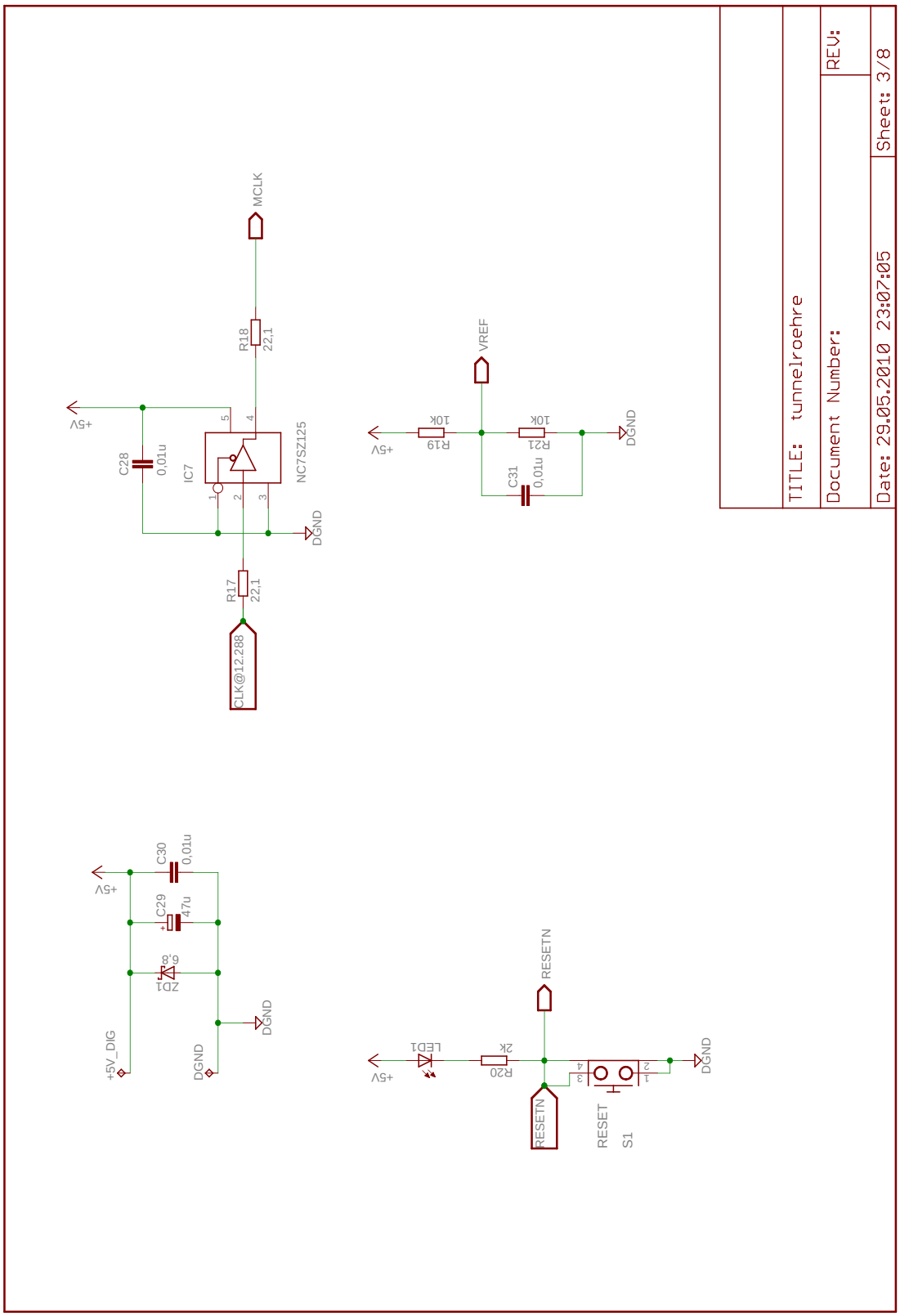
System: TUNNELZENTRALE_MASTER; Platinendimensionen: 160x100 mm

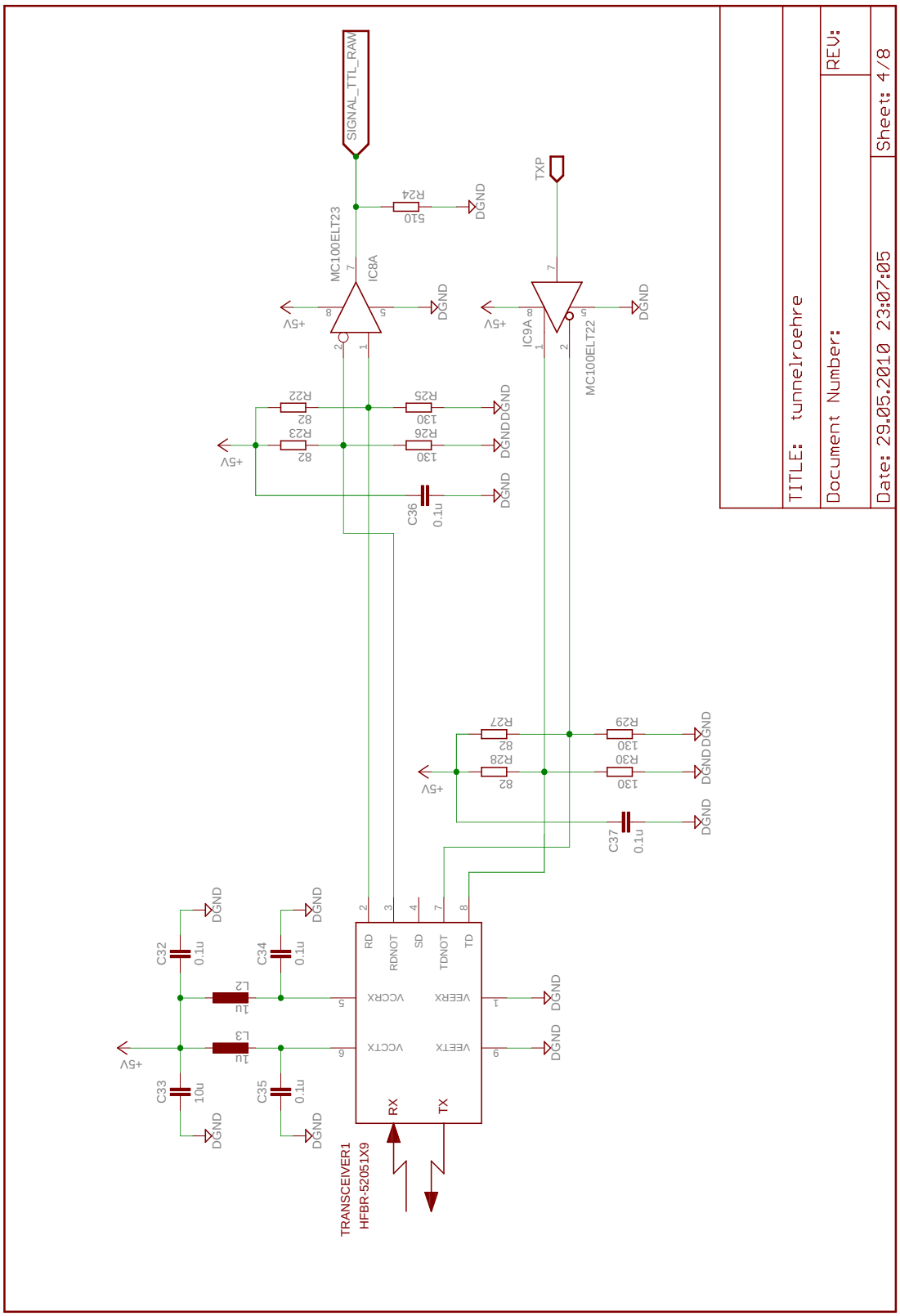
Bauteilliste:

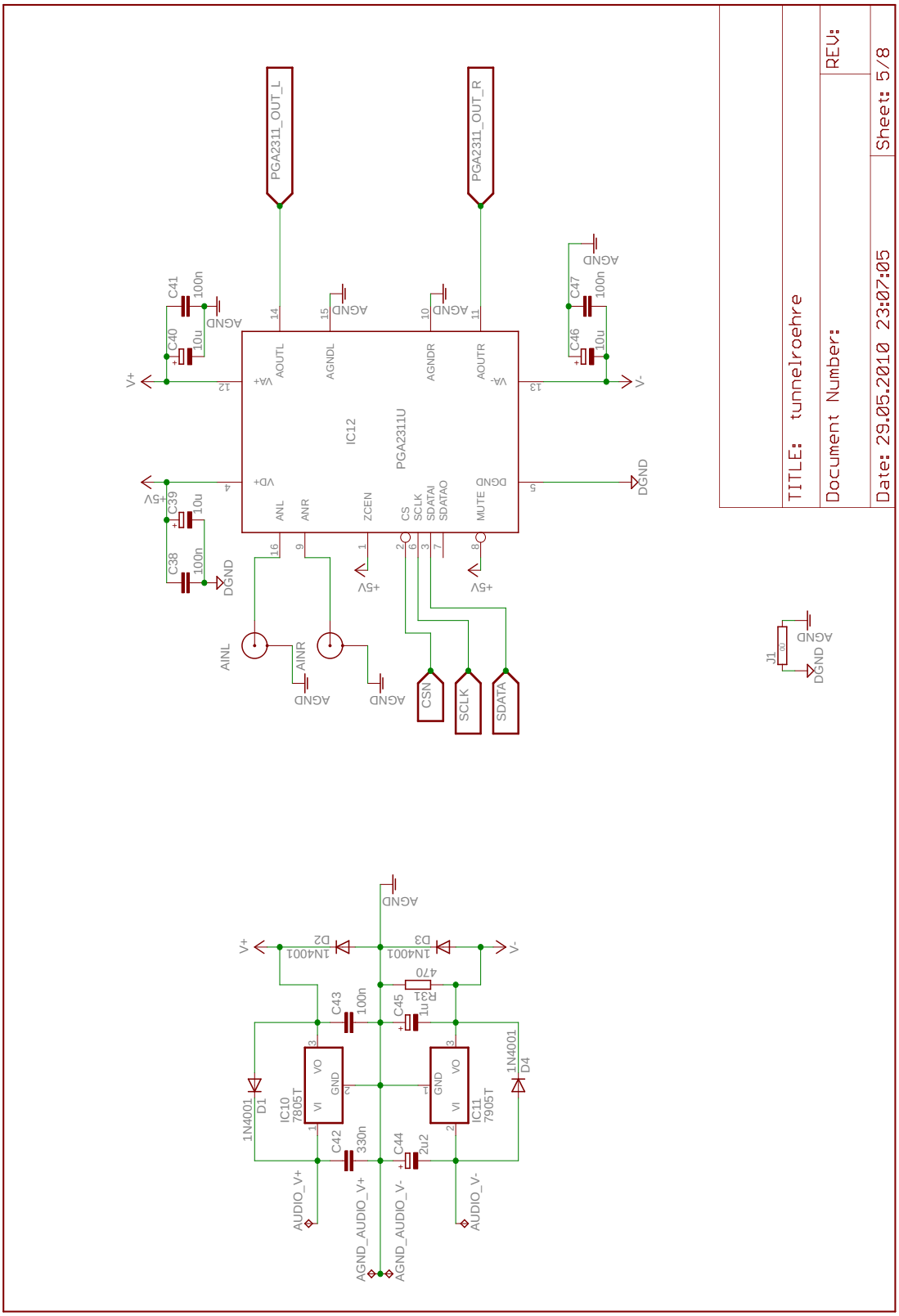
Menge	Wert	Device	Bauteile
1		BNC90	BNC1
1		DIP08YL	S1
1		LED5MM	LED1
1		ML34E	SV2
5	0.1u	C-EU075-032X103-OVAL	C1, C2, C3, C4, C6
8	1k8	R-EU_0204/7	R15, R16, R17, R18, R19, R20, R21, R22
2	1u	L-EU0207/12-OVAL	L1, L2
1	2	R-EU_0207/12-OVAL	R10
1	6MHz	CSTLS_G*	Q1
1	7k5	R-EU_0204/7	R12
1	10u	C-EU275-134X316-OVAL	C5
1	10u	CPOL-EUE2.5-5	C7
1	47u	CPOL-EUE2.5-7	C16
1	62	R-EU_0207/12-OVAL	R11
1	74HC04N	74AC04N	IC3
1	74HCT08D	74HCT08D	IC10
1	74HCT74D	74HCT74D	IC9
1	74HCT86D	74HCT86D	IC8
3	74HCT166D	74HCT166D	IC4, IC5, IC6
1	74HCT688DW	74HCT688DW	IC7
1	82	R-EU_0204/7	R24
4	82	R-EU_0207/12-OVAL	R1, R2, R5, R6
2	100k	R-EU_0204/7	R13, R14
8	100n	C-EU025-024X044	C8, C9, C10, C11, C12, C13, C14, C15
4	130	R-EU_0207/12-OVAL	R3, R4, R7, R8
1	470	R-EU_0204/7	R23
1	510	R-EU_0207/12-OVAL	R9
1	AES-OUT	NC3MD-H	XLR1
1	HFBR-5205	HFBR-52051X9	TRANSCEIVER1
1	IO-WARRIOR40	IO-WARRIOR40	D1
1	LAD4613	SM-T4	TR1
1	MC100ELT22	MC100ELT22	IC1
1	MC100ELT23	MC100ELT23	IC2
1	MINI-USB_4P-85-32004-10X	MINI-USB_4P-85-32004-10X	X1
1	P6KEXXA	P6KEXXA	D2
2	TPPAD1-13Y	TPPAD1-13Y	5V, GND

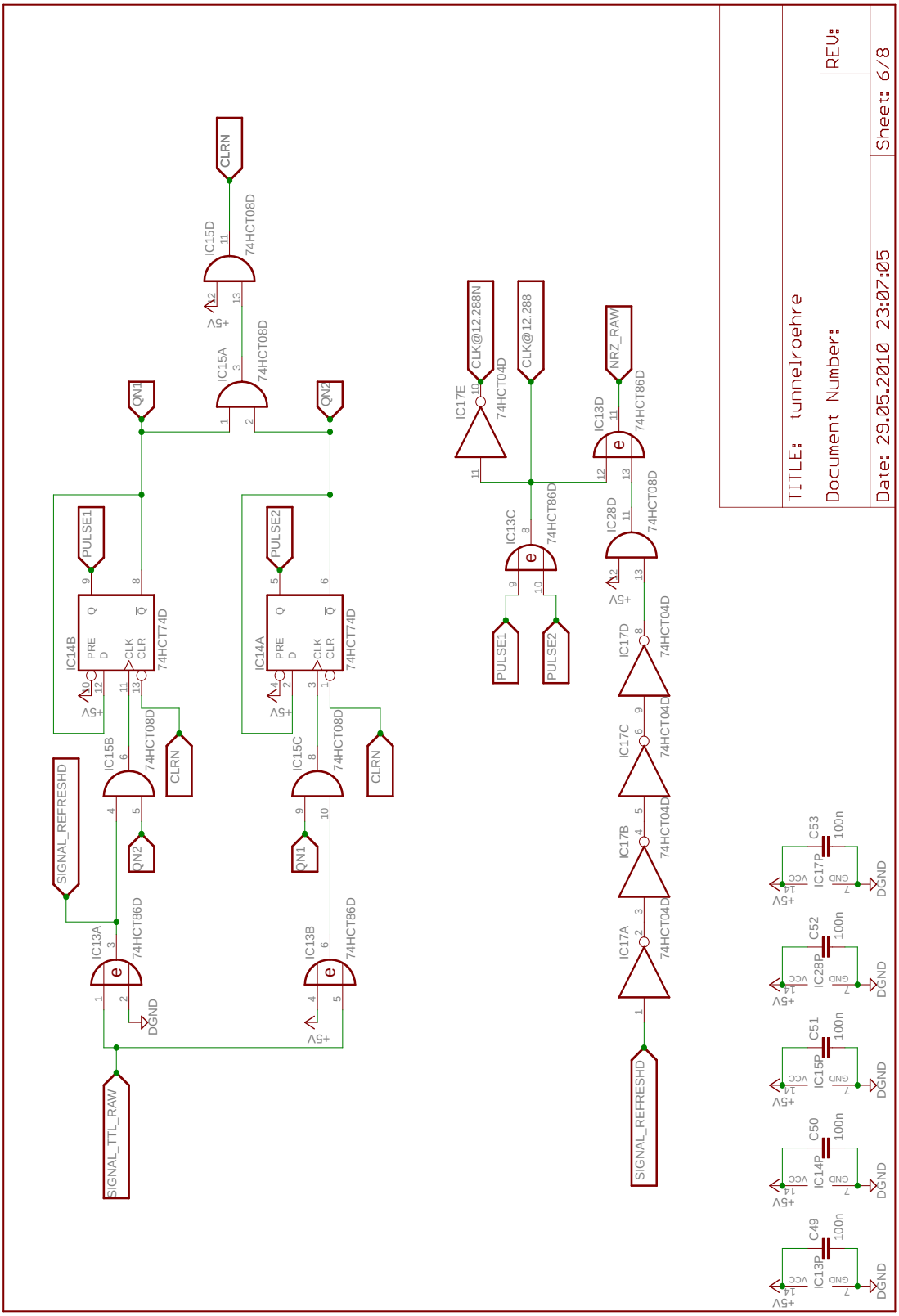

TITLE: tunnelroehre
Document Number:
REV:
Date: 29.05.2010 23:07:05
Sheet: 1/8
IC3
CS8406CS
IC2
CS5340CZ
IC5
74VHC125N
IC1A
MAX962
IC1B
MAX962
IC4A
MAX962
IC4B
MAX962
TXP
MCLK
RESETN
VREF
AINL
AINR
DGND
+5V

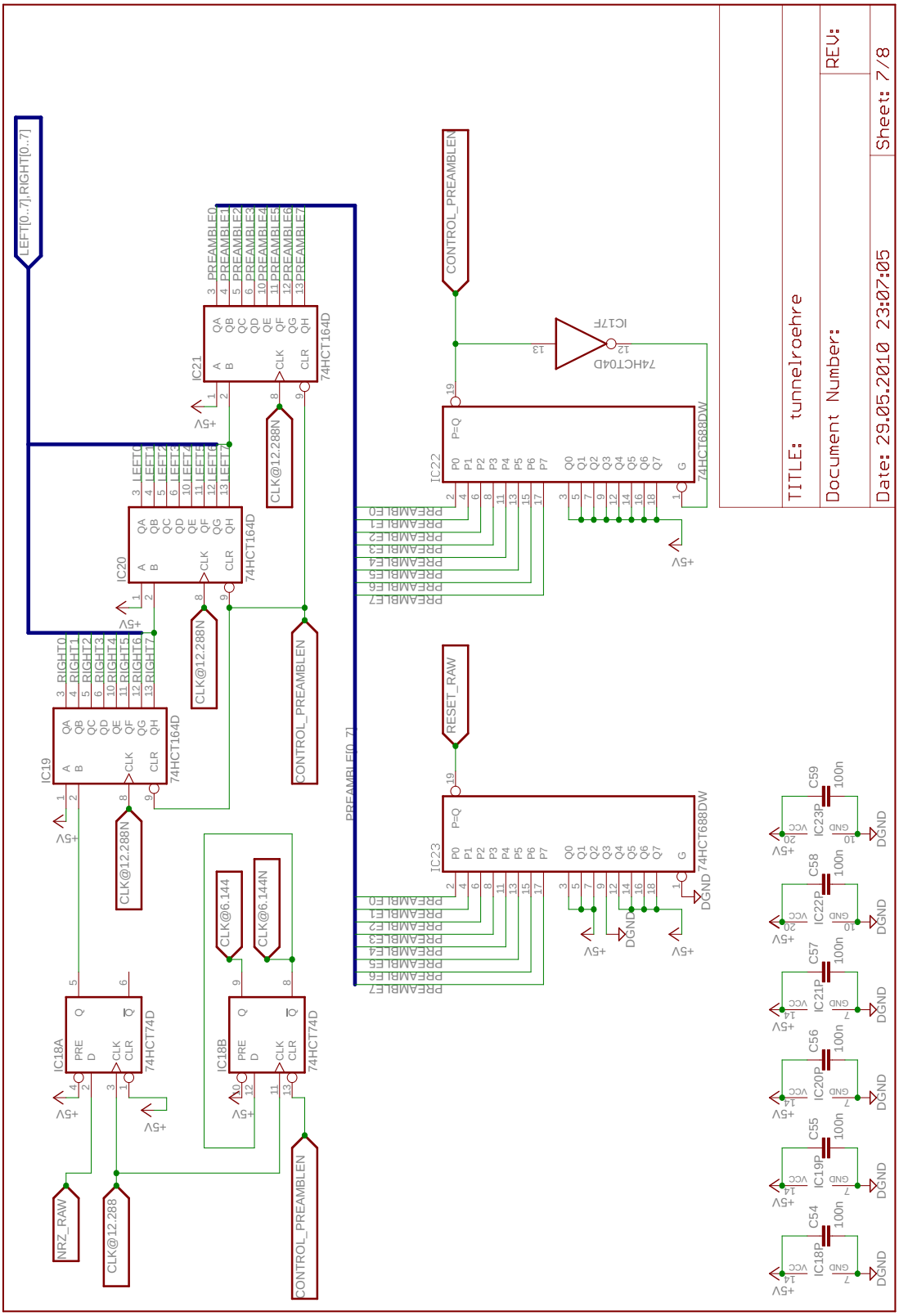

TITLE: tunnelroehre
Document Number:
REV:
Date: 29.05.2010 23:07:05
Sheet: 2/8
AINR
AINL
R8
90,9
R13
90,9
IC6A
MAX4477
IC6B
MAX4477
C19
1u X7R
C22
0,01u
C24
470p COG
R11
634
C27
470p COG
R16
634
C25
0,01u
L1
4,7u
R7
100K
R10
100K
R12
100K
R15
100K
C21
0,01u
C23
4,7u
C26
4,7u
C20
X7R
1u
R9
100K
R14
100K
AGND
V+
PGA2311_OUT_R
PGA2311_OUT_L

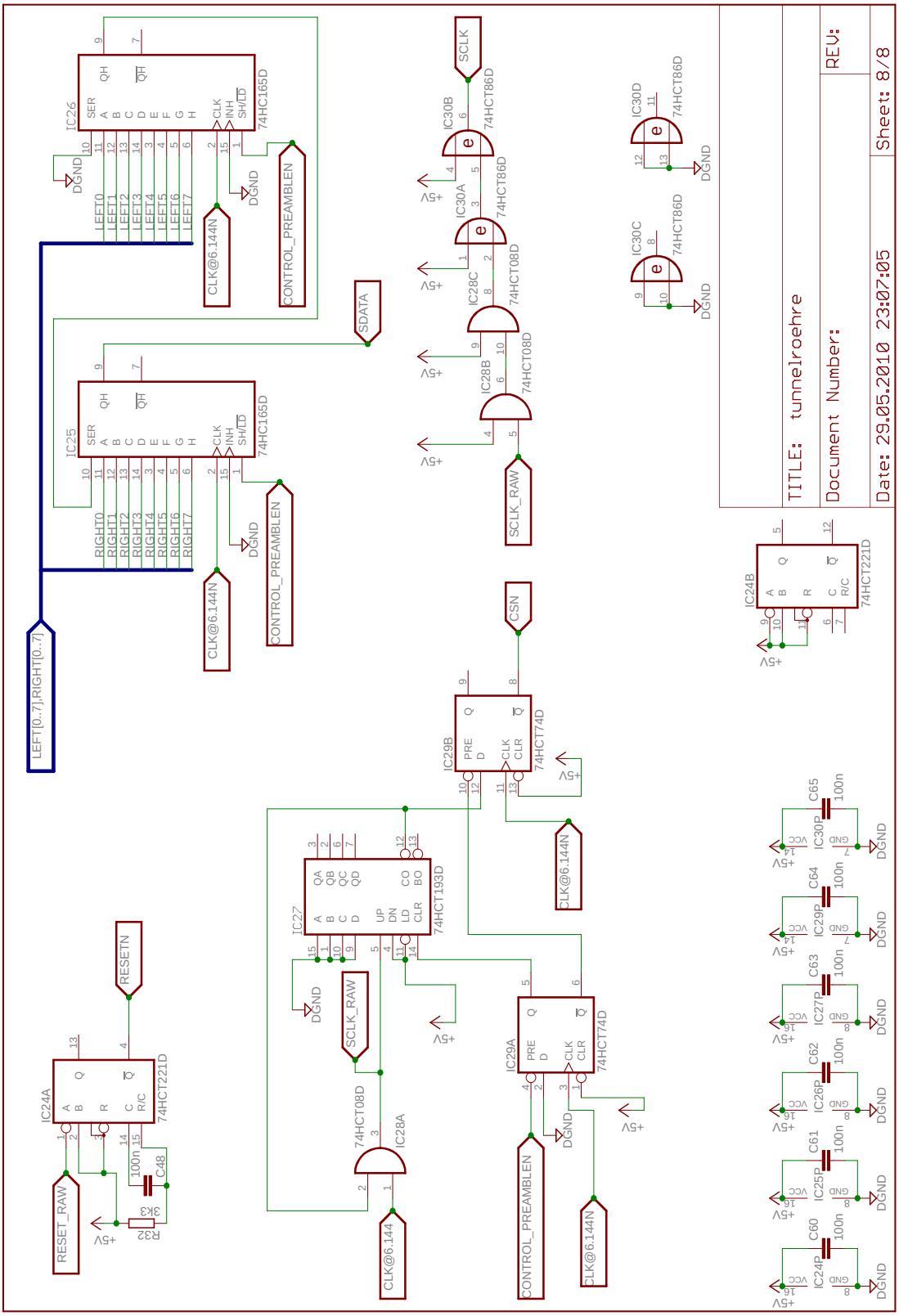

MCLK
R18
22,1
+5V
C28
0,01u
IC7
NC7SZ125
DGND
R17
22,1
CLK@12.288
VREF
R19
10K
R21
10K
C31
0,01u
C30
0,01u
C29
47u
ZD1
6,8
+5V_DIG
DGND
RESETN
LED1
R20
2K
RESET
S1
TITLE: tunnelroehre
Document Number:
REV:
Date: 29.05.2010 23:07:05
Sheet: 3/8

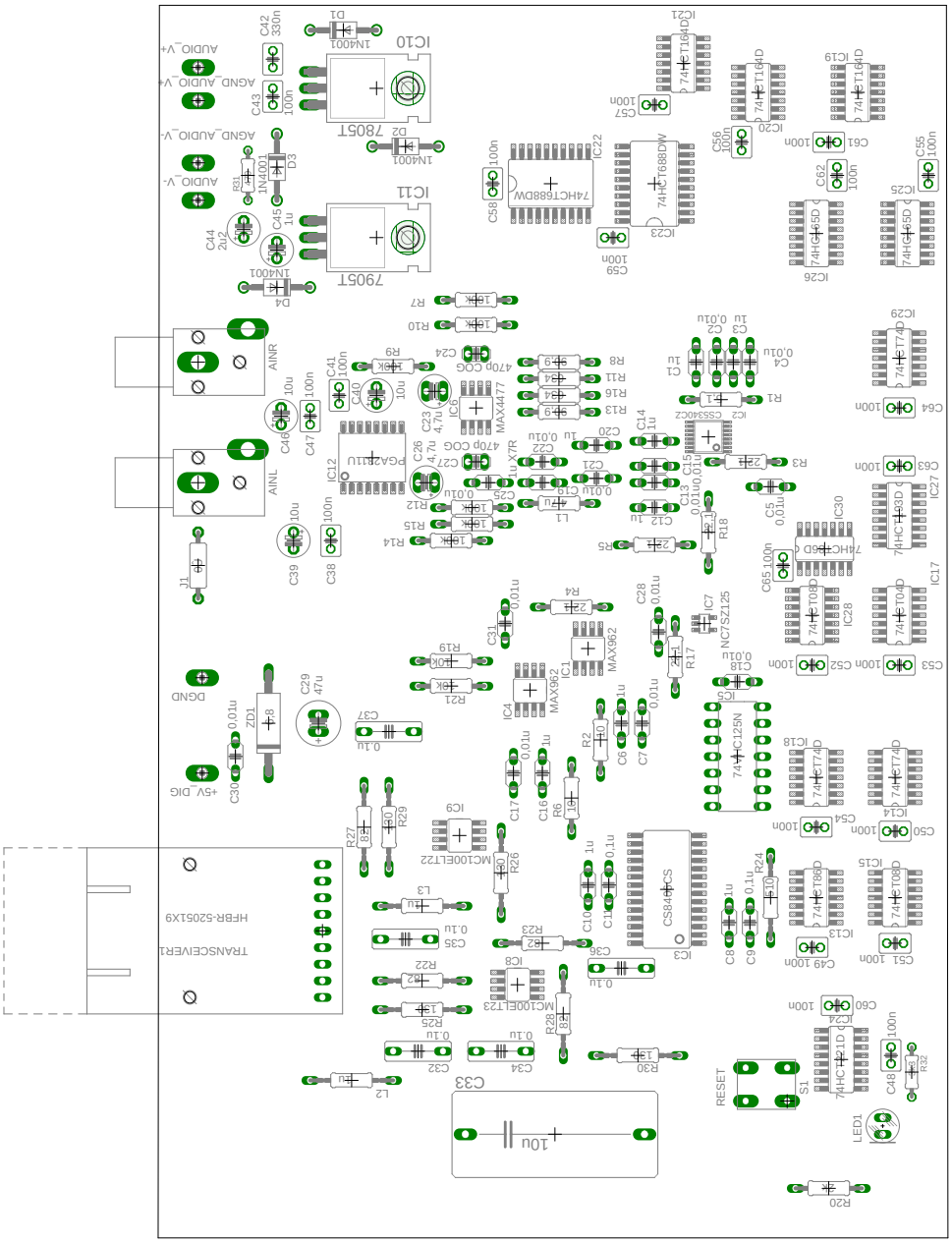

SIGNAL_TTL_RAW
TXP
R24
510
DGND
MC100ELT23
IC8A
IC9A
MC100ELT22
+5V
R22
82
R25
130
R23
82
R26
130
C36
0.1u
R27
82
R29
130
R28
82
R30
130
C37
0.1u
C32
0.1u
C34
0.1u
C33
10u
C35
0.1u
L2
1u
L3
1u
RD
RDNOT
SD
TDNOT
TD
VCCRX
VEERX
VCCTX
VEETX
RX
TX
TRANSCEIVER1
HFBR-52051X9
TITLE: tunnelroehre
Document Number:
REV:
Date: 29.05.2010 23:07:05
Sheet: 4/8

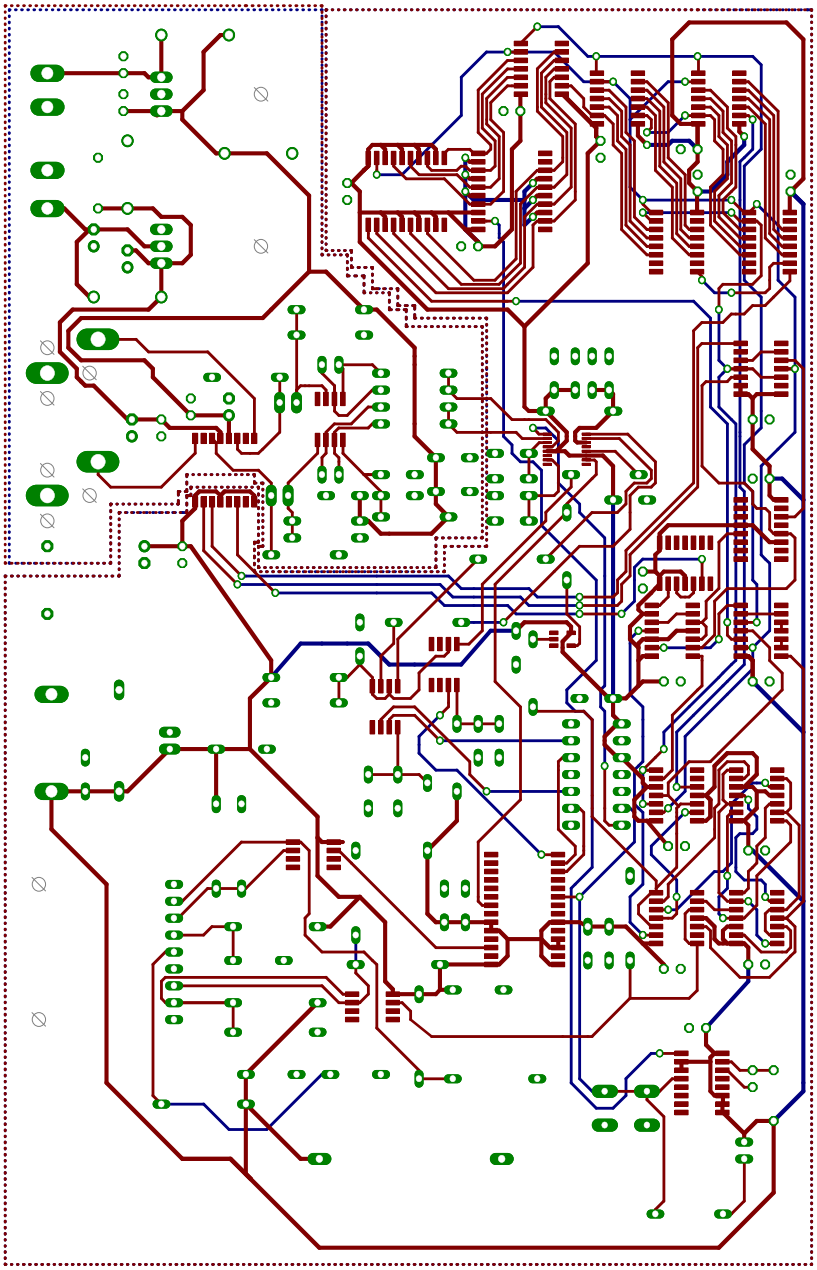

PGA2311_OUT_L
PGA2311_OUT_R
IC12
PGA2311U
AOUTL
AGNDL
AGNDR
AOUTR
VA+
VA-
VD+
DGND
ANL
ANR
ZCEN
CS
SCLK
SDATAI
SDATAO
MUTE
C41
100n
C40
10u
C47
100n
C46
10u
C39
10u
C38
100n
V+
V-
+5V
AGND
DGND
AINL
AINR
CSN
SCLK
SDATA
J1
D1
D2
D3
D4
1N4001
IC10
7805T
IC11
7905T
VI
VO
GND
C42
330n
C43
100n
C44
2u2
C45
1u
R31
470
AUDIO_V+
AUDIO_V-
AGND_AUDIO_V+
AGND_AUDIO_V-
TITLE: tunnelroehre
Document Number:
REV:
Date: 29.05.2010 23:07:05
Sheet: 5/8


TITLE: tunnelroehre
Document Number:
REV:
Date: 29.05.2010 23:07:05
Sheet: 6/8
SIGNAL_REFRESHD
SIGNAL_TTL_RAW
PULSE1
PULSE2
QN1
QN2
CLRN
CLK@12.288N
CLK@12.288
NRZ_RAW
IC13A
IC13B
IC13C
IC13D
IC14A
IC14B
IC15A
IC15B
IC15C
IC15D
IC17A
IC17B
IC17C
IC17D
IC17E
IC28D
74HCT86D
74HCT774D
74HCT08D
74HCT04D
DGND
+5V
C49
C50
C51
C52
C53
100n


LEFT[0..7],RIGHT[0..7]
PREAMBLE0_7
CONTROL_PREAMBLEN
RESET_RAW
NRZ_RAW
CLK@12.288
CLK@12.288N
CLK@6.144
CLK@6.144N
IC18A
IC18B
IC19
IC20
IC21
IC22
IC23
74HCT74D
74HCT164D
74HCT688DW
74HCT04D
IC17F
TITLE: tunnelroehre
Document Number:
REV:
Date: 29.05.2010 23:07:05
Sheet: 7/8


TITLE: tunnelroehre
Document Number:
REV:
Date: 29.05.2010 23:07:05
Sheet: 8/8
LEFT[0..7],RIGHT[0..7]
SDATA
SCLK
CSN
SCLK_RAW
RESETN
RESET_RAW
CONTROL_PREAMBLEN
CLK@6.144
CLK@6.144N
74HC165D
74HCT08D
74HCT74D
74HCT86D
74HCT193D
74HCT221D
74HCT08D





System: TUNNELROEHRE; Platinendimensionen: 190x120 mm

Bauteilliste:

Menge	Wert	Device	Bauteile
1		0R	J1
2		CINCHWBTOR	AINL, AINR
1		LED5MM	LED1
14	0,01u	C-EU050-024X044-OVAL	C2, C4, C5, C7, C13, C15, C17, C18, C21, C22, C25, C28, C30, C31
2	0,1u	C-EU050-024X044-OVAL	C9, C11
5	0.1u	C-EU075-032X103-OVAL	C32, C34, C35, C36, C37
4	1N4001	1N4004	D1, D2, D3, D4
9	1u	C-EU050-024X044-OVAL	C1, C3, C6, C8, C10, C12, C14, C16, C20
1	1u	CPOL-EUE2.5-5	C45
2	1u	L-EU0207/12-OVAL	L2, L3
1	1u	X7R C-EU050-024X044-OVAL	C19
1	2k	R-EU_0207/10	R20
1	2u2	CPOL-EUE2.5-5	C44
1	3k3	R-EU_0204/7	R32
2	4,7u	CPOL-EUE2.5-5OVAL	C23, C26
1	4,7u	L-EU0207/10	L1
1	5,1	R-EU_0207/10	R1
1	6,8	P6KEXXDO15	ZD1
2	10	R-EU_0207/10	R2, R6
2	10k	R-EU_0207/10	R19, R21
1	10u	C-EU275-134X316-OVAL	C33
3	10u	CPOL-EUE2.5-5	C39, C40, C46
5	22,1	R-EU_0207/10	R3, R4, R5, R17, R18
1	47u	CPOL-EUE2.5-7OVAL	C29
2	74HC165D	74HC165D	IC25 (TI), IC26 (TI)
1	74HCT04D	74HCT04D	IC17 (TI)
2	74HCT08D	74HCT08D	IC15 (TI), IC28 (TI)
3	74HCT74D	74HCT74D	IC14 (TI), IC18, IC29
2	74HCT86D	74HCT86D	IC13 (TI), IC30
3	74HCT164D	74HCT164D	IC19, IC20, IC21
1	74HCT193D	74HCT192D	IC27
1	74HCT221D	74HCT221D	IC24
2	74HCT688DW	74HCT688DW	IC22, IC23
1	74VHC125N	74VHC125N	IC5
4	82	R-EU_0207/12-OVAL	R22, R23, R27, R28
2	90,9	R-EU_0207/10	R8, R13
6	100k	R-EU_0207/10	R7, R9, R10, R12, R14, R15

22	100n	C-EU025-030X050	C38, C41, C43, C47, C48, C49, C50, C51, C52, C53, C54, C55, C56, C57, C58, C59, C60, C61, C62, C63, C64, C65
4	130	R-EU_0207/12-OVAL	R25, R26, R29, R30
1	330n	C-EU025-030X050	C42
1	470	R-EU_0204/7	R31
2	470p	COG C-EU025-024X044	C24, C27
1	510	R-EU_0207/12-OVAL	R24
2	634	R-EU_0207/10	R11, R16
1	7805T	7805T	IC10
1	7905T	7905T	IC11
1	CS5340CZ	CS5340CZ	IC2
1	CS8406CS	CS8406CS	IC3
1	HFBR-52051X9	HFBR-52051X9	TRANSCEIVER1
2	MAX962	MAX962	IC1, IC4
1	MAX4477	MAX4477SO08	IC6
1	MC100ELT22	MC100ELT22	IC9
1	MC100ELT23	MC100ELT23	IC8
1	NC7SZ125	NC7SZ125SOT23-5	IC7
1	PGA2311U	PGA2311U	IC12
1	S1	RESET	RESET
6	TPPAD1-17Y	TPPAD1-17Y	+5V_DIG, AGND_AUDIO_V+, AGND_AUDIO_V-, AUDIO_V+, AUDIO_V-, DGND

Anhang B: Header und Beschreibungen der Funktionen der Software

defines.h:

```
#include <string>

const std::string DEFAULT_LOUDNESS_PARAMETER="192"; // as seen in function "write_default_array(...)"

const int LOUDNESS_MAX=255; //      --\  for possible values see
const int LOUDNESS_MIN=0;   //      --/  the pga3211 datasheet

const int CHANNELS=6; // number of audio channels to be operated by akut_control software

const int STORED_PARAMETERS=4;    //(1)channel number
                                  //(2)device_id
                                  //(3)byte_alignment
                                  //(4)loudness_parameter

//=============================================================================

const std::string DEVICE_ID_FOR_SHIFTING_OP="11111111";

// bitstream used to set shiftregisters to shifting mode (not available to address devices (device_id) any more)

//=============================================================================

const std::string RESET_PREAMBLE="11110111";

// as seen in function "reset(...)":
//
// RESET COMMAND: "11110111  +  00000000 + 00000000"
//                  \______/    \_____________________/
//            RESET_PREAMBLE    DEFAULT '0' to not accidently trigger
//                              CONTROL COMMAND or SYNC COMMAND


const std::string CONTROL_PREAMBLE="11111111";

// as seen in functions "absolute(...)", "increment(...)", decrement(...):
//
// CONTROL COMMAND: "11111111    +    xxxxxxxx    +    xxxxxxxx"
//                    \______/       \______/        \______/
//          CONTROL_PREAMBLE    LOUDNESS PARAMETER    LOUDNESS PARAMETER
//                              LEFT CHANNEL          RIGHT CHANNEL
//
// decoder's shiftregisters get cleared (by hardware) afterwards to not accidently trigger other commands


const std::string SYNC_PREAMBLE="00011000";

// as seen in function "sync(...)":
//
// SYNC COMMAND: "00011000  +  00000000 + 00000000"
//                 \______/    \_____________________/
//           SYNC_PREAMBLE    DEFAULT '0' to not accidently trigger
//                            CONTROL COMMAND or SYNC COMMAND
//
// needed to ensure clock recovery at center of manchester code bitcell

//=============================================================================
```

akut_control.h:

```
#ifndef __akut_control__
#define __akut_control__

#include <iostream>
#include <sstream>
#include <fstream>
#include <string>
#include <vector>
#include <bitset>
#include <windows.h> // only needed to resolve: #defines (ULONG, LONG, USHORT,WORD,...) declared in iowkit.h
#include "defines.h"
#include "lut_class.h"
#include "iowkit.h"
#include <time.h>

/*=========================================================================
INPUT:              int time_ms: desired noop duration in milliseconds

DESCRIPTION:        function executes a do-nothing-loop for the specified time

RETURN VALUES:      none
*/
void noop(int time_ms);


/*=========================================================================
INPUT:              string _string: should contain unsigned integer information

DESCRIPTION:        function takes input string and converts it to corresponding integer value

RETURN VALUES:      conversion successful ==> integer value of input string as type integer
                    conversion fail       ==> -1
*/
int string_to_int (std::string _string);


/*=========================================================================
INPUT:              string _string: should contain decimal format

DESCRIPTION:        function takes input numerical string, and returns incremented numerical string

RETURN VALUES:      incremented numerical input string
*/
std::string numerical_string_increment (std::string _string);


/*=========================================================================
INPUT:              string _string: should contain decimal format

DESCRIPTION:        function takes input numerical string, and returns decremented numerical string

RETURN VALUES:      decremented numerical input string
*/
std::string numerical_string_decrement (std::string _string);
```

```
/*=========================================================================
INPUT:              string _string: should contain binary string (max. 8 bit)

DESCRIPTION:        function takes input string     and converts to corresponding integer value

RETURN VALUES:  conversion successful ==> unsigned integer value of input string as type integer
                    conversion fail         ==> -1
*/
int bin2dec(std::string _string);


/*=========================================================================
INPUT:              unsigned int decimal: should contain unsigned int (max. value 255)

DESCRIPTION:        function takes input and converts to corresponding bitstream

RETURN VALUES:  conversion successful ==> 8 digit binary string of input unsigned integer value
                    conversion fail         ==> message: "not a valid 8bit unsigned int"
*/
std::string dec2bin(unsigned int decimal);


/*=========================================================================
INPUT:              string _byte: valid content is either "left" or "right"

DESCRIPTION:        function takes input string     and returns corresponding BYTE for IO-WARRIOR write operation

RETURN VALUES:  valid content   ==> corresponding BYTE for IO-WARRIOR write operation
                    illegal content ==> negative BYTE value (leads to exception @ IO-WARRIOR write operation)
*/
int resolve_byte_alignment(std::string _byte);


/*=========================================================================
INPUT:              IOWKIT40_IO_REPORT report: structure needed for any communication with the IO-WARRIOR
                                                (as specified in iowkit.h)

                    IOWKIT_HANDLE& __iowHandle: reference to IO-WARRIOR device pointer

DESCRIPTION:        function writes information of 'report' onto the 4 IO-WARRIOR ports

RETURN VALUES:  writing successful  ==> TRUE
                    writing failed      ==> FALSE
*/
bool write_to_iowarrior(IOWKIT40_IO_REPORT report, IOWKIT_HANDLE& __iowHandle);


/*=========================================================================
INPUT:              string range_argument:  valid input string is of format "range_begin-range_end"
                                            e.g. "0-3" for information on channels 0,1,2,3 or
                                                 "1-1" for channel 1 only

                    lut _lookuptable:       object of class lut including all information
                                            needed to control one entire akut system

DESCRIPTION:        function prints chosen range of channels of actual '_lookuptable'
                    and its parameters to screen

RETURN VALUES:  none
*/
void print_array(std::string range_argument, lut _lookuptable);
```

```
/*=========================================================================
INPUT:              lut _lookuptable:       object of class lut including all information
                                            needed to control one entire akut system

DESCRIPTION:        function tests integrity of '_lookuptable'

RETURN VALUES:      array okay              ==> TRUE
                    array contains errors   ==> FALSE
*/
bool test_array(lut _lookuptable);


/*=========================================================================
INPUT:              lut& _lookuptable:      object of class lut including all information
                                            needed to control one entire akut system

DESCRIPTION:        function sets up '_lookuptable' with default 'channel', 'device_id',
                    'byte_alignment', and 'DEFAULT_LOUDNESS_PARAMETER' parameters for akut-control purposes

                    e.g.  0,00000000,left,'DEFAULT_LOUDNESS_PARAMETER'
                          1,00000000,right,'DEFAULT_LOUDNESS_PARAMETER'
                          2,00000001,left,'DEFAULT_LOUDNESS_PARAMETER'
                          ...
                          ...
                          ...
                          no_of_channels,...

RETURN VALUES:      none
*/
void write_default_array(lut& _lookuptable);


/*=========================================================================
INPUT:              int _channel_number:    channel containing requested parameter

                    lut _lookuptable:       object of class lut including all information
                                            needed to control one entire akut system


DESCRIPTION:        function returns ('_channel_number')s corresponding 'loudness_parameter'

RETURN VALUES:      'loudness_parameter' of aforementioned channel
*/
std::string channel_to_parameter(int _channel_number, lut _lookuptable);


/*=========================================================================
INPUT:              int _channel_number:    channel containing requested parameter

                    lut _lookuptable:       object of class lut including all information
                                            needed to control one entire akut system


DESCRIPTION:        function returns ('_channel_number')s corresponding 'byte_alignment'
                    parameter ("left" or "right")


RETURN VALUES:      'byte_alignment' parameter of aforementioned channel
*/
std::string channel_to_byte_alignment(int _channel_number, lut _lookuptable);
```

```
/*=========================================================================
INPUT:              int _channel_number:   channel containing requested parameter

                    lut _lookuptable:      object of class lut including all information
                                           needed to control one entire akut system


DESCRIPTION:        function returns ('_channel_number')s corresponding 'device_id'

RETURN VALUES:      'device_id' of aforementioned channel
*/
std::string channel_to_device_id(int _channel_number, lut _lookuptable);


/*=========================================================================
INPUT:              string __byte_alignment:     'byte_aligment' information ("left" or "right")

DESCRIPTION:        function returns opposite of input 'byte_aligment'

RETURN VALUES:      conversion successful ==> opposite of input 'byte_aligment'
                    conversion fail       ==> "opposite is unknown"
*/
std::string opposite_of(std::string __byte_id);


/*=========================================================================
INPUT:              string _device_id:       contains 8 bit device number as binary statement

                    string _byte_alignment: byte alignment information ("left" or "right")

                    lut _lookuptable:       object of class lut including all information
                                            needed to control one entire akut system

DESCRIPTION:        function searches for second channel with same '_device_id' and diametrical '_byte_alignment'

RETURN VALUES:      finding successful  ==> number of associated channel in string format
                    finding fail        ==> string "NO CORRESPONDING CHANNEL FOUND"
*/
std::string find_other_channel_with_same_device_id(std::string _device_id, std::string _byte_id, lut _lookuptable);


/*=========================================================================
INPUT:              string __channel_range: valid input string is of format "range_begin-range_end"
                                             e.g. "0-3" for channels 0,1,2,3 or
                                                  "1-1" for channel 1 only

                    int& range_begin:       reference to int range_begin (which
                                            is needed in further loop declarations)

                    int& range_end:              reference to int range_end (which
                                                 is needed in further loop declarations)

DESCRIPTION:        function test input string '__channel_range' for its validity

RETURN VALUES:      none, but updated integers 'range_begin' and 'range_end' if testing successful
*/
void check_range(std::string __channel_range, int& range_begin, int& range_end);
```

```
/*=====================================================================
INPUT:              string range_argument:      valid input string is of format "range_begin-range_end"
                                                e.g. "0-3" for channels 0,1,2,3 or
                                                     "1-1" for channel 1 only

                    string loudness_parameter:  gain parameter in string container, to which channels
                                                named in 'range_argument' are being set

                    lut& _lookuptable:          object of class lut including all information
                                                needed to control one entire akut system

                    IOWKIT_HANDLE& _iowHandle:  reference to IO-WARRIOR device pointer

DESCRIPTION:        function sets channels referred to by 'range_argument' to new gain
                    factor ('loudness_parameter') and updates object '_lookuptable' if successfully
                    written to hardware

RETURN VALUES:  none
*/
void absolute (std::string range_argument, std::string loudness_parameter, lut& _lookuptable, IOWKIT_HANDLE& _io
~wHandle);


/*=====================================================================
INPUT:              string range_argument:      valid input string is of format "range_begin-range_end"
                                                e.g. "0-3" for channels 0,1,2,3 or
                                                     "1-1" for channel 1 only

                    lut& _lookuptable:          object of class lut including all information
                                                needed to control one entire akut system

                    IOWKIT_HANDLE& _iowHandle:  reference to IO-WARRIOR device pointer

DESCRIPTION:        function sets channels referred to by 'range_argument' +0,5dB and
                    updates object '_lookuptable' if successfully written to hardware

RETURN VALUES:  none
*/
void increment (std::string range_argument, lut& _lookuptable, IOWKIT_HANDLE& _iowHandle);


/*=====================================================================
INPUT:              string range_argument:      valid input string is of format "range_begin-range_end"
                                                e.g. "0-3" for channels 0,1,2,3 or
                                                     "1-1" for channel 1 only

                    lut& _lookuptable:          object of class lut including all information
                                                needed to control one entire akut system

                    IOWKIT_HANDLE& _iowHandle:  reference to IO-WARRIOR device pointer

DESCRIPTION:        function sets channels referred to by 'range_argument' -0,5dB and
                    updates object '_lookuptable' if successfully written to hardware

RETURN VALUES:  none
*/
void decrement (std::string range_argument, lut& _lookuptable, IOWKIT_HANDLE& _iowHandle);
```

```
/*=============================================================================
INPUT:              string range_argument:      valid input string is of format "range_begin-range_end"
                                                e.g. "0-3" for channels 0,1,2,3 or
                                                     "1-1" for channel 1 only

                    lut _lookuptable:           object of class lut including all information
                                                needed to control one entire akut system

                    IOWKIT_HANDLE& _iowHandle:  reference to IO-WARRIOR device pointer

DESCRIPTION:        function resets channels and channels with same 'device_id' referred to by 'range_argument'

RETURN VALUES:      none
*/
void reset(std::string range_argument, lut _lookuptable, IOWKIT_HANDLE& _iowHandle);


/*=============================================================================
INPUT:              string range_argument:      valid input string is of format "range_begin-range_end"
                                                e.g. "0-3" for information on channels 0,1,2,3 or
                                                     "1-1" for channel 1 only

                    lut _lookuptable:           object of class lut including all information
                                                needed to control one entire akut system

                    IOWKIT_HANDLE& _iowHandle:  reference to IO-WARRIOR device pointer

DESCRIPTION:        function syncronizes decoder part of channels (and channels with
                    same 'device_id') referred to by 'range_argument'

RETURN VALUES:      none
*/
void sync(std::string range_argument, lut _lookuptable, IOWKIT_HANDLE& _iowHandle);


/*=============================================================================
INPUT:              IOWKIT_HANDLE& _iowHandle:  reference to IO-WARRIOR device pointer

                    lut& _lookuptable:          object of class lut including all information
                                                needed to control one entire akut system

DESCRIPTION:        function initializes akut control hardware and sets up '_lookuptable' with default values

RETURN VALUES:      none
*/
void initialize_io_warrior (IOWKIT_HANDLE& iowHandle, lut& _lookuptable);
```

```
/*=========================================================================
INPUT:              lut& _lookuptable:          object of class lut including all information
                                                needed to control one entire akut system

                    string filename:            includes filename of saved akut control settings

                    IOWKIT_HANDLE& _iowHandle:  reference to IO-WARRIOR device pointer



DESCRIPTION:        function tests lookuptable stored in  file 'filename' (see function 'test_array'),
                    sets up akut control hardware to file settings (if file is valid),
                    and updates '_lookuptable' (if file is valid)

RETURN VALUES:      none
*/
void read_array_from_file(lut& _lookuptable, std::string _filename, IOWKIT_HANDLE& _iowHandle);


/*=========================================================================
INPUT:              lut _lookuptable:           object of class lut including all information
                                                needed to control one entire akut system

                    string filename:            includes filename to which akut control settings
                                                should be saved

DESCRIPTION:        function stores current lookuptable in a file named 'filename'

RETURN VALUES:      none
*/
void write_array_to_file(lut _lookuptable, std::string __filename);


/*=========================================================================
INPUT:              lut _lookuptable:           object of class lut including all information
                                                needed to control one entire akut system

                    string filename:            includes filename to be checked

DESCRIPTION:        function checks if file named 'filename' already exists and
                    executes function 'write_array_to_file(...)' if 'filename' not exists,
                    or otherwise handles overwrite dialogue

RETURN VALUES:      none
*/
void check_if_file_exists(lut _lookuptable, std::string _filename);
#endif
```

lut_class.h:

```
#ifndef __lut_class__
#define __lut_class__

#include <string>
#include <vector>

lut
class lut
    private:
        int no_of_channels;
        int no_of_stored_parameters;

        std::vector <std::vector <std::string> > matrix;
    public:
        /*=====================================================================
        INPUT:              int _no_of_channels:            number of channels to be controlled by akut control

                            int _no_of_stored_parameters:   number of stored parameters per channel

        DESCRIPTION:        default constuctor for object 'lut' to obtain <vector> made matrix
                            containing <string> containers

        RETURN VALUE:       none
        */
            lut(int _no_of_channels, int _no_of_stored_parameters);


        /*=====================================================================
        INPUT:              int channels:                   number of channels to be controlled by akut control

        DESCRIPTION:        set member variable 'no_of_channels' of object 'lut'

        RETURN VALUE:       none
        */
            void set_no_of_channels (int channels);


        /*=====================================================================
        INPUT:              int parameters:                 number of stored parameters per channel

        DESCRIPTION:        set member variable 'no_of_stored_parameters' of object 'lut'

        RETURN VALUE:       none
        */
            void set_no_of_stored_parameters (int parameters);


        /*=====================================================================
        INPUT:              none

        DESCRIPTION:        get member variable 'no_of_channels' of object 'lut'

        RETURN VALUE:       'number_of_channels' of object 'lut'
        */
            int get_no_of_channels (void);
```

```
    /*=========================================================================
    INPUT:              none

    DESCRIPTION:        get member variable 'no_of_stored_parameters' of object 'lut'

    RETURN VALUE:       'number_of_stored_parameters' of object 'lut'
    */
        int get_no_of_stored_parameters(void);


    /*=========================================================================
    INPUT:              unsigned in row:        row access parameter for matrix of type <string>

                        unsigned int column:    column access parameter for matrix of type <string>

    DESCRIPTION:        get matrix element @[row][column] of object 'lut'

    RETURN VALUE:       matrix element of object 'lut' in <string> container
    */
        std::string get_matrix_element(unsigned int row, unsigned int column);


    /*=========================================================================
    INPUT:              unsigned in row:        row access parameter for matrix of type <string>

                        unsigned int column:    column access parameter for matrix of type <string>

                        string data:            string to be written to matrix @[row][column]

    DESCRIPTION:        set matrix element of object 'lut'

    RETURN VALUE:       none
    */
        void set_matrix_element(unsigned int row, unsigned int column, std::string data);
};
#endif
```