Projekt Elektronik – Abschlußbericht

"USB-Mehrkanalmeßgerät"

(Gruppe: *ele21*)

Peter Kortmann <xarax@cs.tu-berlin.de> (Matr. 186855)

Christian Richter <richterc@cs.tu-berlin.de> (Matr. 192548)

Driton Emini <DritonEmini@yahoo.de> (Matr. 186664)

Shpend Mirta <shpendm@yahoo.de> (Matr. 177536)

Walid Tfayli <tfayli@gmx.net> (Matr. 189453)

Omar El-Mikati <OmarMikati@hotmail.com> (Matr. 203373)

Jörn Hohlwein <JoernHohlwein2@gmx.de> (Matr. 200487)

Helge Krambeck <HelgeKrambeck@t-online.de> (Matr. 197910)

Betreuung: **Berit Herrmann**

1. August 2003

Zusammenfassung

Ziel des Projektlabors Elektronik sind Ideenfindung, Planung, Entwurf und Realisierung, sowie Inbetriebnahme und Test eines elektronischen Geräts.
Damit sollen die in der Vorlesung “Analog- und Digitalelektronik” erworbenen theoretischen Kenntnisse in die Praxis umgesetzt werden. Das zu absolvierende Projekt soll also sowohl analoge als auch digitale Komponenten, sowie Analog-Digital-Umsetzer enthalten.
Bei der Ideenfindung wurden durch Brainstorming folgende Vorschläge aufgebracht:

- *MP3 Player*
- *Regelbare Spannungsversorgung*
- *Drahtloser Audioübertrager*
- *USB-Mehrkanalmeßgerät*
- *Batterieladegerät*

Der drahtlose Audioübertrager hätte große Schwierigkeiten im Platinenlayout nach sich gezogen (Hochfrequenzschaltung) so daß diese Idee leider aufgegeben werden mußte. Nach einer Abstimmung konnte sich die Gruppe schließlich auf das Mehrkanalmeßgerät einigen.

Inhaltsverzeichnis

1 Zielstellung **3**

1.1 Spezifikationen 3

1.2 Aufbau 4

2 Analog-Teil **6**

2.1 Entwurf in Eagle 6

2.2 Verstärkung 6

2.3 Offsetfehler 6

2.4 Wahl des Filterbausteins 7

2.5 Blockschaltbild der neuen Schaltung 7

2.6 Bauen und Testen 7

2.7 Relaisansteuerung 7

2.8 Sperrfrequenzeinstellung am Filter 8

2.9 Schutzschaltung 8

2.10 Fertigung der Platine 8

2.11 Eigenschaften der neuen Schaltung 8

2.12 Bauteilliste: 8

3 Analog-Digital-Umsetzer (ADC) **13**

3.1 ADC 13

3.2 Spannungsversorgung 14

3.3 Platine 16

4 USB-Interface **18**

4.1 Einführung 18

4.1.1 Allgemeines 18

4.1.2 Vorstellung des USB Moduls 18

4.1.3 Inbetriebnahme des USB Moduls 19

4.2 Microcontroller Programmierung 20

4.2.1 Wahl der Entwicklungsumgebung 20
4.2.2 Testprogramme in C++ 20
4.2.3 Entwicklung eines MC-Leseprogramms 21
4.2.4 Entwicklung eines MC-Schreibprogramms 22
4.2.5 Geschwindigkeitsmessung Atmel nach PC 23
4.3 Entwicklung des PC Programms mit Borland Delphi 5 24
4.3.1 Windows XP Treiber 24
4.3.2 Entwicklung der PC-Messkartensoftware 24
4.3.3 Testen der PC-Meßkartensoftware mit MC 26

5 Mikrocontroller (Digitalteil) 30
5.1 Schaltung 30
5.1.1 Der Atmel Mikrocontroller 30
5.1.2 Das Bussystem 30
5.1.3 Das LC-Display 32
5.1.4 Die Taster 33
5.1.5 Hauptplatine 33
5.1.6 Benötigte Bauelemente 34
5.2 Schnittstellen 35
5.2.1 Analog-Digital-Wandler 35
5.2.2 Analog-Teil 36
5.2.3 USB-Modul (PC) 37
5.2.4 LC-Display 37
5.2.5 Taster 38
5.3 Steuerprogramm 38
5.3.1 Struktur des Programmes 38
5.3.2 Ansteuerung des ADC 39
5.3.3 Single-Mode 40
5.3.4 USB-Mode 41

6 Fertigstellung des Projektes 47
6.1 Test der Gesamtschaltung 47
6.2 Vorbereitung der Präsentation 47
6.3 Vorführtermin 48

7 Arbeits- und Zeitplan 49

8 Handelnde Personen 50

Kapitel 1

Zielstellung

Ziel dieses Projektes ist es, ein mehrkanaliges Meßgerät zu entwickeln, welches über eine Universal-Serial-Bus (USB) Schnittstelle an einen Rechner angeschlossen werden kann, um aufgenommene Meßwerte zu speichern und zu visualisieren.

Die aufgenommenen analogen Signale werden digitalisiert und in Echtzeit an den Rechner übertragen. Außerdem ist es möglich, die momentanen Werte der einzelnen Kanäle direkt am Gerät über ein LC-Display und eine Reihe von Tastern abzufragen. Damit ist ein autarker Betrieb als Meßgerät auch ohne PC möglich. Einstellungen können direkt am Gerät bzw. am Rechner über entsprechende Software vorgenommen werden.

1.1 Spezifikationen

Das Design des Meßgerätes ist auf maximal vier Kanäle mit einer Auflösung von $8 Bit$ und einer Abtastrate von insgesamt $500 Kilosamples/s$ ausgelegt. Dabei soll es möglich sein, nicht benutzte Kanäle zu deaktivieren um einzelnen Kanälen mehr Bandbreite zur Verfügung zu stellen um so noch höhere Abtastraten zu ermöglichen. Im Rahmen dieses Projektes war es unser Ziel, mindestens zwei der maximal vier Kanäle als Schaltung aufzubauen.

An allen vier Eingängen ist eine Spannungsmessung im Bereich von $\pm 5V$ bzw. $\pm 15V$ möglich. Dabei kann sowohl der Meßbereich als auch die Einstellung für Gleichstrom bzw. Wechselstrom für jeden Kanal einzeln festgelegt werden. Höhere Spannungen sollen durch entsprechende Schutzschaltungen abgefangen werden.

Folgende Parameter können also für jeden Kanal einzeln am Meßgerät eingestellt werden:

- $\pm 5V$ / $\pm 15V$
- AC / DC / OFF

Die Stromversorgung des Gerätes ist mehrstufig aufgebaut. Während sich der USB-Teil durch den vom USB-Bus gelieferten Strom selbst versorgt, wird der Rest der Schaltung über ein externes Netzteil versorgt.

In Tabelle 1.1 sind noch einmal die wichtigsten Eckdaten festgehalten.

Tabelle 1.1: Spezifikationen

mögliche Kanalkonfigurationen	1, 2, 4
Meßbereiche (Spannungsmessung)	$\pm 5V$ / $\pm 15V$
AC/DC-Umschaltung	für jeden Kanal getrennt
Auflösung	$8Bit$
Abtastrate	4 Kanäle: $4 \times 125kSamples/s$
	2 Kanäle: $2 \times 250kSamples/s$
	1 Kanal: $1 \times 500kSample/s$

1.2 Aufbau

Das Meßgerät besteht aus mehreren Modulen, die jeweils für die Aufnahme, Wandlung und Übertragung der Meßwerte, sowie für die Kontrolle der einzelnen Bausteine verantwortlich sind.

Im *Analog-Teil* werden die aufgenommenen Größen zunächst für eine Digitalisierung vorbereitet. Dazu eine Skalierung der Eingangsspannung auf den Arbeitsbereich des Analog-Digital-Wandlers und eine Tiefpaßfilterung. Außerdem sind hier spezielle Schutzschaltungen vorgesehen, die den Eingang des Meßgerätes absichern.

Anschließend werden die so bearbeiteten Spannungen zur Digitalisierung durch den *Analog-Digital-Umsetzer (ADC)* geschickt.

Die digitalen Daten werden schließlich vom *USB-Teil* entgegengenommen und zum Rechner geschickt, wo sie durch entsprechende Software visualisiert und gespeichert werden können.

Die Kontrolle der einzelnen Teile sowie die Steuerung aller Abläufe übernimmt ein *Microcontroller*. Vor allem muß dabei der zeitgerechte Transport der digitalisierten Daten vom ADC zum USB-Chip, sowie die Ansteuerung des Displays und die Abfrage der Taster realisiert werden. Auch die Befehlskommunikation zwischen PC und Meßkarte wird vom Microcontroller übernommen.

Im Bild 1.1 ist das fertig aufgebaute Gerät zu sehen.

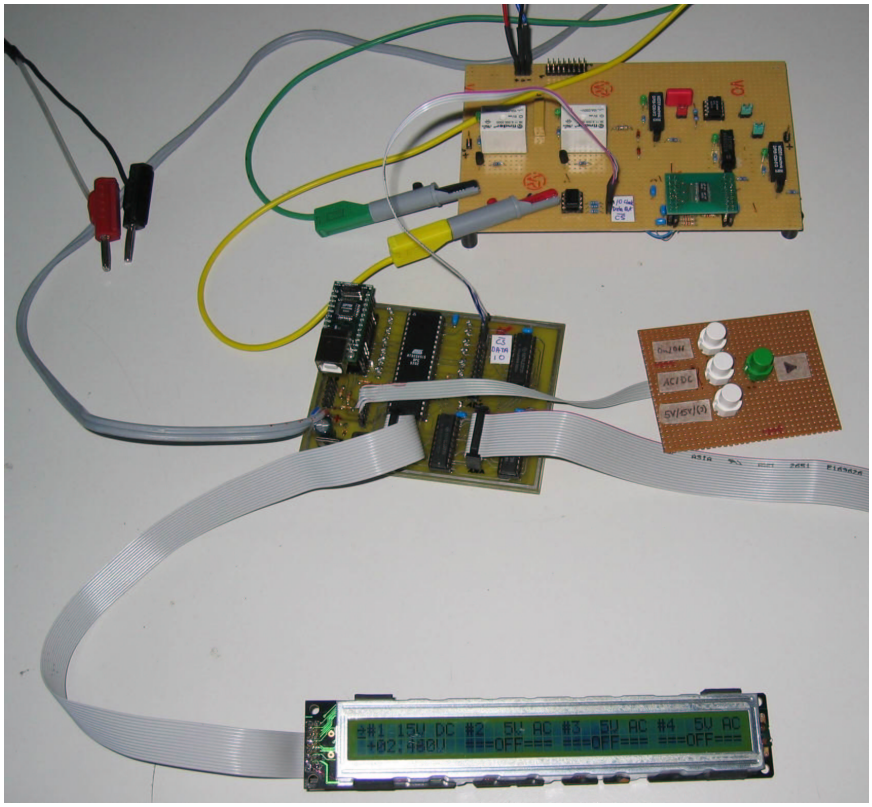


Abbildung 1.1: Fertiges Gerät (Ausbaustufe mit 2 Kanälen)

Kapitel 2

Analog-Teil

Team: OMAR MIKATI und WALID TFAYLI

2.1 Entwurf in Eagle

Nachdem die Schaltung theoretisch entworfen ist, ist es jetzt so weit, die Schaltung in EAGLE zu entwerfen. Zuerst werden die verschiedenen Bauelemente (Analogschalter, Filterbausteine und Verstärker) aus den Libraries eingefügt. Dann wird die gesamte Schaltung gezeichnet (siehe Bild 2.1).

2.2 Verstärkung

Für die Strommessung wird eine Verstärkerstufe mit einer 10 fachen Verstärkung in der Schaltung eingebaut. Dies wird durch einen nicht invertierenden Verstärker (Bild 2.2) verwirklicht.

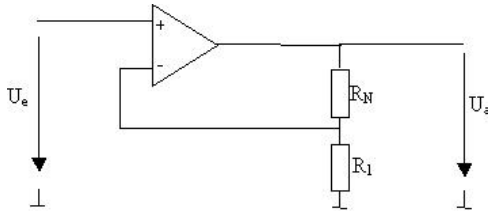


Abbildung 2.2: nicht invertierender Verstärker

Die Verstärkung lässt sich folgendermaßen berechnen: $\frac{U_a}{U_e} = 1 + \frac{R_N}{R_1}$.

2.3 Offsetfehler

Berechnung des LEAST SIGNIFICANT BIT: $U_{LSB} = \frac{1}{210-1} = 98mV$, da wir einen 12 Bit ADC haben. Der Offsetfehler nach der Verstärkung ist $60mV$ d.h. eine Offsetkompensation ist nicht nötig ($60mV < 98mV$).

2.4 Wahl des Filterbausteins

Um die halbe Sperrfrequenz am Filterbaustein **LTC1563-3** einzustellen muss man 6 gleichgroße Widerstände dazuschalten (siehe Datenblatt). Das ist sehr aufwändig und ungünstig denn es müssen sechs Schalter eingebaut werden. Nach kurzer Recherche war es möglich, mit Hilfe der Software FILTERCAD (`http://www.linear.com/`), durch Eingabe von Durchlaß- und Sperrfrequenz einen anderen Filterbaustein zu finden, den **LTC1569-7**, der nur einen externen Widerstand braucht. (siehe Datenblatt).

2.5 Blockschaltbild der neuen Schaltung

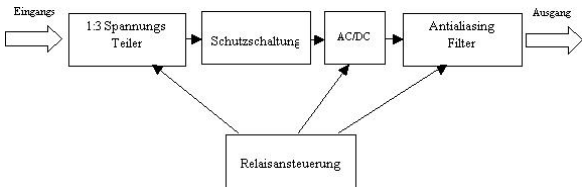


Abbildung 2.3: Blockschaltbild

2.6 Bauen und Testen

Danach wurde der Entwurf der Schaltung noch einmal verbessert. Als nächstes wurde das Board (Bild 2.4) entworfen und die Platine geätzt. Die Platine wurde bestückt. Es wurden verschiedene Messungen an der Platine gemacht. Dabei wurde festgestellt dass die Platine ein paar Fehler aufweist. Die Analogschalter schalten nicht alle wie gewünscht. Der Spannungsfolger ist defekt. Auf der Platine waren auch Leitungen, die nicht vorgesehen waren. Deswegen wurde für uns neue Schaltung zu entworfen die sich auf einen Kanal beschränkt. In dieser Schaltung (Bild 2.5) wurden die Analogschalter durch Relais ersetzt.

2.7 Relaisansteuerung

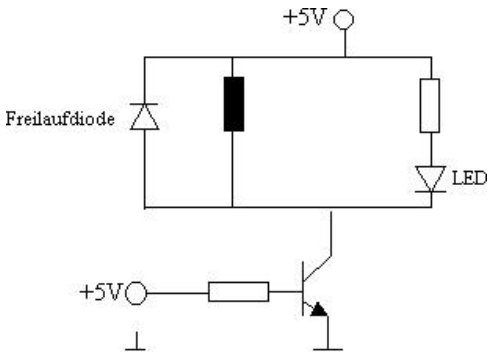


Abbildung 2.6: Relaisansteuerung

Die Freilaufdiode schützt die Schaltung vor der induzierten Spannung in der Spule L vom Relais. Die LED Diode zeigt den Status des Relais (high oder low) an. Der Transistor dient als Schalter der den Mikrokontroller von Stromüberziehung.

2.8 Sperrfrequenzeinstellung am Filter

Um die Sperrfrequenzeinzustellung am Filter zu ermöglichen gibt es zwei Möglichkeiten (siehe oben). Da die LTC1569-7 Filterbausteine kaputt gegangen sind, wurden die LTC1563-3 eingesetzt. Um nicht sechs Schalter in der Schaltung einzubauen, wurden zwei LTC1563-3 verwendet. Einer von den hat externe Widerstände, die doppelt so groß sind als beim anderen. Es wird zwischen den beiden Filtern umgeschaltet. Die Ausgänge der Filter sind entkoppelt.

2.9 Schutzschaltung

Die Schutzschaltung (Bild 2.7) besteht jetzt nur aus zwei Z-Dioden die entgegengesetzt geschaltet sind.

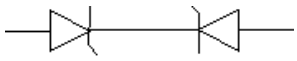


Abbildung 2.7: Schutzschaltung

Bei positiver Spannung sperrt eine Diode und die andere läßt durch. Bei negativer Spannung anders rum.

2.10 Fertigung der Platine

Die neue Schaltung wurde für uns auf einer Löcherplatine gebaut. Es ist ein Kanal gebaut worden.

2.11 Eigenschaften der neuen Schaltung

- Messbereichseinstellung
- Wechselstrom-/ Gleichstrommessung
- Filterung
- Sperrfrequenzeinstellung

2.12 Bauteilliste:

- 2x LTC1563-3
- 6x 42kΩ

- 6x 21kΩ
- 1x 2M Potentiometer
- 5x 100kΩ
- 5x LED Dioden
- 1x Rail to Rail OPV +/- 5V Spannungsversorgung
- 2x 1uF Kondensatoren
- 4x 0.1uF Kondensatoren
- 5x Umschalterrelais +5V Versorgungsspannung
- 5x npn Transitoren
- 5x Dioden

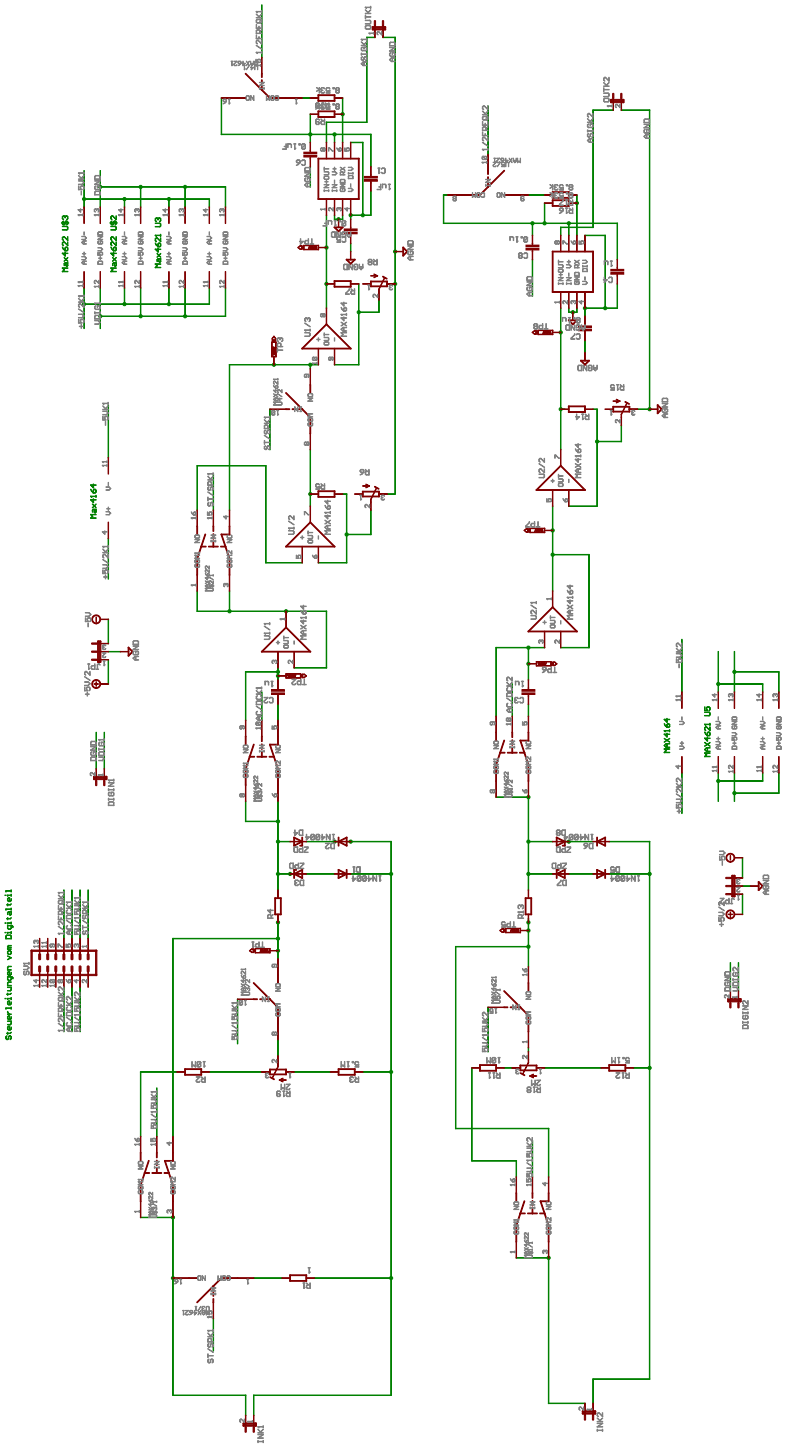


Abbildung 2.1: Schaltplan erste Analog-Platine

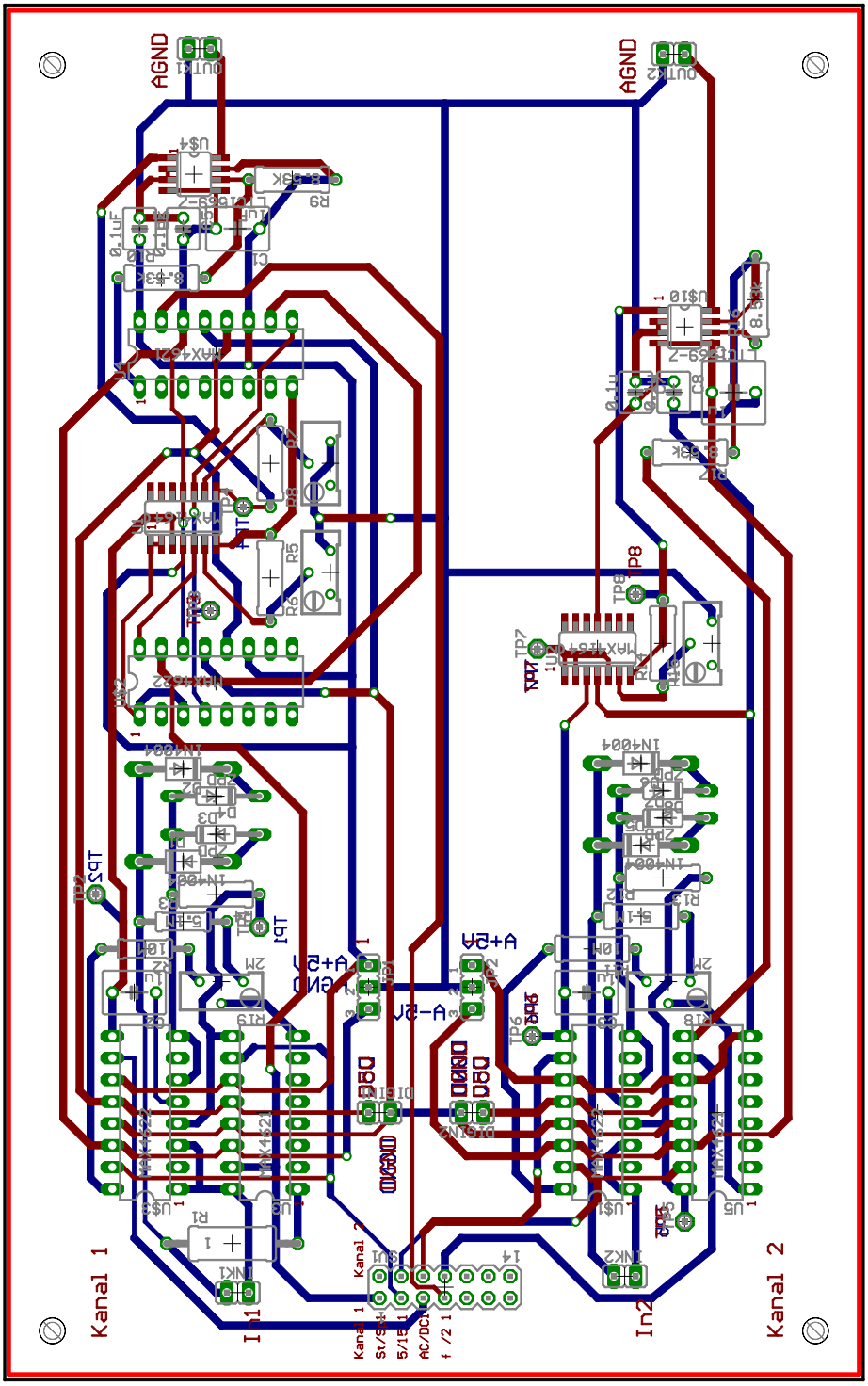


Abbildung 2.4: Erste Analog-Platine (defekt)

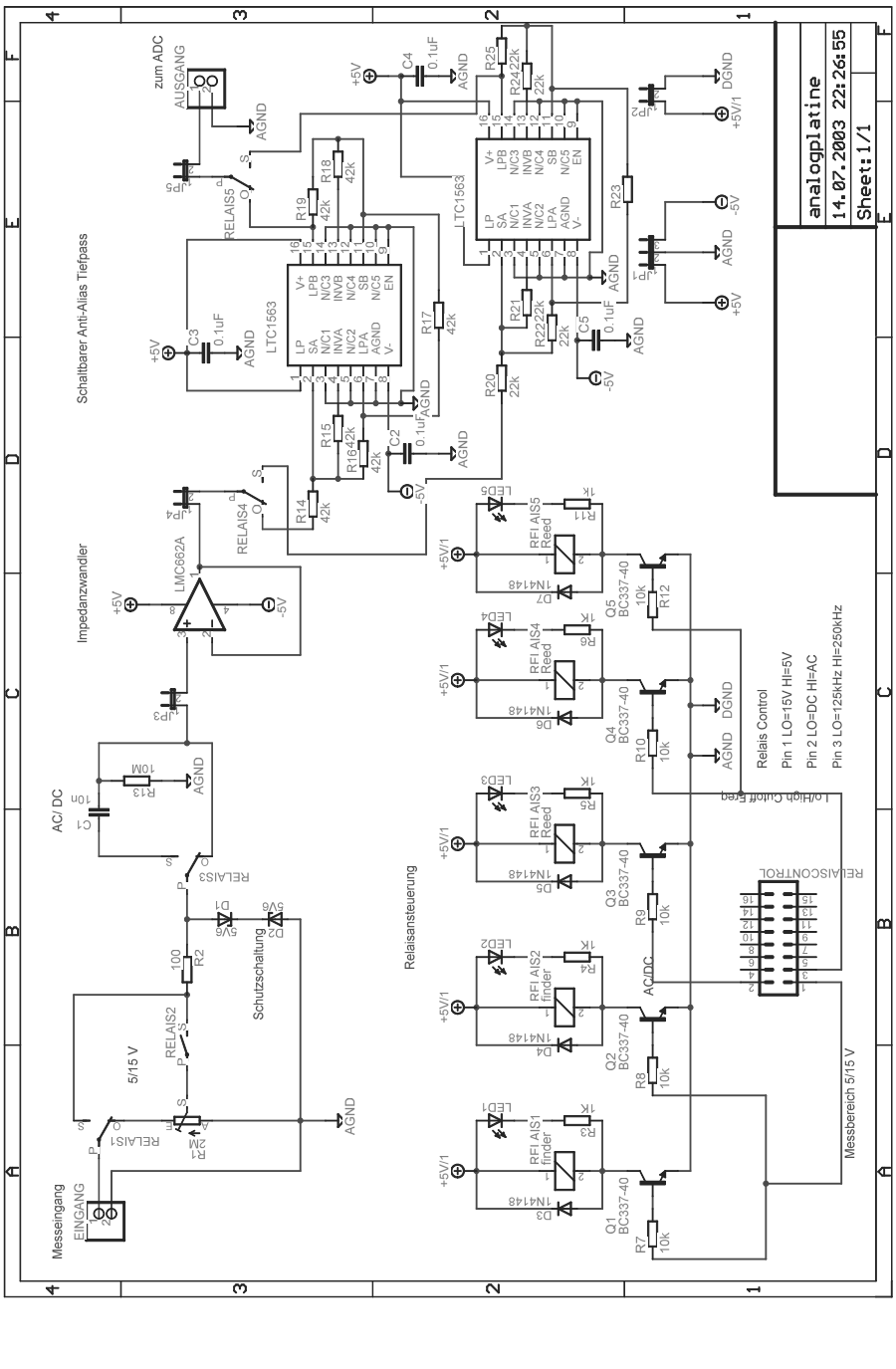


Abbildung 2.5: Neuer Schaltplan Analog-Platine

Kapitel 3

Analog-Digital-Umsetzer (ADC)

Team: EMINI DRITON und SHPEND MIRTA

3.1 ADC

Die mit den anderen Untergruppen festgelegten Spezifikationen für den Analog-Digital-Umsetzer waren:

- 4 Kanäle
- Mindestens 500 kS/s
- Mindestens 12 Bit paralleler Ausgang
- möglichst +5V Versorgungsspannung
- symmetrischer Eingang

Um einen entsprechenden ADC zu finden wurde bei folgenden Anbietern recherchiert: `http://www.maxim-ic.com/` und `http://www.analog.com/`. Für die Erfüllung der oben genannten Anforderungen kamen zwei potentielle ADCs in Frage: **AD 7864-1** und **AD 7892-1**.

Schließlich wurde für den AD 7864-1 entschieden. Dieses Bauteil kann simultan 4 Eingangssignale abtasten und mit 12 Bit-Auflösung umsetzen. Die einzelne zu verarbeitenden Kanäle können auf eine einfache Art und Weise (durch die Pins SL1-SL4) selektiert werden. Der Bereich der Eingangssignale ist einstellbar auf $\pm 5V$ oder $\pm 10V$.

Dieser ADU benötigt eine Versorgungsspannung von $+5V$. Die Umsetzungsgeschwindigkeit beträgt maximal (wenn nur ein Kanal betrieben wird) 500 kS/s. Werden alle 4 Kanäle betrieben, beträgt die Umsetzungsgeschwindigkeit 130 kS/s. Die umgesetzten Daten werden in 4 Ausgangsdatenregistern gespeichert. Auf die Datenregister ist ein einfacher Zugriff möglich. Die gelesenen Daten werden parallel weitergeleitet (**DB0** - **DB11**), wobei **DB11** das MSB ist.

Der einzige Nachteil war der kleine Pinabstand und das fehlende Package und Symbol in den vorhandenen EAGLE-Bibliotheken. Ein komplett neues Package und ein neues Symbol wurden für den AD7864 in EAGLE erstellt. Entsprechend der Anforderungen wurde der ADC wie in Bild 3.1 gezeigt beschaltet.

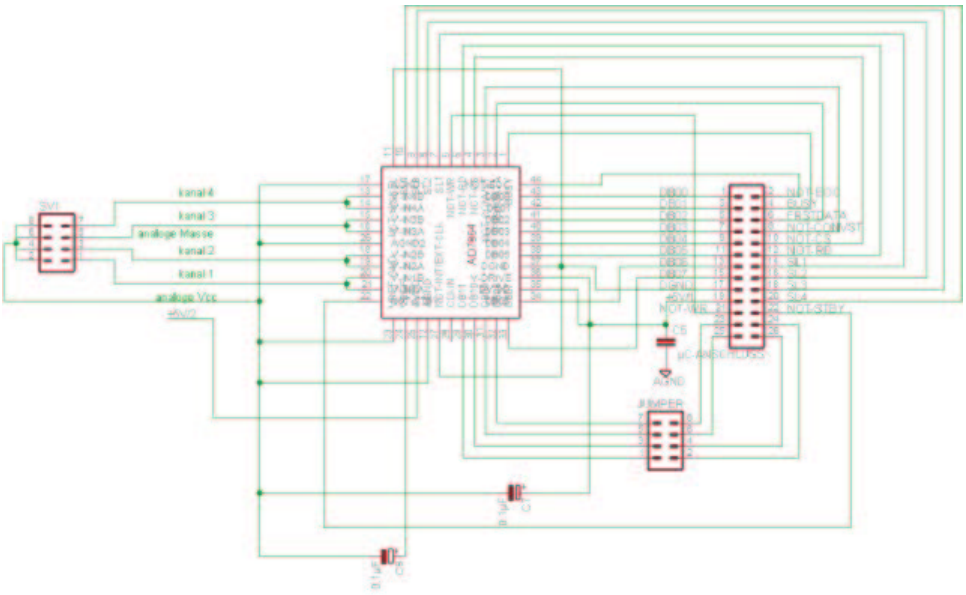


Abbildung 3.1: Beschaltung des ADC

Die Belegung des Pfostensteckers, der die Schnittstelle zum Mikrocontroller bildet ist im Abschnitt 5.2.1 genau beschrieben.

Die vollständigen Datenblätter für den ADU 7864-1 sind unter der folgenden Adresse zu finden: `http://www.analog.com/Analog_Root/productPage/productHome/0%2C2121%2CAD7864%2C00.html`

3.2 Spannungsversorgung

Für die Spannungsversorgung standen zur Debatte zwei Möglichkeiten: Batterien Netzteil Nach weiterer Diskussion wurde für den Netzteil entschieden. Der Netzteil liefert einen Ausgangsstrom von 500mA bei einer Spannung bis zu 12V. Nun bestand unsere Aufgabe darin, zunächst eine negative (-5V) und eine positive (+5V) Spannung daraus zu erzeugen. Die jeweiligen Spannungen sollen anschließend stabilisiert werden. Die stabilisierte positive Spannung soll sowohl der analogen als auch der digitalen Schaltungen zu Verfügung stehen.

Nach langen Recherchen haben wir uns für die in Bild 3.2 gezeigte Schaltung entschieden, die die Anforderungen erfüllt.

Die Strombegrenzung wird durch eine Sicherung realisiert. Die positiven 12V werden durch den Spannungsteiler und den OPV zunächst symmetrisch in +6V und -6V aufgeteilt. Die beiden erzeugten Spannungen werden durch die gewählten Regler (**MAX603** und **MAX1735**) auf +5V und -5V stabilisiert. Mit den +5V werden die Mikrocontroller-Schaltung, die Bauteile der analogen Schaltung sowie der ADU versorgt.

Um eine bessere Signalaufteilung zu erzielen, wurde eine digitale und eine analoge Masse erzeugt. Um den Einfluß der Störung seitens der digitalen Bauelemente auf die Versorgungsspannung zu beseitigen, wurden Spulen in die Schaltung eingefügt. Die Spulen bilden mit den Eingangskapazitäten der zu versorgenden Schaltungen eine Art Siebglied, das die Wirkung eines Filters hat. Für das Layout in EAGLE mußten wir zunächst für beide Regler Libraries erstellen, da sie für die gewählten Bauteile nicht existierten.

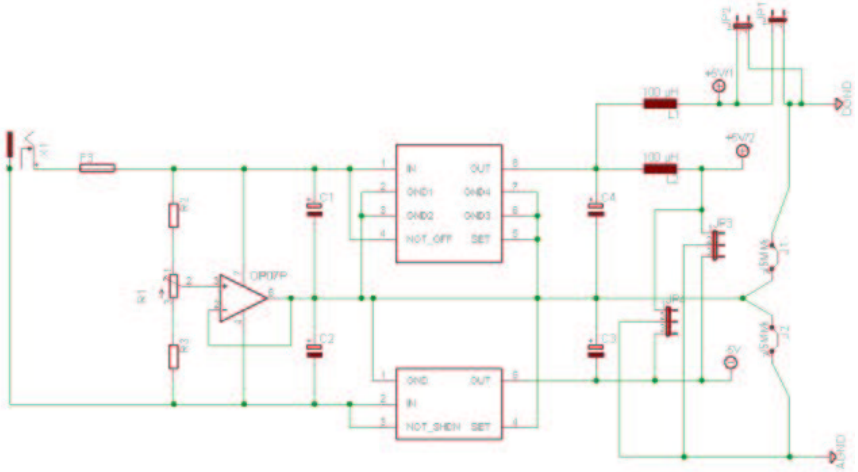
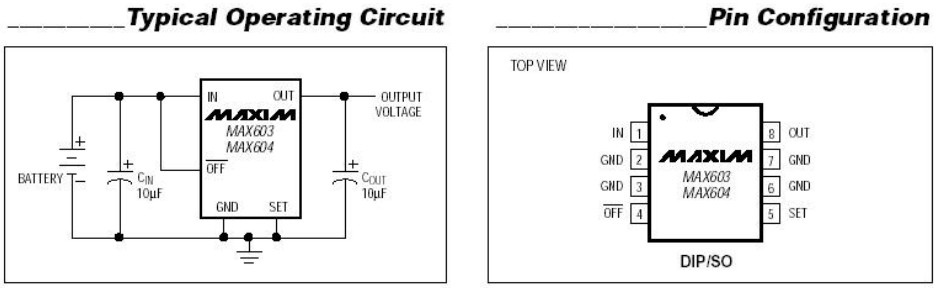


Abbildung 3.2: Schaltplan Spannungsversorgung

Leider hat die oben entworfene Schaltung einen Haken!!! Bei den durchgeführten Tests hat sie im Großen und Ganzen funktioniert. Jedoch haben die Regler nicht ausreichend Strom geliefert. Der verwendete OPV (**OP07**) war für die Anforderungen zu schwach und lieferte nicht ausreichenden Strom für die beiden Regler. Durch die Verwendung eines leistungsfähigeren OPV müsste die Funktionalität der Schaltung gewährleistet sein. Leider reichte uns die Zeit nicht, einen leistungsfähigeren OPV in die Schaltung einzusetzen.

MAX603 (CPA), Low-Dropout, 5 V- Spannungsregler

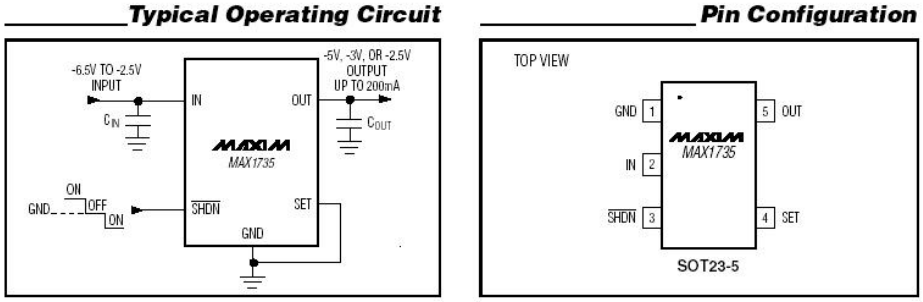


™ Dual Mode is a trademark of Maxim Integrated Products.

- 500mA Ausgangsstrom mit Foldback- Strombegrenzung
- Genau $5V$ oder einstellbare Ausgangsspannung
- Eingangsbereich: $2,7V \ldots\ 11,5V$
- $15\mu A$ Ruhestrom
- $2\mu A$ Strom im Shutdown-Mode
- Übertemperaturschutz
- Ruhestromschutz
- Verlustleistung $727mW$

Die vollständigen Datenblätter des Reglers sind unter der folgenden Adresse zu finden: `http://pdfserv.maxim-ic.com/arpdf/MAX603-MAX604.pdf`

MAX 1735 (SOT 23-5)



- 200mAAusgangsstrom
- Genau $-5V$ oder einstellbare Ausgangsspannung
- Eingangsbereich: $[2,5V \ldots -6,5V]$
- 85μA Ruhestrom
- 2μA Strom Shutdown-Mode
- Übertemperaturschutz
- Verlustleistung 727mW

Die vollständigen Datenblätter des Reglers sind unter der folgenden Adresse zu finden: `http://pdfserv.maxim-ic.com/arpdf/MAX1735.pdf`

3.3 Platine

Beide Schaltungen wurden auf eine gemeinsame Platine (Bild 3.3) entworfen.

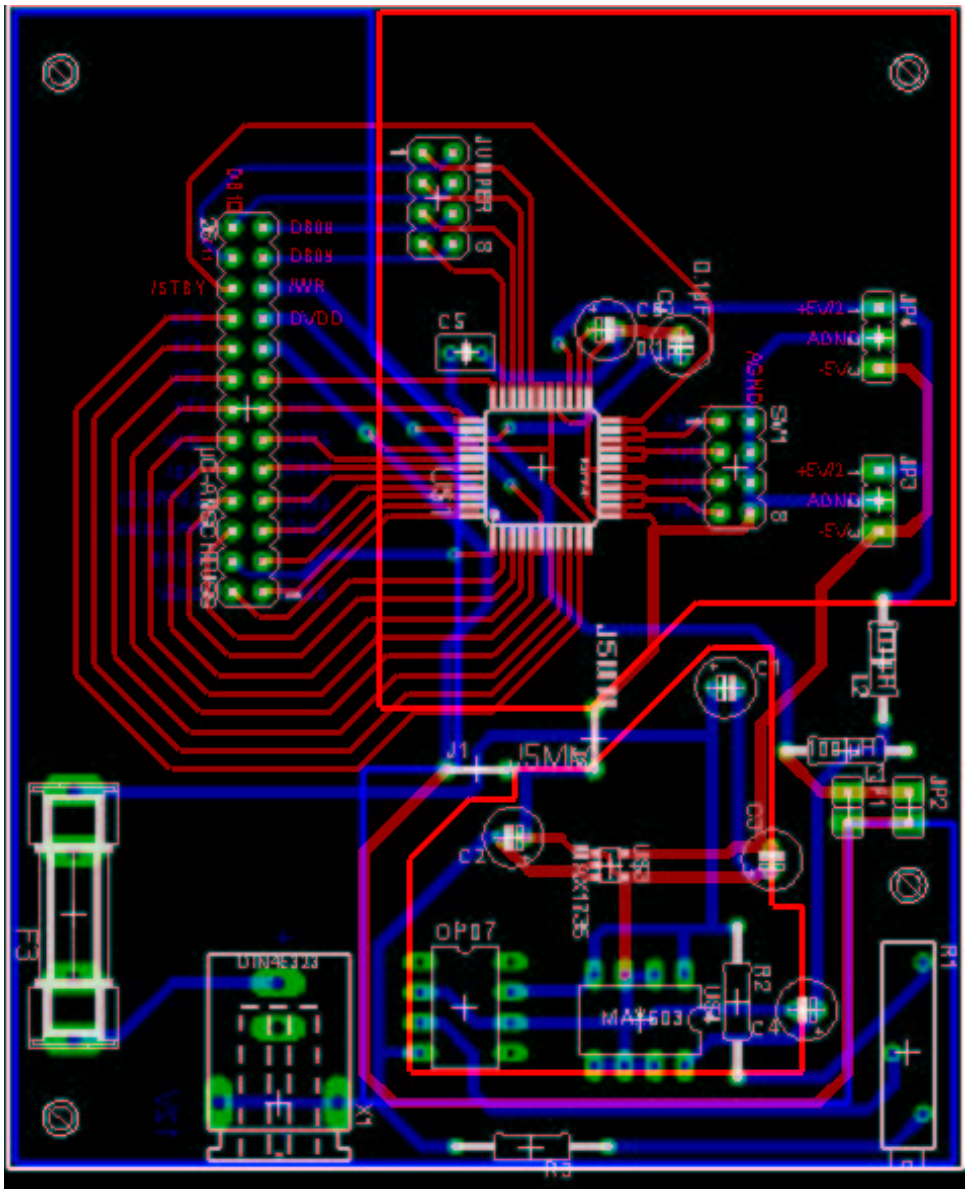


Abbildung 3.3: Gemeinsame Platine mit ADC und Spannungsversorgung

Kapitel 4

USB-Interface

Team: JÖRN HOHLWEIN und HELGE KRAMBECK

4.1 Einführung

4.1.1 Allgemeines

Für die USB Meßkarte entwickelt die USB Gruppe die Software, die für die Anbindung der Karte an den PC nötig ist. Für die schaltungstechnische Anbindung wurde ein im Praktikum verfügbares USB Modul gewählt, und zwar das **DLP-USB245M** von DLP-DESIGN (`http://www.dlpdesign.com`). Dieses Modul muß an den Atmel Microcontroller **AT90S8515** angebunden werden, der die Steuerung der Meßkarte übernimmt. Ausserdem muß das Modul mit einer PC-Software, die eine komfortable Benutzeroberfläche bietet, angesprochen werden.

4.1.2 Vorstellung des USB Moduls

Das DLP-USB245M Modul basiert auf dem hochintegrierten **FT245BM USB FIFO Chip** von `http://www.ftdichip.com`. Es läßt sich in einen DIL 24 Sockel einsetzen, der sich leicht löten und in Eagle für das Platinenlayout verarbeiten läßt. Das Modul hat folgende wichtige Merkmale:

- Maximale Übertragungsgeschwindigkeit 8 Megabit/s (tatsächlich mögliche Geschwindigkeit muß experimentell bestimmt werden)
- FIFO Puffer-Architektur, einfach an Microcontroller koppelbar
- 5 V Spannungsversorgung (direkt über USB möglich, mit vielen Power Management Einschränkungen und daher nicht für größere Schaltungen empfohlen)
- Stromverbrauch 25 mA
- D2XX- DLL Treiber für WINDOWS 9X,ME,2000,XP einfach integrierbar in VISUAL C++, VISUAL BASIC, DELPHI etc.
- Virtual Com Port Treiber auch für LINUX, MAC, WIN CE etc.

Anhand dieser Merkmale wurden erste Parameter für die Analoggruppe und die ADC Gruppe (z.B. maximale Durchlaßgrenzfrequenz der Tiefpässe) abgeleitet.

4.1.3 Inbetriebnahme des USB Moduls

Um das Modul zu testen bedarf es einer Spannungsversorgung, die durch den USB Bus erfolgen kann (hierzu wird der Pin 12 des Moduls mit den entsprechenden Eingängen verbunden). Außerdem bedarf es einer Schnittstelle zu einem Microcontroller, in unserem Falle dem **Atmel 90S8515**, der auf einem **STK 200 Evaluation Board** sitzt.
Damit diese Anbindungen gewährleistet sind und die USB Gruppe somit die Treiber für das USB Modul installieren kann und mit VISUAL C++ 6 per DLL und Testprogramm das Modul in Betrieb nehmen kann, wurde eine Testplatine geätzt, die den obigen Anforderungen genügt.
Zusätzlich wurde ein Jumper eingefügt, der die Spannungsversorgung auch über das STK 200 erlaubt. Der Schaltplan ist in Abbildung 4.1 und das Platinenlayout in Abbildung 4.2 dargestellt.

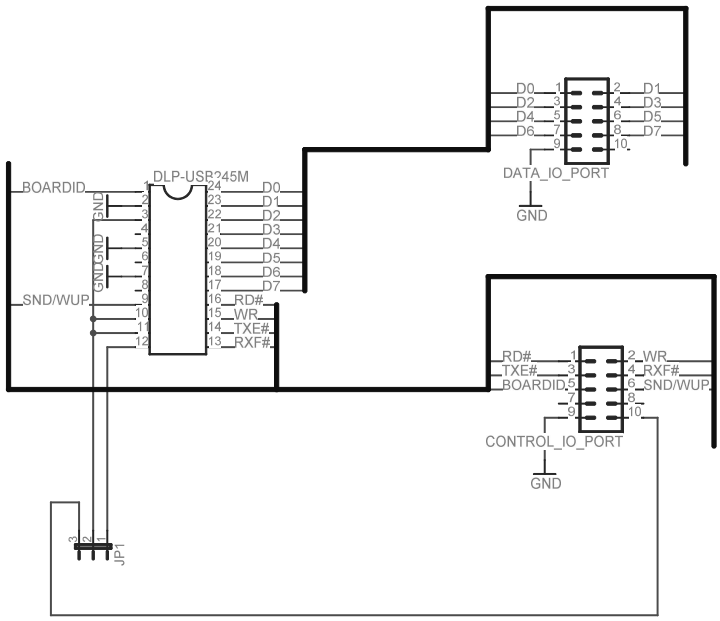


Abbildung 4.1: Schaltplan USB nach Microcontroller Platine

Auf dem Schaltplan sind die Kontroll- und Datenleitungen zu erkennen, die auf zwei 10 polige Pins herausgeführt werden. Die Pinbelegung orientiert sich an der Pinbelegung des STK200 Evaluation Boards für die Amtel Microcontroller. Zusätzlich wurden die Pins Board ID und SND/WUP zu Testzwecken herausgeführt. Das ist auch kein Problem, da die Ports des Atmel ja jeweils 8 Leitungen frei haben, die per Software definiert werden können.

Die Platine wurde von Berit in zweifacher Ausführung geätzt, damit sie auch für das zweite USB Modul benutzt werden kann. Das Modul wurde dann erfolgreich in Betrieb genommen, zumindest wurde es vom Windows 2000 Laborrechner erkannt und durch das Testprogramm konnte die Seriennummer ausgelesen werden, allerdings reagierte das Modul noch nicht weiter, vermutlich weil die Steuerleitungen noch nicht an den Atmel gekoppelt waren.

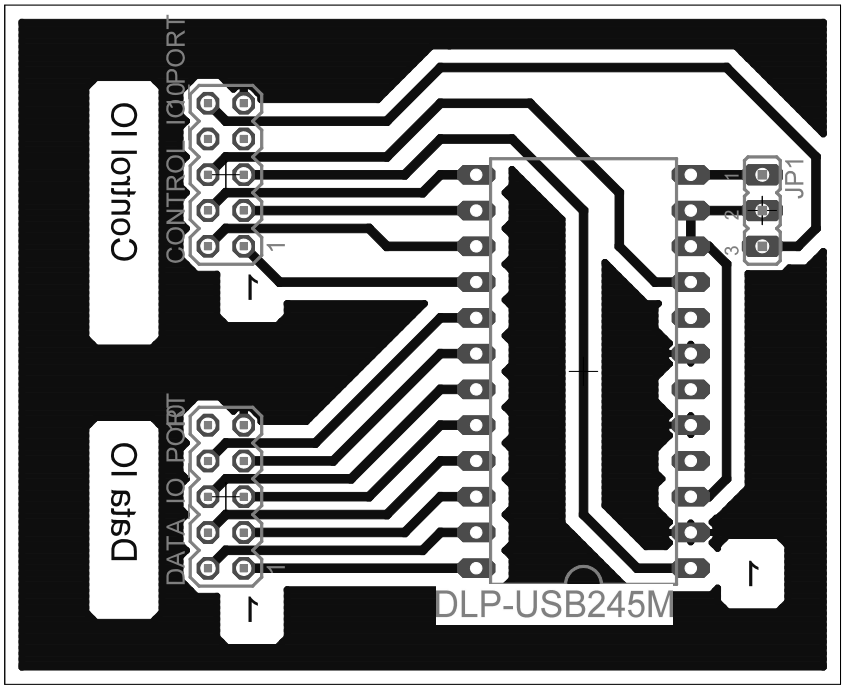


Abbildung 4.2: Platinenlayout USB nach Microcontroller Platine

4.2 Microcontroller Programmierung

4.2.1 Wahl der Entwicklungsumgebung

In diesem Termin haben wir die Programmierung des im Labor eingesetzten Atmel 90S8515 kennengelernt. Hierzu haben wir Testprogramme, die zum Beispiel LEDs blinken lassen ausgeführt (diese sind an Port B des STK 200 Evaluation Boards angeschlossen). Dabei haben wir ein Testprogramm in Assembler (aus einer Atmel Application Note) mit AVR-STUDIO kompiliert und das .HEX File per STK 200 in den Atmel MC geladen. Dabei gab es aber zahlreiche Probleme, da das STK 200 das Programm oft fehlerhaft zum MC übertragen hat. Zusätzlich fehlten uns Grundlagen in der Assembler Programmierung, die sicherlich für echtzeitkritische Anwendungen wichtig ist, aber für unsere Bedürfnisse nicht unbedingt erforderlich ist.

Daher fiel die Entscheidung auf WINAVR, einer Entwicklungsumgebung, die auf dem GNU C COMPILER beruht (mit einer Library, die speziell auf die verschiedenen Atmel Microcontroller angepasst ist). Die Programmierprobleme wurden mit dem STK 500 Evaluation Board gelöst, daß unter WINDOWS 2000 ein besseres Programmierverhalten als das STK 200 zeigt. Es wurde ein erstes Beispielprogramm kompiliert und in den MC geladen.

4.2.2 Testprogramme in C++

Für die geplanten Bandbreitenmessungen des USB Chips wurden zwei Testprogramme entwickelt. Das erste Testprogramm hat ein Lauflicht erzeugt, um die Ausgabe von Daten auf MC Ports kennenzulernen. Hier die main Routine des C Quellcodes:

Program 4.1: Quellcode Lauflicht-Programm

```
int main (void) {
   ioinit ();
   outp(0xff, DDRB);  // Port B als Ausgang initialisieren
   uint8_t a=1;       // Zaehlvariable fuer Lauflicht
   uint8_t s=0;       // Umschalter fuer Lauflichtrichtung

   for (;;) {  // Endlosschleife
      for (i = 0; i <255; i++)
         for (j = 0; j<255; j++) ; // geschachtelte for schleife als Delay
      outp(~a, PORTB); // Variable a auf PORTB ausgeben (LEDs)
      if (s==0) a=a*2; // Lauflicht hochlaufen
      else
         a=a/2;       // Lauflicht herunterlaufen
      if (a==128) s=1; // Laufrichtung umschalten
      if (a==1) s=0;  // Laufrichtung umschalten
   }
   return (0);
}
```

Das Programm wurde erfolgreich mit WINAVR kompiliert (make Befehl unter DOS) und per AVR STUDIO und STK 500 in den Atmel geladen. Es wurde ein weiteres Programm entwickelt, das einen Port mit Schaltern als Eingang nutzte und die Schalterstellung auf Port B mit LEDs signalisierte. Hier die main Routine dazu:

Program 4.2: Quellcode Schalter-Programm

```
int main (void) {
   ioinit ();
   outp(0xff, DDRB);  // Port B ist Ausgang
   outp(0x00, DDRA);  // Port A ist Eingang
   uint8_t schalter;
   for (;;) {  // Endlosschleife
      schalter=inp(PINA);  // Schalterstellung auf Port A auslesen
      outp(schalter, PORTB); // Schalterstellung auf PORTB (LEDs) ausgeben
   }
   return (0);
}
```

Damit wurden die Grundlagen geschaffen, um Microcontroller und USB Modul mithilfe der Daten- und Flußkontrolleleitungen zu koppeln.

4.2.3 Entwicklung eines MC-Leseprogramms

Mit den gewonnen Kenntnissen aus den grundlegenden MC-Programmen haben wir uns an die Entwicklung eines MC-Leseprogramms für das USB Modul gemacht.
Ziel dieses Programmes ist es, Daten, die vom PC an das USB Modul geschickt werden und dort im FiFo Puffer liegen mit dem Atmel auszulesen und Zwecks Kontrolle auf LEDs auszugeben oder mittels der seriellen Schnittstelle des Atmels an einen anderen PC zu schicken, der per Hyperterminal die serielle Schnittstelle überwacht.
Der Atmel bietet an seinem Port D auf den Pins 0 und 1 Rx bzw. Tx Leitungen, die direkt für die serielle Schnittstelle genutzt werden können und an eine RS232 Buchse auf dem STK 500 Board angeschlossen werden können (durch einfaches patchen von Pins, wie auf dem STK 500 üblich). Damit die UART-Funktionen genutzt werden können, muß die Datei ***uart.h*** in das Programm included werden und dort muß die gewünschte Baudrate sowie die Oszillatorfrequenz des Atmels festgelegt werden.
Wenn Daten im USB Modul Empfangs-FiFo vorhanden sind, geht der Pin RXF# des USB Moduls low, ansonsten ist er high. Der Pin RXF# wird mit der Anweisung `if (bit_is_clear(PINC,3))` untersucht, dies geschieht durch Pollen (Endlosschleife). Sobald der Pin RXF# low ist, wird

durch `cbi(PORTC, 0);` und `sbi(PORTC, 0);` der Pin RD gestrobt, was dem USB Modul signalisiert, das wir Daten haben möchten.
Daraufhin schreibt das USB Modul das erste Byte im FiFo auf seine Datenleitungen, die wir mit dem Atmel durch die Anweisung `ausg=PIND;` auslesen und dann für Diagnosezwecke auf dem Port A des Atmels mit `outp(ausg,PORTA);` ausgeben und mit den Zeilen 28 bis 32 an die serielle Schnittstelle unseres Überwachungs-PCs schicken.
Dazu müssen wir das Byte in eine dreistellige Dezimalzahl (0-255) umwandeln, da sonst auf dem anderen Rechner nur ein unverständliches ASCII Zeichen erscheint. Dies geschiet in den Zeilen 25 bis 27.

Program 4.3: Quellcode USB-Leseprogramm

```
1  int main (void) {
2     outp(0xe3, DDRC); //PORT C als Kontrollleitungen initialisieren
3     outp(0xff, DDRB); //PORT B als Ausgang initialisiert fuer die LEDs
4     outp(0xff, DDRA); //PORT A als Ausgang initialisiert fuer Datenanzeige
5     outp(0x00, PORTA); // PORT A mit 0 initialisieren
6     outp(0x23, PORTC); // Initialisierung: WR,RD# und SND/WUP auf high
7                        // setzen, es wird nichts
8                        // gesendet und nichts empfangen zu Anfang
9     outp(0x00, DDRD); //PORT D als Lese-Datenport initialisiert
10    ausg=0;
11    UART_Init(); // Serielle Schnittstelle initialsieren
12
13    for (;;) { /* Note [6] */
14      outp(~(inp(PINC)), PORTB); //Kontrolleitungen auf LEDs ausgeben
15      outp(ausg,PORTA); // gelesene Daten auf Port A ausgeben
16      if (bit_is_clear(PINC,3)) {
17         // Daten im FIFO Empfangspuffer vorhanden
18         cbi(PORTC, 0); // RD auf Low Setzen
19         sbi(PORTC, 0); // RD auf High Setzen
20
21         // Byte einlesen
22         ausg=PINA;
23
24         // Konvertieren des Bytes in dreistellige Dezimalzahl
25         ausg1=ausg / 100;
26         ausg2=(ausg - ausg1*100)/ 10;
27         ausg3=(ausg - ausg1*100 - ausg2*10)/ 1;
28         UART_SendByte(0x30+ausg1); // 100er Stelle
29         UART_SendByte(0x30+ausg2); // 10er Stelle
30         UART_SendByte(0x30+ausg3); // 1er Stelle
31         UART_SendByte(0x0A); // Line Feed
32         UART_SendByte(0x0D); // Carriage Return
33
34      }
35    }
36    return (0);
37 }
```

Beim Testen des Programms wurden die Daten korrekt vom Rechner übernommen, wir konnten durch patchen der LEDs auf Port A des Atmels die gesendeten Bitmuster kontrollieren. Hier zeigte sich der Vorteil des STK500 Evaluation-Boards. Außerdem wurden die Daten einwandfrei von dem einen PC über USB, den Atmel Controller und seine serielle Schnittstelle an einen anderen PC geschickt. Die Flußkontrolle verhinderte ein Überlasten des Puffers, das System hat sich immer an die Geschwindigkeit des schwächsten Glieds (hier die RS232 übertragung mit 9600 baud) angepaßt. Außerdem gingen keine Bytes verloren, das ist für die Kontrollkommunikation mit der Messkarte von großer Wichtigkeit.

4.2.4 Entwicklung eines MC-Schreibprogramms

Mit den so gewonnenen Erkentnissen aus dem USB Leseprogramm haben wir ein USB Schreibprogramm entwickelt, was die Übertragung von Meßdaten an das USB Modul und

damit den PC simulieren sollte. Die Vorgehensweise entspricht dem Leseprogramm, da der Handshake des USB Moduls symmetrisch aufgebaut ist. Wir stroben einfach das WR Signal, wenn das USB Modul TXE# Low gesetzt hat und damit signalisiert, daß Daten geschrieben werden können. Dazu wird ein Port des Atmels (hier Port A) als Ausgangsport definiert. Wir haben eine Delayschleife eingefügt, um den PC erstmal nicht zu überlasten und ein Rechtecksignal erzeugt, indem wir die ausgegebene Variable einfach hochgezählt haben.

Program 4.4: Quellcode USB-Schreibprogramm

```
1  int main (void) {
2     outp(0xe3, DDRC); //PORT C als Kontrollleitungen initialisieren
3     outp(0xff, DDRB); //PORT B als Ausgang initialisiert für die LEDs
4     outp(0x23, PORTC); // Initialisierung: WR,RD# und SND/WUP auf high
5                        // setzen, es wird nichts
6                        // gesendet und nichts empfangen zu Anfang
7     outp(0xff, DDRA); // PORT A als Schreib-Datenport initialisiert
8     outp(0x00, PORTA);// PORT A mit Nullen initialisiert
9     test=0;
10    for (;;) { // Endlosschleife
11      if (bit_is_clear(PINC,2)) {
12        // TXE# ist low, Daten können geschrieben werden
13        cbi(PORTC, 1); // WR Low setzen
14        sbi(PORTC, 1); // WR High setzen
15        outp(test, PORTA); // Daten in USB Modul schreiben
16        test++; // "Sägezahnspannung" erzeugen
17      }
18      // Warten, damit PC Daten in Echtzeit darstellen kann
19      for(j=0;j<255;j++) {
20         for(i=0;i<20;i++) { }
21      }
22    }
23    return 0;
24 }
```

4.2.5 Geschwindigkeitsmessung Atmel nach PC

Da wir in unserer ersten Spezifikation die gesamte Datenrate des USB Moduls von 1 Megabyte pro Sekunde ausnutzen wollten, mußten wir prüfen, ob wir mit dem Atmel diese Datenrate auch liefern können. Der Atmel AT90S8515 wird maximal mit 8 MHz betrieben, es dürfen also nicht mehr als 8 Takte für ein Byte verbraucht werden, damit eine Übertragungsrate von 1 Megabyte pro Sekunde möglich ist. Da die Clock des Atmels und die des USB Moduls nicht synchron sind, entstehen Taktverluste. Außerdem erfordert das Programm sicherlich mehr Takte, da ja mehr getan werden muß, als nur Daten zu schicken (LCD Display, Taster, Relaissteuerung etc.). Der Atmel AT90S8515 kann leicht durch einen ATMEGA ersetzt werden, der 16 MHz Taktung bietet und ansonsten pinkompatibel zum AT90S8515 ist. Wir haben unsere Programme soweit vereinfacht, daß nur doch die WR / RD Leitungen gestrobt wurden, ohne irgendein Handshake zu betreiben. Allerdings haben wir das mit dem gcc compiler und avr-libc gemacht, was sicherlich zu nicht ganz so optimalem Code wie Assembler geführt hat. Trotzdem ist das Ergebnis brauchbar, da die Microcontrollergruppe ja wie gesagt wesentlich mehr zu tun hat, während Daten an den PC geschickt wird. Hier die Meßergebnisse:

Durch diese Meßergbnisse wurde die Spezifikation von 1 Megabyte pro Sekunde auf 500 kilobyte pro Sekunde reduziert, spätere Ergebnisse haben gezeigt, daß der Microcontroller mit der gesamten Steuerung nur noch auf 330 kb/s gekommen ist, trotz Programmierung in Assembler und dem Konzept, das nur die Kontrolleitungen des USB Moduls und des ADCs gesteuert werden und die Daten ansonsten ohne Kontrolle durch den MC über den Bus gehen.

Tabelle 4.1: Geschwindigeitsmessung Microcontroller

Richtung	AT90S851 (8 MHz)	ATMEGA (16 Mhz)
Lesen	590 kb/s	685 kb/s
Schreiben	560 kb/s	677 kb/s

4.3 Entwicklung des PC Programms mit Borland Delphi 5

4.3.1 Windows XP Treiber

Für die weitere Entwicklung des USB Schaltungsteils erschien der Einsatz eines Notebooks mit Windows XP als sinnvoll, da dieses der USB Gruppe neuerdings zur Verfügung steht. Die Einbindung des Moduls in das Betriebssystem muß jeweils vom Administrator vorgenommen werden, was Probleme am Laborrechner erzeugte (Standardmäßig besitzt unsere Gruppe *ele21* ja keine Administratorrechte). Außerdem wäre auf dem Notebook eine Entwicklung des PC Programms mit BORLAND DELPHI 5 möglich, mit dem die USB Gruppe bereits umfassende Erfahrungen hat.

Dies brachte nun Probleme mit sich, da Windows XP das USB Modul standardmäßig mit verkehrten Treibern in das System einbindet, die zu Störungen mit den D2XX Treibern führen. Deshalb wurde eine Neuprogrammierung der EEPROMs der beiden USB Module nötig, damit sich diese mit der PID (Product ID) von 6006 statt 6001 am System anmelden, die unter Windows XP keine Konflikte verursacht. Außerdem wurde eine Modifizierung der .INF Dateien des Treibers nötig, da diese PIDs in der .INF Datei stehen müssen. Der EEPROM wurde mit dem FTD2XXST Programm neu geschrieben, das sehr umständlich zu bedienen ist (es müssen erst alle Felder ausgefüllt werden, bevor weitere Optionen eingestellt werden können, der EEPROM kann nicht im Klartext ausgelesen werden etc.) Schließlich wurden beide Module neue programmiert und die modifizierten D2XX Treiber wurden am Laborrechner installiert.

Nachdem wir nun eine saubere Schnittstelle zum Microcontroller hatten, die wir ausführlich getestet haben, damit der MC Gruppe unangenehme Überraschungen erspart bleiben, konnten wir uns an die Entwicklung der PC Anwendung machen. Um die Entwicklung zu beschleunigen, haben wir ein Windows XP Notebook eingesetzt, die USB Treiber und der EEPROM des USB Moduls waren ja schon angepaßt auf Windows XP (die PID haben wir auf 6006 gesetzt bei beiden, siehe Kapitel 5).

Wir haben uns für Borland Delphi 5 Professional als Entwicklungsumgebung entschieden, da wir schon umfassende Erfahrungen mit dieser Entwicklungsumgebung hatten. FTDI Chip hat eine sehr gute Delphi-Testanwendung und eine Delphi-Library für das Modul bereitgestellt, so daß die grundsätzliche Programmierung des USB Moduls kein großes Problem darzustellen schien.

4.3.2 Entwicklung der PC-Messkartensoftware

Zunächst haben wir die grundsätzliche Oberfläche des Programms designt. Es sollten 2 digitale Oszilloskope sichtbar sein, sowie ein Statusfeld für das USB Modul und Steuerfelder für die Kanaleinstellung und Triggerung. Das USB Modul wird über Funktionen in der **D2XXUnit.pas** angesteuert, die gekapselte Funktionsaufrufe an die DLL des D2XX Treibers enthält, um dem Entwickler einige Arbeit mit Pointern usw. abzunehmen. Die gesamten Delphi-Quelltexte (ca. 2000 Zeilen Quelltext) finden sich in dem getrennten Dokument

delphi_listings_usb.pdf. Hier einige Quelltextauszüge, die die Ansteuerung des USB Moduls mit den D2XX Treibern zeigen (gekapselt über D2XXUnit.pas) Sie werden im Delphi Programm zyklisch ausgeführt, da eine Interrupt-Steuerung bei vollem Empfangspuffer mit den D2XX Treibern nicht möglich ist:

Program 4.5: Delphi-Ansteuerung DLP USB Modul

```
1  // Anzahl der angeschlossenen DLP USB Geraete ermitteln
2  GetFTDeviceCount;
3  if (FT_Device_Count=1) and (UsbDeviceOpen=false) then
4    begin
5      // Seriennummer und Beschreibung auslesen
6      GetFTDeviceSerialNo(0);
7      SerialEdit.text:=FT_Device_String;
8      SerialLabel.enabled:=true;
9      GetFTDeviceDescription(0);
10     BeschreibungMemo.text:=FT_Device_String;
11     BeschreibungLabel.enabled:=true;
12    if (Open_USB_Device()=FT_OK) then
13     begin
14       Reset_USB_Device;  // warning - this will destroy any pending data.
15       Set_USB_Device_TimeOuts(2000,2000); // read and write timeouts = 1 s
16     end;
17   end;
18
19 // Daten vorhanden ?
20 usbstatus:=Get_USB_Device_QueueStatus();
21
22 if (FT_Q_Bytes>0) then
23   begin
24     // Daten sind vorhanden und k"onnen ausgelesen werden
25    i := Read_USB_Device_Buffer(FT_Q_Bytes)
26    for k:=0 to i-1 do
27      begin
28        showmessage('Byte '+inttostr(FT_In_Buffer[k])+' empfangen');
29      end;
30   end;
31
32 // Befehl senden
33 FT_Out_Buffer[0]:=befehl;
34 // 1 Byte schreiben
35 written:= Write_USB_Device_Buffer(1)
36 if (written=1) then
37   begin
38     showmessage('Byte erfolgreich geschickt');
39   end
40 else
41   begin
42     showmessage('Keine Daten geschickt !');
43   end;
```

Die Oberfläche des USB Programms ist in Bild 4.3 gezeigt.

Damit unser Programm die Initialisierung des USB Moduls von alleine erledigt, haben wir uns ein Zustandsdiagramm (Bild 4.4) ausgedacht. Das Programm überwacht mit Timern den aktuellen Zustand des USB Moduls und fällt in den Ausgangszustand zurück, wenn das USB Modul nicht mehr verfügbar ist. Das befreit den Anwender von der Arbeit, eine Menge Buttons zum Suchen und Öffnen des Moduls durchzuklicken, wie es in den Beispielprogrammen nötig ist.

Zusätzlich wurde eine Kontrollkommunikation mit dem Mikrocontroller notwendig, damit wir die Meßkarte vom Programm aus neu konfigurieren können und Kanäle zu und wegschalten können. Da der Kanal vom PC zum MC nur für Kontrollkommunikation genutzt wird, können hier Befehle Byte für Byte gesendet werden.
Der zugehörige Befehlssatz findet sich Abbildung 4.5. Der Kanal vom MC zum PC wird auch für Nutzdaten genutzt, so daß eine Auftrennung in Befehls- und Datenmodus (siehe

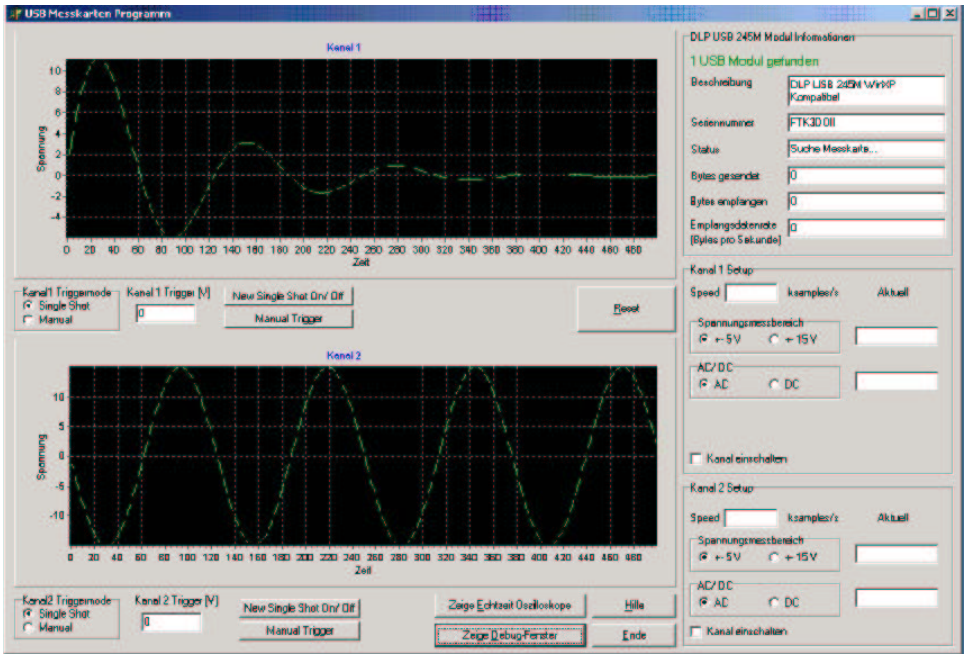


Abbildung 4.3: Oberfläche USB-Programm

Zustandsdiagramm in Abbildung 4.4) notwendig wird. Die Umschaltung geschieht mit einer Präambel und einem Nachlauf. Diese bestehen jeweils aus 40 Bit und werden so nur mit einer vernachlässigbaren Wahrscheinlichkeit in den Nutzdaten auftreten.

Zusätzlich wurden dem USB Programm noch ein Debug Formular (Bild 4.7) und ein Echtzeit-Oszilloskop (Bild 4.6) hinzugefügt. Das Debug-Formular zeigt die Kommunikation in beiden Richtungen Byte für Byte an und kann die Messkarte simulieren. Diese Features haben wir genutzt, um die Kontrollkommunikation ohne den MC zu testen, da dieser zeitweise nicht verfügbar war.

Das Echtzeit-Oszilloskop benutzt wesentlich einfachere Graphikroutinen und kann so ohne Probleme 500 kSamples/s darstellen. Auf dem Hauptformular wird eine sogenannte *TeeChart Komponente* eingesetzt, die wesentlich mehr kann (Skalierung, Interpolation, Zooming, etc...) aber dafür die Daten nicht in Echtzeit darstellen kann. Zusätzlich kann das Echtzeitoszilloskop alle empfangenen Daten darstellen, egal ob Befehl oder Daten, fürs Debuggen.

4.3.3 Testen der PC-Meßkartensoftware mit MC

Wir haben gemeinsam mit der Mikrocontroller-Gruppe das USB Modul auf der MC Platine (siehe Abschnitt 5.1) in Betrieb genommen, und erste Versuche mit der Kontrollkommunikation zwischen USB Modul und MC unternommen. Der MC schickt beim Einschalten eine Initialisierungssequenz (**0A**), woraufhin der PC mit einem Acknowledgement (**A0**) antwortet. Dadurch schaltet der MC in den USB Modus, sonst schaltet er in den Single-Modus. Danach wurde die Kontrollkommunikation nach und nach soweit ausgebaut, daß die FET Schalter bzw. Relais auf der Analogplatine geschaltet werden können.

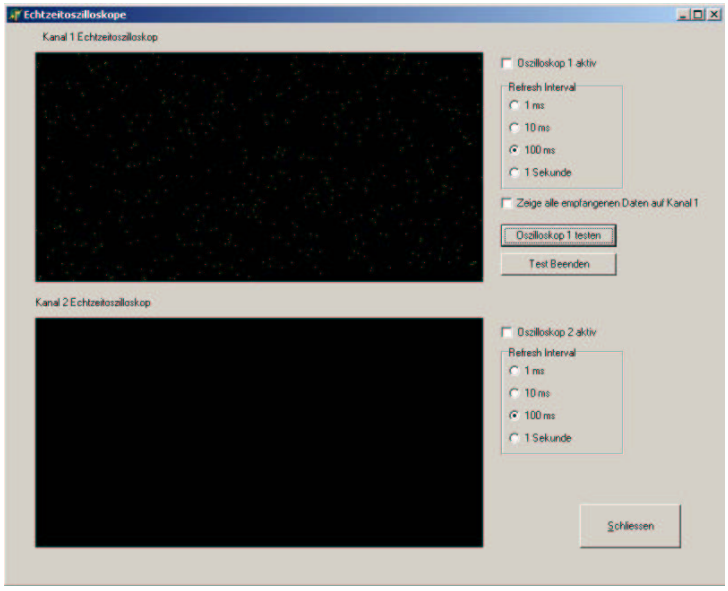


Abbildung 4.6: Echtzeitoszilloskop

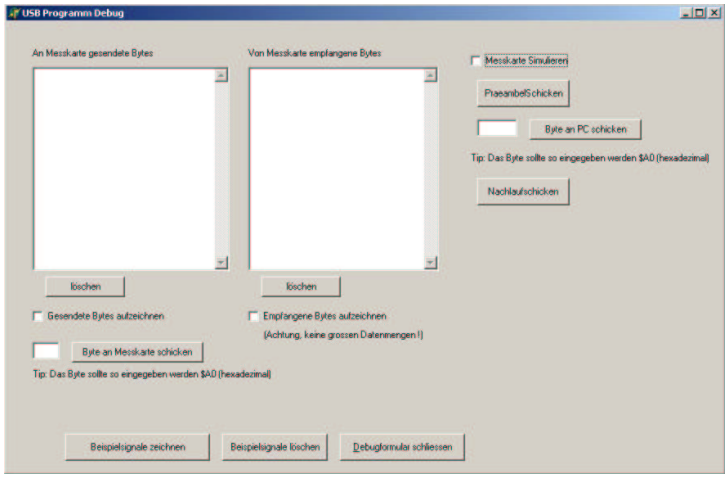
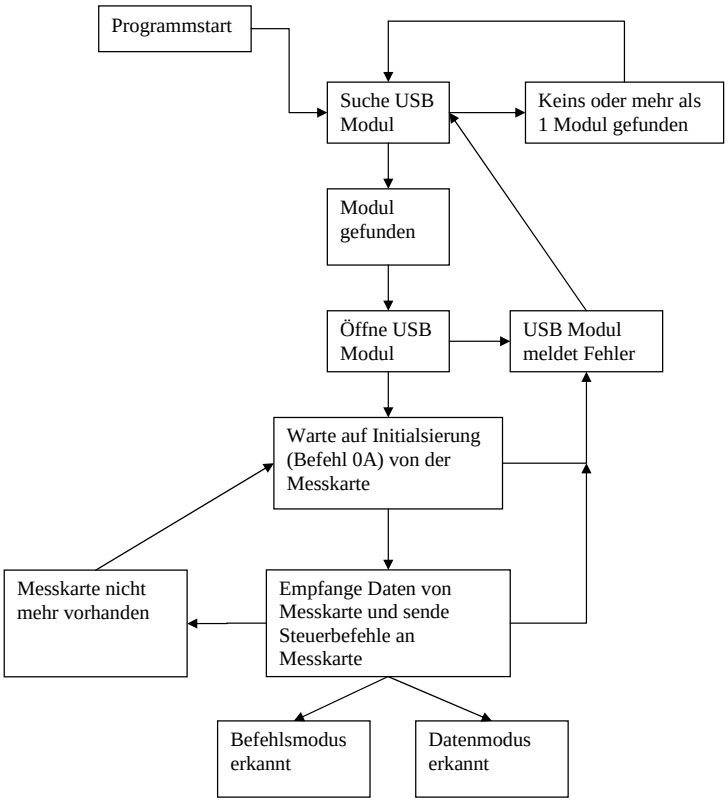


Abbildung 4.7: Debugformular

Zustandsdiagramm USB Programm



Kanalnutzung Microcontroller nach PC

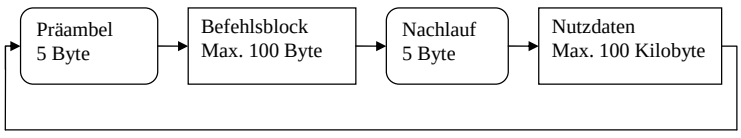


Abbildung 4.4: Zustandsdiagramm USB Modul mit Kanalnutzung

Steuerbefehle PC nach Microcontroller, per Interrupt vom MC entgegengenommen			
	Bytewert		
Hexadezimal	**Dezimal**	**Binär**	**Bedeutung**
A0	160	10100000	Antwort auf Initialisierungsanfrage 0x0A vom MC
10	16	00010000	Kanal 1 Einschalten
11	17	00010001	Kanal 1 Spannungsmessbereich + - 5 V
12	18	00010010	Kanal 1 Spannungsmessbereich + - 15 V
13	19	00010011	Kanal 1 Spannung DC Messung
14	20	00010100	Kanal 1 Spannung AC Messung
15	21	00010101	Kanal 1 Strommessung
1F	31	00011111	Kanal 1 Ausschalten
20	32	00100000	Kanal 2 Einschalten
21	33	00100001	Kanal 2 Spannungsmessbereich + - 5 V
22	34	00100010	Kanal 2 Spannungsmessbereich + - 15 V
23	35	00100011	Kanal 2 Spannung DC Messung
24	36	00100100	Kanal 2 Spannung AC Messung
2F	47	00101111	Kanal 2 Ausschalten
F0	240	11110000	Reset der Messkarte
Steuerbefehle Microcontroller nach PC, im Datenstrom mindestens alle 100 kilosamples			
	Bytewert		
Hexadezimal	**Dezimal**	**Binär**	**Bedeutung**
0	0	00000000	Präambel, Byte 1
FF	255	11111111	Präambel, Byte 2
AA	170	10101010	Präambel, Byte 3
FF	255	11111111	Präambel, Byte 4
0	0	00000000	Präambel, Byte 5
FF	255	11111111	Nachlauf, Byte 1
0	0	00000000	Nachlauf, Byte 2
55	85	01010101	Nachlauf, Byte 3
0	0	00000000	Nachlauf, Byte 4
FF	255	11111111	Nachlauf, Byte 5
0A	10	00001010	Anfrage an PC, um herauszufinden, ob USB Teil angeschlossen ist
30	48	00110000	Nach dem Nachlauf folgen keine Nutzdaten
31	49	00110001	Nach dem Nachlauf folgen Nutzdaten nur für Kanal 1
32	50	00110010	Nach dem Nachlauf folgen Nutzdaten nur für Kanal 2
33	51	00110011	Nach dem Nachlauf folgen Nutzdaten für Kanal 1 und 2, Byte für Kanal 1 zuerst
70	112	01110000	ACK Kanal 1 Einschalten
71	113	01110001	ACK Kanal 1 Spannungsmessbereich + - 5 V
72	114	01110010	ACK Kanal 1 Spannungsmessbereich + - 15 V
73	115	01110011	ACK Kanal 1 Spannung DC Messung
74	116	01110100	ACK Kanal 1 Spannung AC Messung
75	117	01110101	ACK Kanal 1 Strommessung
7F	127	01111111	ACK Kanal 1 Ausschalten
80	128	10000000	ACK Kanal 2 Einschalten
81	129	10000001	ACK Kanal 2 Spannungsmessbereich + - 5 V
82	130	10000010	ACK Kanal 2 Spannungsmessbereich + - 15 V
83	131	10000011	ACK Kanal 2 Spannung DC Messung
84	132	10000100	ACK Kanal 2 Spannung AC Messung
8F	143	10001111	ACK Kanal 2 Ausschalten
Beispiel einer Kommunikation			
MC->PC			
00 FF AA FF 00 (Präambel) 0A (PC Vorhanden ?) FF 00 55 00 FF (Nachlauf)			
PC->MC			
A0 (ja, ich bin da)			

Abbildung 4.5: Kontrollkommunikation USB Modul

Kapitel 5

Mikrocontroller (Digitalteil)

Team: PETER KORTMANN und CHRISTIAN RICHTER

5.1 Schaltung

Das Herz des Meßgeräts bildet der Mikrocontroller ***AT90S8515*** der Firma ATMEL. Er dient dazu, den Analog-Digital-Umsetzer, das USB-Modul, den Analog-Teil (Tiefpaß-Knickfrequenzen, Meßbereichseinstellungen usw.) und das LC-Display zu steuern. Außerdem nimmt er Bedienbefehle des Benutzers über vier Taster entgegen und setzt diese entsprechend um.

5.1.1 Der Atmel Mikrocontroller

Der Atmel ***AT90S8515*** basiert intern auf der Harvard RISC-Architektur und besitzt 8KByte Flash-Programmspeicher, 512 Bytes SRAM sowie 512 Bytes EEPROM Speicher, der durch den Controller selbst programmiert werden kann und so beispielsweise Einstellungen permanent speichern kann.

Der Mikrocontroller arbeitet mit einer Taktfrequenz von bis zu $8MHz$ und erreicht dabei im Optimalfall eine maximale Leistung von 8MIPS (Million Instructions Per Second). Außerdem erhält er 32 multifunktionale I/O Pins mit Tri-State Ausgängen, die einzeln als Ausgang oder Eingang selektiert werden können.

5.1.2 Das Bussystem

Abbildung 5.1: Schematische Darstellung der Mikrocontroller-Schaltung

Ein großes Problem aller Mikrocontroller ist die beschränkte Zahl der I/O-Pins. Auch wenn 32 zunächst nach einer kaum auszuschöpfenden Zahl klingt, so zeigt sich bei näherer Betrachtung doch, daß man ohne besondere Maßnahmen keine Chance hat, alle benötigten Baugruppen anzuschließen. Man halte sich nur vor Augen, welche Leitungen alles benötigt werden:

- 8 Pins — Eingang für Daten aus dem Analog-Digital Wandler
- 8 Pins — Datenausgangsleitungen zum USB-Interface

- 6 Pins — Steuerleitungen für den ADU
- 4 Pins — Kanalauswahlleitungen für den ADU
- 4 Pins — Steuerleitungen für den USB-Chip
- 12 Pins – Schaltleitungen für den Analog-Teil
- 8 Pins — Datenleitungen für das LC-Display
- 2 Pins — Steuerleitung für das LC-Display
- 4 Pins — Eingangsleitung für die Taster

Man würde also mindestens 56 I/O-Pins benötigen, um alles anzuschließen. Eine Möglichkeit den Pincount zu verringern wäre es, die Daten seriell vom ADU entgegen zunehmen und auch seriell an das USB-Interface zu liefern. Dies ist jedoch außerordentlich ungünstig, da das USB-Interface (8Bit-)parallele Daten erwartet, die dann erst extern aus dem seriellen Datenstrom erzeugt werden müßten.

Abgesehen davon würde ein serielles Abholen der Daten vom ADU nur äußerst geringe Meßbandbreiten ermöglichen. Geht man von der (sehr optimistischen) Annahme aus, das der Atmel zum Abholen eines jeden Datums nur $n = 5$ Takte benötigt, so ergibt sich bei einer Auflösung von $b = 8Bit$ und einer Taktfrequenz von $f_T = 8MHz$ lediglich eine Geschwindigkeit von

$$R_{Sample} = \frac{f_T}{nb} = \frac{8 \cdot 10^6 Hz}{5 \cdot 8} = 200kSamples/s$$

für alle Kanäle gemeinsam. Müssen sich 2 Kanäle diese Übertragungsrate teilen, erhält man für jeden Kanal nur $100kSamples/s$, bei der maximalen Ausbaustufe von 4 Kanälen sogar nur die Hälfte.

Die von uns gewählte Methode, die oben genannten Probleme mit den I/O-Pins in den Griff zu bekommen ist die Benutzung einer Bus-Architektur, wie in Bild 5.1 gezeigt.
Alle Baugruppen wie der ADU, der USB-Chip und das LCD werden über 8Bit-Transparentlatches, also I/O Demultiplexer, an den Bus angeschlossen, welche einzeln vom Mikrocontroller angesteuert werden können. Sollen nun Daten an eine dieser Komponenten übergeben werden so legt der Mikrocontroller kurz die dem Zustand der Steuerleitungen entsprechende Bitmaske auf den Bus und triggert das entsprechende Latch, welches die Daten daraufhin übernimmt und der Komponente zur Verfügung stellt.

Auf diese Weise ist es möglich, mit nur 27 Pins auszukommen. Die genaue Pin-Belegung des Atmel ist in Tabelle 5.1 zu finden.

Die Benutzung eines Datenbusses hat einen zusätzlichen Vorteil. Die Meßdaten können über den Bus direkt an den USB-Controller übergeben werden. Dieses Verfahren entlastet den Mikrocontroller, weil er nur die Steuerleitungen des ADU und des USB-Controllers kontrollieren muß, die aus Effizienzgründen direkt an den Microcontrollerpins hängen und nicht über Latches angekoppelt sind. Er ist dadurch nicht mit mit dem ununterbrochenen Durchschleusen der Daten beschäftigt und kann in der Zwischenzeit andere Aufgaben erledigen.

Tabelle 5.1: Belegung der ATMEL I/O-Pins

Pin	Port	Anz	Signalname	Dir	Bedeutung
21…28	PC0…PC7	8	D0…D7	I/O	Datenbus
32	PA7	1	A_RD#	OUT	ADU-Steuerleitung
33	PA6	1	A_CS#	OUT	ADU-Steuerleitung
34	PA5	1	A_CSTART#	OUT	ADU-Steuerleitung
35	PA4	1	A_FIRSTDATA	IN	ADU-Steuerleitung
36	PA3	1	A_BUSY	IN	ADU-Steuerleitung
37	PA2	1	A_EOC	IN	ADU-Steuerleitung
10	PD0	1	RD#	OUT	USB-Steuerleitung
11	PD1	1	WR	OUT	USB-Steuerleitung
12	PD2	1	TXE#	IN	USB-Steuerleitung
13	PD3	1	RXF#	IN	USB-Steuerleitung
17	PD7	1	LCD-RS	OUT	LCD Moduswahl (Befehl/Daten)
5	PB4	1	T0	I/O	Taster 0
6	PB5	1	T1 / AVR1	I/O	Taster 1 / ISP-Programmierleitung
7	PB6	1	T2 / AVR9	I/O	Taster 2 / ISP-Programmierleitung
8	PB7	1	T3 / AVR7	I/O	Taster 3 / ISP-Programmierleitung
14	PD4	1	ANA2-STORE	OUT	Steuerleitung Analog-Latch 2
15	PD5	1	ANA1-STORE	OUT	Steuerleitung Analog-Latch 1
18	PD6	1	LCD-ENABLE	OUT	Steuerleitung LCD
38	PA1	1	ADC-STORE	OUT	Steuerleitung ADC-Latch

5.1.3 Das LC-Display

Als LCD stand uns das **LM40X2IA** (Bild 5.2) der Firma SHARP zur Verfügung, welches netterweise von Peter Kortman für dieses Projekt gesponsort wurde. Dieses hintergrundbeleuchtete Dotmatrix-Display bietet mit zwei Zeilen zu je 40 Zeichen genügend Platz, um sämtliche Informationen darauf unterzubringen.

Abbildung 5.2: LC-Display

In der oberen Zeile ist der Status des Gerätes zu erkennen. Für jeden einzelnen der (potentiell) 4 Kanäle wird – neben der Kanalnummer – der Meßbereich (5V oder 15V) und die Kopplung (AC oder DC) angezeigt. Eine Änderung der Einstellung (durch die Taster) bezieht sich immer auf den aktuell selektierte Kanal, welcher durch einen kleinen Pfeil vor der Kanalnummer hervorgehoben ist.
Wenn sich das Gerät im Single-Modus befindet (näheres über Single- und USB-Modus im Abschnitt 5.3) wird gleich unter dem Kanalstatus der dazugehörige Meßwert in Volt angezeigt. Abgeschaltete Kanäle werden durch ein **==OFF==** an dieser Stelle gekennzeichnet.

Das Display wird durch einen 8 Bit breiten, parallelen Datenbus angesteuert, und läßt sich somit ideal in unsere Schaltung integrieren. Näheres zur Anschluß des Displays ist im Abschnitt 5.2.4 zu finden.

5.1.4 Die Taster

Die Steuerung des Gerätes im Single-Mode erfolgt, wie bereits erwähnt, über vier Taster (Abbildung 5.3).

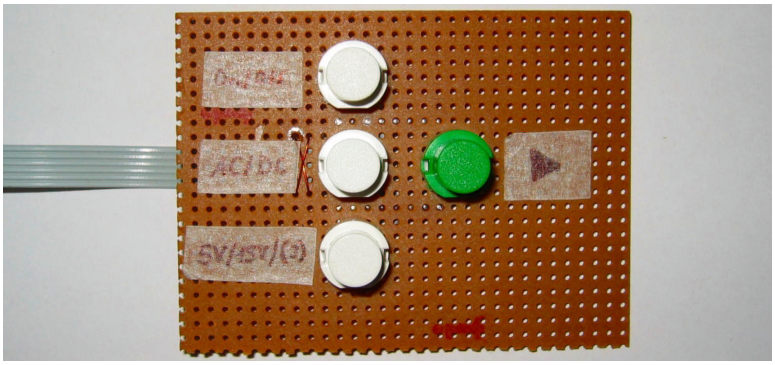


Abbildung 5.3: Platine Taster

Der grüne Taster wechselt den gerade selektierten Kanal, während, die anderen drei weißen Taster zum Manipulieren der Kanaleinstellungen benutzt werden.

Der oberste Knopf schaltet den aktuellen Kanal ein bzw. aus, der mittlere wechselt zwischen AC- und DC-Kopplung und der untere schaltet die Meßbereiche.

5.1.5 Hauptplatine

Da es sich organisatorisch als äußerst schwierig erwiesen hätte, die Schaltungen des Analogteils und des ADC-Teils gleich mit in die Unsere zu integrieren haben wir uns entschlossen, den Digitalteil auf einer getrennten Platine (Bild 5.4) aufzubauen, welche nur den Microcontroller und den USB-Chip, sowie das in Abschnitt5.1.2 angesprochene Bussystem enthält. Für die Verbindung zu den anderen Baugruppen sind Stecksockel vorgesehen. Die genauen Schnittstellen für diese Verbindungen wurden ja bereits in Abschnitt 5.2 detailiert beschrieben.

Dieses Vorgehen hat den zusätzlichen Vorteil, daß sich das Testen des Digitalteils sehr vereinfacht, da nur die momentan benötigten Baugruppen angeschlossen werden müssen und sich an den Steckverbindern die momentan anliegenden Signale sehr gut messen und überprüfen lassen. Tatsächlich war die ürsprüngliche Platine hauptsächlich zum Testen des Mikrocontroller-Teils entworfen worden, sie ließ sich aber glücklicherweise mit nur minimalen Änderungen in das fertige Gerät übernehmen.

Um die Softwareentwicklung zu vereinfachen und das schnelle und unkomplizierte Debuggen des Mikrocontrollercodes (siehe Abschnitt 5.3) zu ermöglichen, wurde außerdem ein In-System-Programmiersockel (ISP) mit auf der Platine integriert. Dieser mach die Demontage und Programmierung des Atmels in einem externen Gerät überflüssig, was sich sehr positiv auf die Entwicklungsdauer auswirkt.

Die komplette Schaltung des Digitalteils ist in Abbildung 5.5 zu sehen. Wie bereits in der schematischen Zeichnung in Bild 5.1 angedeutet, sind die einzelnen Schaltungskomponenten über Latches (ANA1-Latch, ANA2-Latch, ADC-Latch) an den Datenbus (D[0..7]) angekoppelt.
Beim Analog-Digital-Umsetzer werden dadurch die Steuerleitungen S0 bis S3 geschaltet,

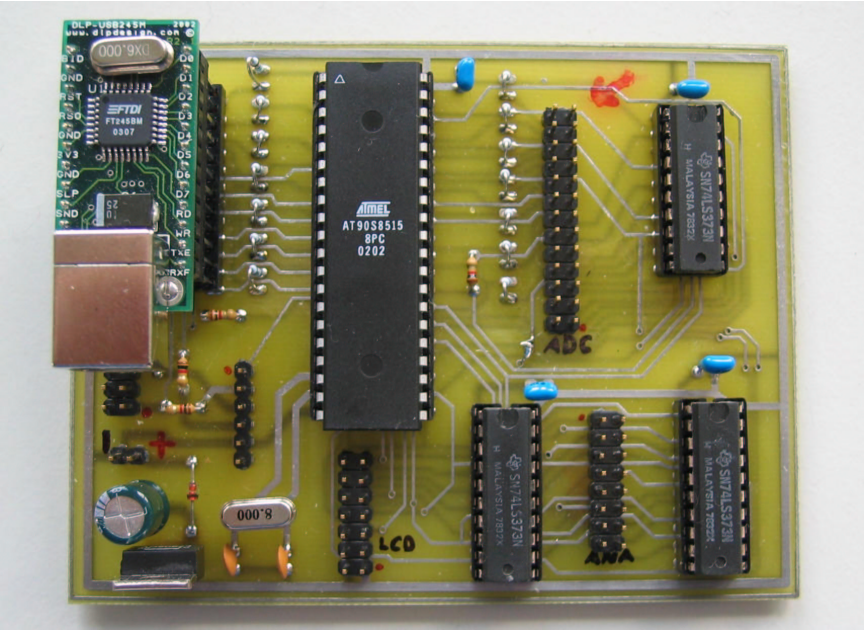


Abbildung 5.4: Fertig aufgebaute Hauptplatine

die das Ein- bzw. Ausschalten der einzelnen Kanäle steuern. Sämtliche anderen Steuerleitungen des Chips sind aus Geschwindigkeitsgründen direkt an den Mikrocontoller angeschlossen.
Die ANA1/2-Latches schalten die 16 Steuerleitungen für den Analogteil, während die vier Steuerleitungen des USB-Chips wieder direkt am Atmel angeschlossen sind.

Das LC-Display benötigt kein eigenes Latch, da es das am Datenbus anliegende Bitmuster beim Schalten der LCD_ENABLE-Leitung übernimmt und selbst speichert. Das Schalten der anderen Latches erfolgt über die Leitungen ANA1-STORE, ANA2-STORE und ADC-STORE.

Die vier Taster (T0 bis T3) teilen sich die Pins mit den Programmierleitungen für die ISP-Schnittstelle. Im Nachhinein betrachtet war dies eine etwas unkluge Design-Entscheidung da es dadurch nötig wurde, das ISP-Kabel nach jedem Programmiervorgang abzuziehen, um ordnungsgemäß mit den Tastern arbeiten zu können.

Zur stabilen und verpolungssicheren Spannungsversorgung der Schaltung dient ein 7805 Spannungsstabilisator nebst eines Stützkondensators und einer Schutzdiode.

Das fertige Board (Abbildung 5.6) ist komplett auf einer halben Euro-Platine untergebracht, um zum Entwurf die eingeschränkte Light-Version des Layouters EAGLE von Cadsoft nutzen zu können. Leider erforderte das eine relativ geringe Leitungsbreite von nur 16 mil, so daß die Platine in der Technischen Fachhochschule gefertigt werden mußte, was eine nicht unbedeutende Verzögerung zur Folge hatte.

5.1.6 Benötigte Bauelemente

Hier kommt die Partlist hin ...

5.2 Schnittstellen

Da der Mikrocontroller mit allen Komponenten kommunizieren muß, ist es besonders wichtig, die Schnittstellen zum Rest der Schaltung genau festzulegen und sorgfältig zu implementieren.

5.2.1 Analog-Digital-Wandler

Der Analog-Digital-Umsetzer sitzt auf der ADC-Platine, welche über einen 26poligen Pfostenstecker (2x13 Pins, nicht abgewinkelt) mit der Microcontroller-Platine verbunden ist. Die genaue Anordnung, sowie die Belegung der Pins und die Bedeutung der High- bzw. Low-Pegel ist aus Bild 5.7 zu entnehmen.

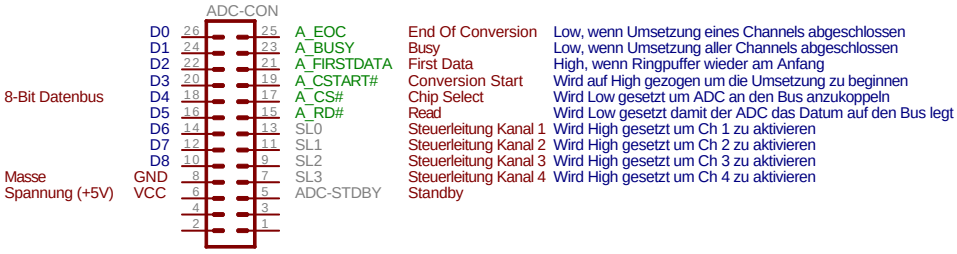


Abbildung 5.7: Schnittstellenspezifikation μC < -> ADC

Wie deutlich zu erkennen ist, werden über diesen Stecker sowohl der *Datenbus* (blau) als auch die *Steuerleitungen* (grün und grau) geführt. Außerdem versorgt die Microcontroller-Platine die ADC-Platine noch mit einer stabilisierten *Betriebsspannung* von $\pm 5V$ und mit *Masse*.

Die Datenbus-Pins D0 bis D7 sind auf Microcontroller-Seite direkt mit dem bereits in Abschnitt 5.1.2 beschriebenen Bussystem verbunden. Auf der ADC-Seite sind dort die 8 **höchstwertigen** Bits angeschlossen.

Die grün gekennzeichneten Steuerleitungen sind aus Geschwindigkeitsgründen direkt an den Atmel-μC angeschlossen. Besonders wichtig für die Steuerung der Umsetzung sind dabei die Leitungen CSTART (Conversion Start) und EOC (End Of Conversion). Mit Conversion Start wird das Signal zur Umsetzung des nächsten Kanals gegeben. Sobald diese beendet ist und ein gültiges Datenbyte zur Verfügung steht, wird das durch ein Low auf der EOC-Leitung signalisiert. Durch ein Low setzen der CS (Chip Select) und RD (Read) Leitungen wird dieses Byte dann auf den Datenbus gelegt. Bild 5.8 zeigt den entsprechenden Ablauf als Timing-Diagramm.

Welche Kanäle beim Toggeln der Conversion Start Leitung nacheinander umgesetzt werden, wird durch die grau gekennzeichneten Steuerleitungen S0 bis S3 bestimmt. Liegt die Leitung auf High, ist der Kanal aktiv und sein Sample-Wert steht zum entsprechenden Zeitpunkt zur Verfügung. Ist sie Low, bleibt der entsprechende Kanal abgeschaltet und wird nicht berücksichtigt. Diese Steuerleitungen sind nur indirekt, über das ADC-Latch an den Atmel angeschlossen, da sich der Status der Kanäle (On/Off) nur hin und wieder ändert.

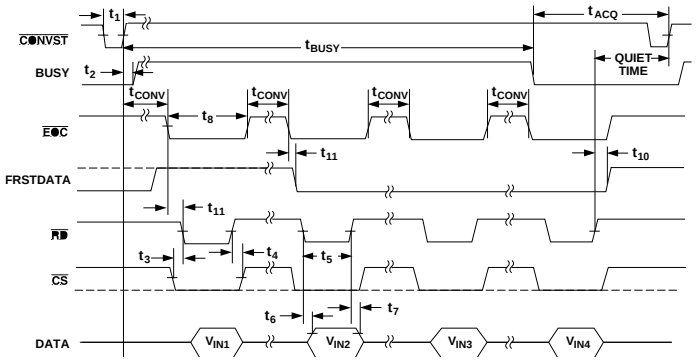


Abbildung 5.8: Ausschnitt Timingdiagramm ADC

Durch Low setzen der Standby-Leitung (ADC-STDBY) wäre es theoretisch möglich, den ADC in einen stromsparenden Ruhemodus zu versetzen. In diesem würde er dann nur noch etwa 5μA Strom (typisch) benötigen. Dieses Feature wird von uns jedoch nicht genutzt, die Standby-Leitung ist auf unserem Microcontrollerboard fest mit der 5V Betriebsspannung verbunden und liegt deshalb dauerhaft auf High.
Die Ausnutzung dieses Standby-Zustandes wäre eine Möglichkeit für einen zukünftigen Ausbau des Meßgerätes, welches dann vielleicht komplett über die USB-Schnittstelle versorgt werden könnte.

5.2.2 Analog-Teil

Auf der Analog-Platine geht es für den Microcontroller darum Relais zu schalten, um so für jeden Kanal z.B. Spannungsteiler (Meßbereich) und Koppelkondensatoren (AC/DC) in den Signalweg zu schalten, oder die Knickfrequenzen der Tiefpaßflter (Kanalbandbreite) zu ändern. Dafür ist ein dauerhaft anliegender Signalpegel nötig, weshalb diese Leitungen über zwei Latches (ANA1-Latch und ANA2-Latch) an den Mikrocontroller angekoppelt sind.

Abbildung 5.9: Schnittstellenspezifikation μC < -> Analog-Board

Die Verbindung zur Analog-Platine erfolgt über einen 16-poligen Pfostenstecker, der in Abbildung 5.9 dargestellt ist. Die linke Seite des Steckers (gerade Pin-Nummern) werden durch das ANA1-Latch, die rechte Seite (ungerade Pin-Nummern) durch das ANA2-Latch geschaltet.

Für jeden der vier Kanäle existieren drei Leitungen:

5/15V steuert den Meßbereich des Kanals. Ist sie Low, so ist er auf 15V festgelegt, bei High sind es 5V

AC/DC bestimmt die Kopplung. High bedeutet AC, Low bedeutet DC

FREQ stellt die Knickfrequenz des Tiefpasses ein, der die Anti-Alias Filterung für diesen Kanal übernimmt. High bedeutet hohe Knickfrequenz (maximale Bandbreite, nur ein Kanal aktiv), während Low eine niedrige Knickfrequenz (halbierte Bandbreite, zwei Kanäle gleichzeitig aktiv) bedeutet.[1]

5.2.3 USB-Modul (PC)

!!!!!!!!!!!! mache ich ...

5.2.4 LC-Display

Das LC-Display ist, wie bereits in Abschnitt 5.1.3 erwähnt, direkt und ohne Latch an den Datenbus der Microcontrollerplatine angeschlossen. Dies ist möglich, da es selbst über eingebaute Datenregister verfügt und die Steuerzustände daher (im Gegensatz z.B. zur Analog-Schnittstelle nicht dauerhaft gehalten werden müssen.
Es genügt also, kurz das Bitmuster eines ASCII-Zeichens auf den Bus zu legen und den Enable-Eingang (LCD-ENABLE) auf High zu setzen. Das am Bus anliegende Zeichen wird daraufhin in ein internes Register übernommen und auf dem Display dargestellt.

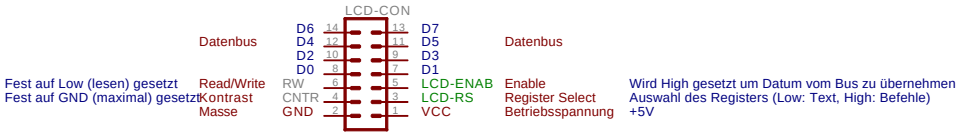


Abbildung 5.10: Schnittstellenspezifikation μC < -> LCD

Die Belegung des 14-poligen Verbindungssteckers ist aus Bild 5.10 zu ersehen. Die blau gekennzeichneten Leitungen D0 bis D7 gehören auch hier wieder zum Datenbus. Auch wird, genau wie beim der ADC-Schnittstelle in Abschnitt 5.2.1 wieder die Masse und eine Betriebsspannung von $+5V$ (rot) übergeben.

Entscheidend sind hier die beiden grün dargestellten Pins LCD-ENAB (Enable) und LCD-RS (Register Select), die direkt an den Microcontroller angeschlossen sind.
Sobald die Enable-Leitung auf High gesetzt wird, übernimmt das LCD das momentan am Bus anliegende Datenbyte in ein Zwischenregister. In welchem, das entscheidet die Register-Select-Leitung. Ist sie High, so wird das anliegende Byte in das Befehlsregister gelesen und dort als Steuerbefehl (lösche Bildschirm, bewege Cursor usw.) interpretiert. Ist sie dagegen Low, so landet das Byte im Datenregister und wird von dort aus aufs Display geschrieben.

Die Read/Write-Leitung entscheidet ob das Display Daten vom Bus liest (und darstellt) oder in den Registern gespeicherte Daten ausgibt. Da das LCD in dieser Anwendung ausschließlich lesen soll, wird sie in unserer Schaltung fest auf Low (Ground) gelegt.

Ähnlich Kontrast-Leitung (CNTR). Die dort anliegende Spannung ($0V - 5V$) bestimmt den Kontrast des Displays. Wir haben sie fest auf Masse (maximaler Kontrast) gelegt.

[1] Für die momentan realisierten 2 Kanäle genügt diese 1-Bit Kodierung. Bei einem Ausbau mit 4 Kanälen muß man die momentan noch unbenutzten Pins 13-16 hinzunehmen.

5.2.5 Taster

Die Taster sind über einen 5-poligen, diesmal ausnahmsweise einreihigen, Pfostenstecker mit dem Atmel verbunden. Neben den vier für die Taster nötigen Leitungen, wird außerdem die Masse noch mitgeführt.

Abbildung 5.11: Schnittstellenspezifikation μC < -> Taster

Wird nun durch Drücken eines Schalters eine leitende Verbindung zwischen der Taster-Leitung und Masse hergestellt, diese also auf Low gelegt, so gilt der dazugehörige Schalter als betätigt.

Da die letzten drei der insgesamt vier Taster sich die Mikrocontroller-Eingänge mit der ISP-Programmierschnittstelle teilen, sind diese noch zusätzlich mit $1k\Omega$ Widerständen abgesichert, um Kurzschlüsse zu vermeiden.

Die Zuordnung der Pins zu den einzelnen Schaltern ist in Bild 5.11 zu sehen. Eine genauere Beschreibung, was ein Druck auf die einzelnen Taster bewirkt findet sich im Abschnitt 5.1.4.

5.3 Steuerprogramm

Die Initialisierung nach dem Einschalten, die Ansteuerung des LCD, der Latches, des USB-Moduls, des AD-Wandlers und die Übergabe der Daten an den PC über die USB-Schnittstelle wurde von dem Steuerprogramm für den Mikrokontroller erledigt. Da auf dem PCB eine ISP-Schnittstelle implementiert wurde, konnte die Software direkt in den Atmel Mikrocontroller geflasht und sofort getestet werden. Der Mikrokontroller mußte für diesen Zweck nicht aus dem Board entfernt werden. Diese Tatsache erleichterte und beschleunigte den Entwicklungs- und Erprobungsprozeß ganz erheblich.

Die Software selbst wurde im Assembler geschrieben. Die Entwicklung in einer höheren Programmiersprache wie C oder C++ wäre zwar einfacher gewesen, impliziert jedoch einen entscheidenden Nachteil: Das Timing kann nicht genau berechnet werden, da das komplexe Verhalten des Compilers nicht vorhersehbar ist. Da in der Spezifikation eine hohe Samplingrate von ca. 500 KSamples/s als Ziel definiert war, mußte das Programm außerdem noch auf Geschwindigkeit optimiert werden, was sich am besten direkt im Assembler erreichen läßt.

5.3.1 Struktur des Programmes

Nach dem Einschalten der Betriebsspannung wird über einem Pull-Up Widerstand der Kondensator an der negierten *RESET*-Leitung auf TTL-High (5V) aufgeladen. Diese Verzögerung soll den Betrieb des Mikrokontrollers während der von Spannungschwankungen gestörten Einschaltphase verhindern. Anschließend beginnt der Mikrokontroller mit der Abarbeitung des Programms.

Zunächst wird die Interruptvektortabelle und der Stack initialisiert. Die Interruptvektortabelle beinhaltet die Sprungadressen für die ISR (Interrupt Service Routine). Um einen definierten Zustand der Meßkarte zu erreichen, werden anschließend die IO-Ports auf Ein- oder Ausgänge gesetzt. Alle am Bus hängenden geräte werden ebenfalls durch das Setzten oder Löschen der Strobe-Leitung vom Bus entfernt.

Es folgt nun die Initialisierung des LC-Displays. Das Display wird im 8 Bit Modus betrieben. Die Datenleitungen hängen am Bus, wobei die Steuerleitungen exklusiv durch den Mikrocontroller angesteuert werden (siehe dazu auch Abschnitt 5.2.4). Dadurch wird die LCD-Ausgabe simpel und effizient gestaltet.

Nach der Initialisierung des LCDs entscheidet die Meßkarte, ob sie im USB- oder im Single-Mode betrieben wird. Dazu wird ein Startdatenwort `0x0A` an den PC geschickt. Das Programm auf dem PC antwortet dann mit , wenn das Gerät ordungsgemäß angeschlossen ist und die USB Kommunikation problemlos verläuft.

Erhält die Meßkarte nach 500*ms* keine gültige Antwort vom PC, so wird sie im Singlemode betrieben. Eine spätere Umschaltung der Modi ist nur durch das Abschalten der Betriebsspannung möglich. Auf die beiden Modi wird im weiterem Verlauf noch näher eingegangen. Die Funktionsweise ist auch in dem in Bild 4.4 gezeigten Datenflußdiagramm visualisiert.

5.3.2 Ansteuerung des ADC

Paralellumsetzer

Urspünglich wurde ein 4-kanaliger 12 Bit-Parallelumsetzter der Firma ANALOG DEVICES verwendet. Dieser ADC ist mit bis zu 400 KSamples/s relativ schnell. Durch die Leitungen S1, S2, S3 und S4 kann der zu sampelnde Kanal ausgewählt werden. Diese Leitungen werden durch den ADC-Latch gesteuert (siehe Abschnitt 5.2.1). Will man beispielsweise den ersten und dritten Kanal umwandeln, so muß S1 und S3 auf high gesetzt werden. Nach dem `STARTCONV` Signal erhält man durch das Takten die Datenwörter vom Kanal 1 und anschließend Kanal 2. Auf diese Weise lassen sich alle Kanäle bequem per Software konfigurieren. Die `RD`-Leitung wurde zum Takten verwendet. Der Abtastvorgang kann am besten an dem Timingdiagramm (Bild 5.8) im Abschnitt 5.2.1 verstanden werden.

Der Analog-Digital-Umsetzer war auf dem Spannungsversorgungsboard integriert. Leider wurde beim Testen der Spannungsversorgung der Analogumsetzer irreparabel beschädigt, woraufhin wir versuchten, zwei einkanalige (und viel langsamere) 8-Bit Parallelumsetzer anzuschließen. Diese stellten sich jedoch leider als defekt heraus, so daß als letze Möglichkeit ein Low Cost 8 Bit Seriell-ADC (bei der Firma Conrad gekauft) herhalten mußte, welcher im nächsten Abschnitt beschrieben ist.

Serieller Umsetzer

Der **ADC TLC549** der Firma TEXAS INSTRUMENTS kann nur positive Spannungen im Bereich von $0V - 5V$ in ein 8 Bit Datenwort umwandeln, welches dann seriell ausgeben wird. Die Taktung erfolgt durch die `CLK` Leitung. Kostenbedingt (der Preis lag bei 2 EUR) ist dieser Umsetzer leider relativ langsam. Der Mikrokontroller mußte mehrere Wartetakte einlegen, bis die Daten abgeholt werden können. Daraus ergab sich nach einigen Tests und Optimierungen eine Performance von ca 33 KSamples/s, also weniger als 1/10 des Parallelumsetzer-Geschwindigkeit.

Die Funktion `TLC549Read` liest die einzelnen Bits aus, und gibt sie im Register `r17` (Arg0) aus. Sie wird sowohl im Single, als auch im USB-Modus verwendet.

Program 5.1: Ausleseroutine für den TCL549

```
1  ; Funktion: TLC549Read
2  ;   Den ADC auslesen und eine neue Wandlung starten.
3  ;   Zwischen zwei Aufrufen min. 17us warten.
4  ; Parameter:           keine
5  ; Veränderte Register: Temp0 = Bitzähler
6  ; Rückgabe:            Arg0 = ADC-Wert
7
8  ;TLC549_PORT = porta
9  ;TLC549_PIN = pina
10 ;TLC549_DATA = 3
11 ;TLC549_CS  = 2
12 ;TLC549_CLOCK= 4
13
14 TLC549Read:
15                 cbi  porta, TLC549_CS   ; /CS auf 0
16                 rcall TLC549ReadX       ; min. 1,4us warten
17                 rcall TLC549ReadX
18                                           rcall TLC549ReadX
19                 ldi Temp0, 8            ; 8 Bits
20 TLC549Read1:
21                 clc                     ; Bit einlesen
22                 sbic PINa, TLC549_DATA
23                 sec
24                 rol Arg0
25                 sbi PORTa, TLC549_CLOCK ; CLOCK auf 1
26                 rcall TLC549ReadX       ; min. 0,5us warten
27                 cbi PORTa, TLC549_CLOCK ; CLOCK auf 0
28                 rcall TLC549ReadX       ; min. 0,5us warten
29                 dec Temp0               ; noch weitere Bits?
30                 brne TLC549Read1
31                 sbi PORTa, TLC549_CS    ; /CS auf 1
32
33 TLC549ReadX:    ret
```

Da der ADC nur im positiven Bereich wandelt, stellt das MSB kein Vorzeichen mehr dar, sondern entspricht $0,5 \cdot V_{cc} = 2,5V$. Die Umrechungroutine mußte dementsprechend angepaßt werden.

Der Seriellumsetzer hat sogut wie auf Anhieb funktioniert. Die Vorführung konnte mit diesem Bauteil am nächsten Morgen erfolgreich durchgeführt werden.

5.3.3 Single-Mode

Im Single-Mode liest das Gerät die Meßdaten aus dem ADC aus, rechnet sie in einen entsprechenden Spannungswert um und schreibt diesen als ASCII-Zeichen auf das LC-Display.

Es stehen 4 Kanäle zur Verfügung, deren Konfiguration (Kanalwahl, An/Aus, AC/DC und 5V/15V) über die Taster verändert werden kann (siehe Abschnitt 5.1.4).

Der Zustand der Tasten wird permanent gepollt. Nach einer Zusandsänderung wird der Status des Geräts auf dem LC-Display erneut dargestellt. Es sei an dieser Stelle auf den Programmcode verwiesen, der teilweise recht umfangreich dokumentiert ist. Besonders zu beachten ist hierbei die Unterroutine `check_taster`, welche die Taster abfragt.

Bei der Umrechnung der vom ADC gelieferten Daten in tatsächlich Spannungen wird als Rechenoperation eine Multiplikation benötigt. Da der Atmel ein RISC Processor ist, verfügt er nur über einen beschränkten Befehlssatz. Im Gegensatz zu vielen CISC-CPUs muß die Multiplikation daher in Software durchgeführt werden. Zu diesem Zweck wurde von uns eine Routine aus dem Internet verwendet.

Der gesamte Text wird schließlich auf dem LCD dargestellt. Um die Programmierung in Assembler zu erleichtern, wurde eine Zeichenausgaberoutine für das LCD verwendet.

Program 5.2: Makro für Zeichenausgabe auf dem LCD

```
1   .macro rs1  ;LCD-RS auf High setzten - Datenmodus
2         sbi portd,7
3   .endmacro
4
5   .macro rs0  ;LCD-RS auf Low setzten - Befehlsmodus
6         cbi portd,7
7   .endmacro
8
9   .macro clk1 ;CLK LCD auf High setzten
10        sbi portd,6
11  .endmacro
12
13  .macro clk0 ;CLK LCD auf Low setzten
14        cbi portd,6
15  .endmacro
16
17  .macro lcd_putchar ;Gibt das Zeichen in r17 auf dem LCD aus
18        out portc,r17 ;Inhalt von r17 auf den Bus legen
19        clk1
20        rcall w100us  ;100us warten
21        clk0
22        rcall w100us  ;100us warten
23  .endmacro
```

Das LCD beherrscht 2 Modi - einen Steuer/Programmmodus und einen Datenmodus. Im ersten kann die Funktionsweise des Displays angepaßt werden. In diesem Modus wird auch das Display initialisiert. Im zweiten Modus wird ein Zeichen auf das Display und in dessen RAM geschrieben. Der RAM kann auch ausgelesen werden. Diese Funktion wurde jedoch nicht verwendet, weil das Display relativ langsam ist und die Verarbeitungsgeschwindigkeit schon jetzt genügend herabsetzt. Die Daten- und Steuermodi können durch das Setzen von Bit 7 auf PORTD gesteuert werden.

Damit das Display die Daten entgegennimmt, muß es auch getaktet werden. Die Taktung wird durch die Makros `clk0` und `clk1` erzeugt. Diese setzten und löschen das Bit 6 am PORTD, wodurch die Datenübernahme ermöglicht wird.

5.3.4 USB-Mode

Ist ein PC mit der laufenden Meßsoftware an das Gerät angeschlossen, so wird dies durch ein Acknowledgement (`0xA0`) der PC-Software an die Mikrocontrollerkarte bestätigt. Diese versetzt sich dann während des Einschaltvorgangs in den USB Modus.

In dieser Betriebsart sind die Tasten des Gerätes deaktiviert. Die Einstellungen werden vom PC aus durchgeführt und über die USB-Schnittstelle an die Meßkarte geschickt. Diese wertet sie aus und setzt dementsprechend die Register, triggert die Latches und schaltet somit die Relais. Zum Schluß wird noch der Gerätestatus auf dem LCD aufgefrischt, der ja in der ersten Zeile dargestellt wird. In der zweiten Zeile werden nun keine Meßwerte angezeigt, da die Umrechung in ASCII-Zeichen relativ rechenaufwendig ist und die ständige Ausgabe auf dem LCD zu äußerst langen Wartezyklen zwingen würde. Die Übertragungsrate an den PC würde somit schlagartig sinken. Aus diesem Grund wird in dieser Zeile nur die Ausgabe

"**--------- < USB Modus aktiv > --------**"

dargestellt. Der Mikrocontroller kann sich nun voll auf die Abfrage des ADC und die Übertragung an das USB-Modul konzentrieren.

Wird ein oben genannter Steuerbefehl vom PC an die Karte geschickt, so generiert das USB-Modul einen IRQ (Interupt Request) am Mikrocontrollereingang. Wenn IRQs zugelassen sind, springt der Mikrocontroller in die ISR (Interrupt Sevice Routine), die das Byte aus dem 184 Bytes großen Empfangspuffer des USB-Moduls abholt und auswertet.

Damit die Interrupts überhaupt ausgelöst werden, müssen die dafür zuständigen Maskenregister (MCUCR, SREG, GIMSK) dementsprechend gesetzt werden, wofür folgender Code verantwortlich ist:

Program 5.3: Erlauben der Interrupts

```
1  in r18,sreg                ;Lade sreg in r18
2  ori r18,0b10000000   ;Setze bit 7 für IRQ an
3  out sreg,r18               ;schreibe nach sreg ...
4  out gimsk,r18        ;und gimsk
5  clr r18                           ;r18 komplett auf 0 setzen
6  out MCUCR,r18        ;und nach MCUCR schreiben
```

Die Routine `ISR_usbcommand` liest das Byte aus und wertet es anhand der Protokolldefinition aus. Danach wird der Gerätestatus entsprechend gesetzt und an den PC eine Bestätigung zurückgeschickt. Die Protokolldefinition aus der Tabelle in Bild 4.5 zu entnehmen.

Das USB-Modul liegt ähnlich wie das LCD direkt am den Datenbus. Für die Kommunikation wurden folgende Makros verwendet:

Program 5.4: Makros zum Auslesen des USB-Moduls

```
1  .macro usbwr1       ;USB Write Leitung auf HIGH setzen
2          sbi portd,1
3  .endmacro
4
5  .macro usbwr0       ;USB Write Leitung auf LOW setzen
6          cbi portd,1
7  .endmacro
8
9  .macro usbrd1       ;USB Read Leitung auf HIGH setzen
10         sbi portd,0 ;Damit nimmt das USB Modul die Daten WEG vom Bus
11 .endmacro
12
13 .macro usbrd0       ;USB Read Leitung auf LOW setzen
14         cbi portd,0 ;Damit legt USB Modul die Daten AUF dem Bus
15 .endmacro
16
17 .macro usb_putbyte  ;Gibt das Zeichen in r17 an das USB Modul aus
18         out portc,r17
19         sbi portd,1
20         cbi portd,1
21 .endmacro
22
23 .macro usb_getbyte  ;Liest ein Zeichen aus dem USB Modul nach r17
24         sbi portd,0
25         cbi portd,0
26         nop
27         nop
28         nop
29         nop
30         in r17,pinc
31       ;Wenn RD# high wird das Signal vom Usdmodul vom Bus genommen
32 .endmacro
```

Interessant ist hier die Funktion `usb_getbyte` in Zeile 23. Die vier NOP Befehle (***N***o ***OP***eration) in den Zeilen 26 bis 29 wurden vor dem Einlesekommando eingefügt damit das USB-Modul genügend Zeit hat, die empfangenen Daten auf den Bus zu legen.

Die Meßkarte schickt alle 100*ms* einen "Heartbeat"-Code an die PC-Software, der den funktionierenden Zustand der Karte signalisiert. Dieser Heartbeat enthält ein Kontrollsymbol das Aufschluß darüber gibt, von welchen Kanälen momentan Daten gesendet werden. Auf diese Weise kann die Zuordnung der (im Zeitmultiplex übertragenen) Meßwerte zu den entsprechenden Kanälen wieder synchronisiert werden, falls z.B. mal ein Meßwert oder ein Kommando verloren gegangen ist. Das Senden wird ebenfalls von einem Interrupt gesteuert, der nach einer gewissen Zeit automatisch ausgelöst wird. Der ATMEL 90S8515 besitzt zwei skalierbare Timer, die jeweils einen solchen Timer-Interrupt auslösen können. Folgender Code stellt den Timer ein:

Program 5.5: Initialisierung des Timer-Interrupts

```
clr r18           ; Timer1 einstellen
out TCCR1A,r18
ldi r18,0b00001101
out TCCR1B,r18       ;1024 / prescaled clock
ldi r18,0b00000011   ;780 mal zählen lassen, dann IRQ
out OCR1AH,r18
ldi r18,0b00001100
out OCR1AL,r18
ldi r18,0b01000000   ;Timer Counter Output compare enable
out TIMSK,r18
```

Zuerst wird das Vorzählwerk für den Timer eingestellt. Der Zählmodus muß ebenfalls gesetzt werden. Hier wird der Zähler nach jedem Überlauf gelöscht und fängt wieder vom Anfang zu zählen an. Diese Einstellungen werden in den Registern TCCR1B, OCR1AL und OCR1AH vorgenommen. Schließlich muß noch das Interruptmaskeregister TIMSK entsprechend gesetzt werden.

Damit die ACKs und Steuerbefehle von dem Meßdatenstrom, der am PC ankommt, unterschieden werden können, wurde bei jedem an den PC geschickten Befehl eine Preambel (`0x00ffaaff00`) und ein Nachlauf (`0xff005500ff`) mitgeschickt. Die Routine `usb_sendcommand` bettet darin automatisch das Befehlsbyte ein:

Program 5.6: Senderoutine für USB-Befehle

```
usb_sendcommand:
          push r17  ;Register auf Stack sichern

         ;Praeambel (0x00ffaaff00) senden
          usbrd1
          ldi r17,0
          usb_putbyte
          ldi r17,$ff
          usb_putbyte
          ldi r17,$AA
          usb_putbyte
          ldi r17,$FF
          usb_putbyte
          ldi r17,0
          usb_putbyte

          ;Zeichen in r17 als Befehl senden
          pop r17
          usb_putbyte

          ;Nachlauf (0xff005500ff) senden
          ldi r17,$ff
          usb_putbyte
          ldi r17,$00
          usb_putbyte
          ldi r17,$55
          usb_putbyte
          ldi r17,$00
```

```
29          usb_putbyte
30          ldi r17,$ff
31          usb_putbyte
32
33          ret
```

Der PC prüft die ankommenden Daten auf die Preambel und den Nachlauf. Werden diese erkannt, so wird das ankommende Byte vom PC als Steuerbefehl oder ACK interpretiert und die angekommenen Steuerdaten nicht als Meßdaten verwendet.

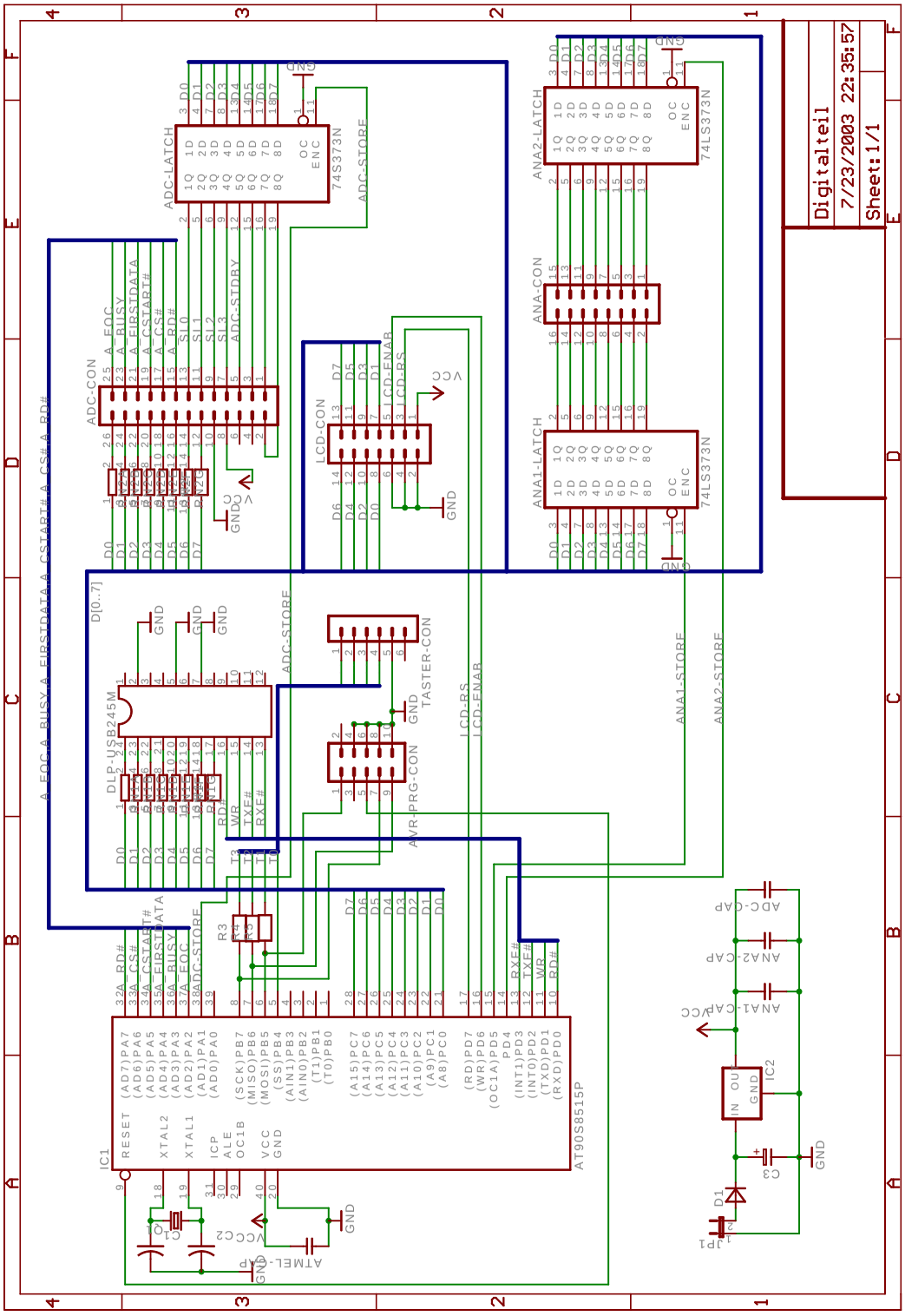


Abbildung 5.5: Schaltung der der Mikrocontroller-Platine

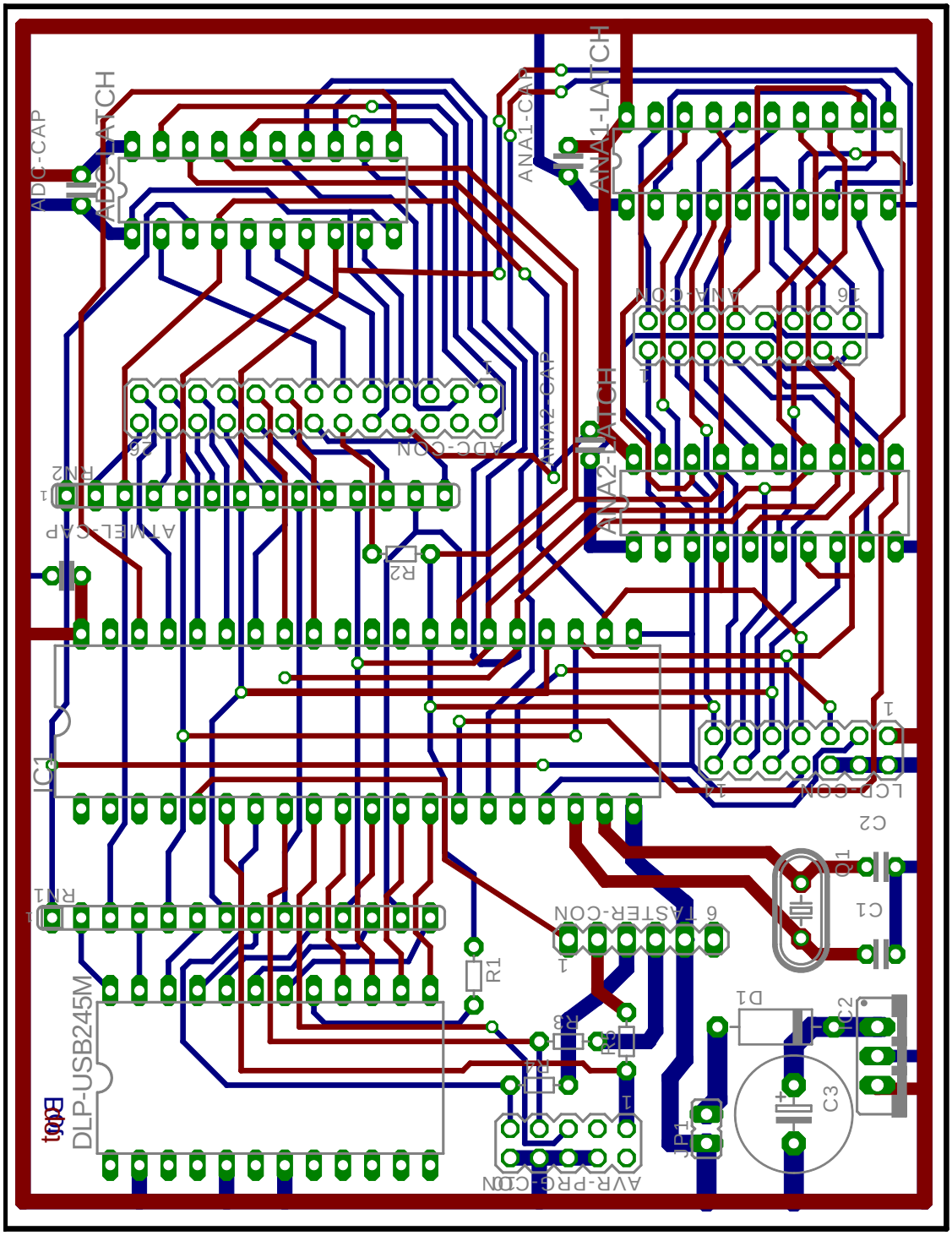


Abbildung 5.6: Layaut der Microkontroller-Platine

Kapitel 6

Fertigstellung des Projektes

6.1 Test der Gesamtschaltung

Zum Testen der Gesamtschaltung haben wir zunächst den 12-Bit Analog-Digital-Umsetzer angeschlossen. Dieser hat auch Meßwerte über den Mikrocontroller an das PC-Programm geliefert. Leider waren zunächst die 8 niederwertigen Bits des ADC statt der 8 höchstwertigen angeschlossen, so daß wir mit unserer Schaltung nur im Millivoltbereich messen konnten. Durch Vertauschen der Leitungen im Flachbandkabel zum ADC Board konnten wir dieses kleine Mißgeschick glücklicherweise schnell beheben.

Beim Testen des Analogteils traten jedoch größere Probleme auf, da die FET-Schalter am Eingang nicht für den Messbereich von $\pm 15V$ geeignet waren. Nach Versuchen, diese durch Relais zu ersetzen, gab es einen Kurzschluß, der vermutlich durch falsch geroutete Leiterbahnen, die nicht im Schaltplan erkennbar waren, ausgelöst wurde. Dadurch wurden die Filterbausteine zerstört und ein Neuaufbau der Analogplatine wurde notwendig.

Dieser Neuaufbau erfolgte am Wochenende, wobei alle FET Schalter durch Relais ersetzt wurden. Als Filterbausteine haben wir die Bauteile genommen, die ursprünglich für den Analogteil vorgesehen waren, aufgrund ihrer sehr komplizierten und aufwändigen Schaltung jedoch eigentlich nicht benutzt werden sollten.

Bei Versuchen, die Spannungsversorgung auf dem ADC Board zu optimieren, wurde dann leider der ADC-Chip zerstört, womit das eigentlich bereits funktionierende Gerät plötzlich unbrauchbar wurde. Da sich in den verbleibenden 21 Stunden bis zum Vorführtermin auch kein Ersatzchip beschaffen ließ, haben wir in der folgenden Nacht notdürftig eine neue ADC-Platine für einen seriellen 8-Bit Umsetzer (Conrad Elektronik 2 EUR) gebaut und die Ansteuerung im Mikrocontroller-Programm entsprechend angepaßt.

6.2 Vorbereitung der Präsentation

Am letzten Termin vor dem Vorführtermin haben wir eine POWERPOINT-Präsentation über unser Projekt zusammengestellt, die am Vorführtermin den Betreuern und der Logic-Analyser Gruppe vorgeführt werden sollte. Die Präsentation befindet sich – wie alle anderen Daten zum Projekt auch – auf der **Projekt-CD** unter **??????????????????????.**

6.3 Vorführtermin

Am Vorführtermin haben wir zunächst eine halbe Stunde die Präsentation der Logic Analyser Gruppe verfolgt, um dann selbst eine gute halbe Stunde unser Projekt mit Laptop und Beamer zu präsentieren. Danach wurden beide Projekte Prof. Orglmeister, den Betreuern und den Mitgliedern der anderen Gruppe präsentiert.

Kapitel 7

Arbeits- und Zeitplan

In der untenstehenden Abbildung ist eine tabellarische Übersicht über den Zeitplan des Projektes angegeben. Deadlines sind rot markiert.

KANN SICH HIER NOCH JEMAND WAS EINFALLEN LASSEN? PETER, HAST DU EINE IDEE???

Laborwoche:	1	2	3	4	5	6	7	8	9	10	11	12	13
	21. Apr	28. Apr	05. Mai	12. Mai	19. Mai	26. Mai	02. Mai	09. Jun	16. Jun	23. Jun	30. Jun	07. Jul	16. Jul
Themasuche													
Grobdesign und Lösungsvorschläge													
erste Application Notes													
Verbesserungen & Weiterentwicklung													
Abgabe des Zwischenberichtes													
1. kompletter Funktionstest													
2. kompletter Funktionstest													
Abgabe des Abschlußberichtes													
Vorführtermin													

Kapitel 8

Handelnde Personen

Jörn Hohlwein, Walid Tfayli, Omar El-Mikati, Helge Krambeck, Christian Richter, Berit Herrmann
Shpend Mirta, Peter Kortmann, Driton Emini

Abbildungsverzeichnis

1.1 Fertiges Gerät (Ausbaustufe mit 2 Kanälen) 5

2.2 nicht invertierender Verstärker 6
2.3 Blockschaltbild 7
2.6 Relaisansteuerung 7
2.7 Schutzschaltung 8
2.1 Schaltplan erste Analog-Platine 10
2.4 Erste Analog-Platine (defekt) 11
2.5 Neuer Schaltplan Analog-Platine 12

3.1 Beschaltung des ADC 14
3.2 Schaltplan Spannungsversorgung 15
3.3 Gemeinsame Platine mit ADC und Spannungsversorgung 17

4.1 Schaltplan USB nach Microcontroller Platine 19
4.2 Platinenlayout USB nach Microcontroller Platine 20
4.3 Oberfläche USB-Programm 26
4.6 Echtzeitoszilloskop 27
4.7 Debugformular 27
4.4 Zustandsdiagramm USB Modul mit Kanalnutzung 28
4.5 Kontrollkommunikation USB Modul 29

5.1 Schematische Darstellung der Mikrocontroller-Schaltung 30
5.2 LC-Display 32
5.3 Platine Taster 33
5.4 Fertig aufgebaute Hauptplatine 34
5.7 Schnittstellenspezifikation μC < -> ADC 35
5.8 Ausschnitt Timingdiagramm ADC 36
5.9 Schnittstellenspezifikation μC < -> Analog-Board 36
5.10 Schnittstellenspezifikation μC < -> LCD 37

5.11 Schnittstellenspezifikation μC < -> Taster 38
5.5 Schaltung der der Mikrocontroller-Platine 45
5.6 Layaut der Microkontroller-Platine 46

Tabellenverzeichnis

1.1 Spezifikationen 4

4.1 Geschwindigeitsmessung Microcontroller 24

5.1 Belegung der ATMEL I/O-Pins 32