Labor Computertechnik (LCT)
Laborbericht zu Versuch: 1
Logikanalysatortechnik und RS232

Andreas Hofmeier

Auftraggeber:	Prof. Dipl.-Ing. S. Myrzik, Fachhochschule Bremen
Durchführung am:	22.03.2006
Ort der Durchführung:	FH Bremen, Flughafenallee 10, Raum: 232
Abgabe am:	19.04.2006

Versuchsgruppe 1

Andreas Hofmeier	94453
Raphael Rosendahl	95437
Jakob Schröter	91150

Zusammenfassung

Dieses Dokument beschreibt die Durchführung des Versuches Nr. 1 im Labor RST LCT der Versuchsgruppe 1.

Die serielle Datenübertragung zwischen einem Mikrocontroller und einem PC wurde abgefangen und ausgewertet. Die übertragenen Zeichen wurden korrekt interpretiert. Im zweiten Teil des Versuches wurde die Datenübertragung abgefangen, als ein bestimmte Bitmuster an einem Port anstand. Das zu diesem Zeitpunkt gesendete Byte lautete ASCII(E).

Die Versuchsgruppe erlangte ein grundlegendes Verständnis über die serielle Schnittstelle sowie die Funktionsweise und den Einsatz von Logikanalysatoren.

Inhaltsverzeichnis

1 Versuchsziele

Ziele dieses Versuches sind:

1. Einen Überblick über die Funktionsweise der seriellen Schnittstelle, auch bekannt unter den Namen V24 oder RS232, zu erlangen.
2. Einen Einblick in die Messtechnik, hier im Speziellen in den Logikanalysator, zu bekommen

2 Versuchsaufbau

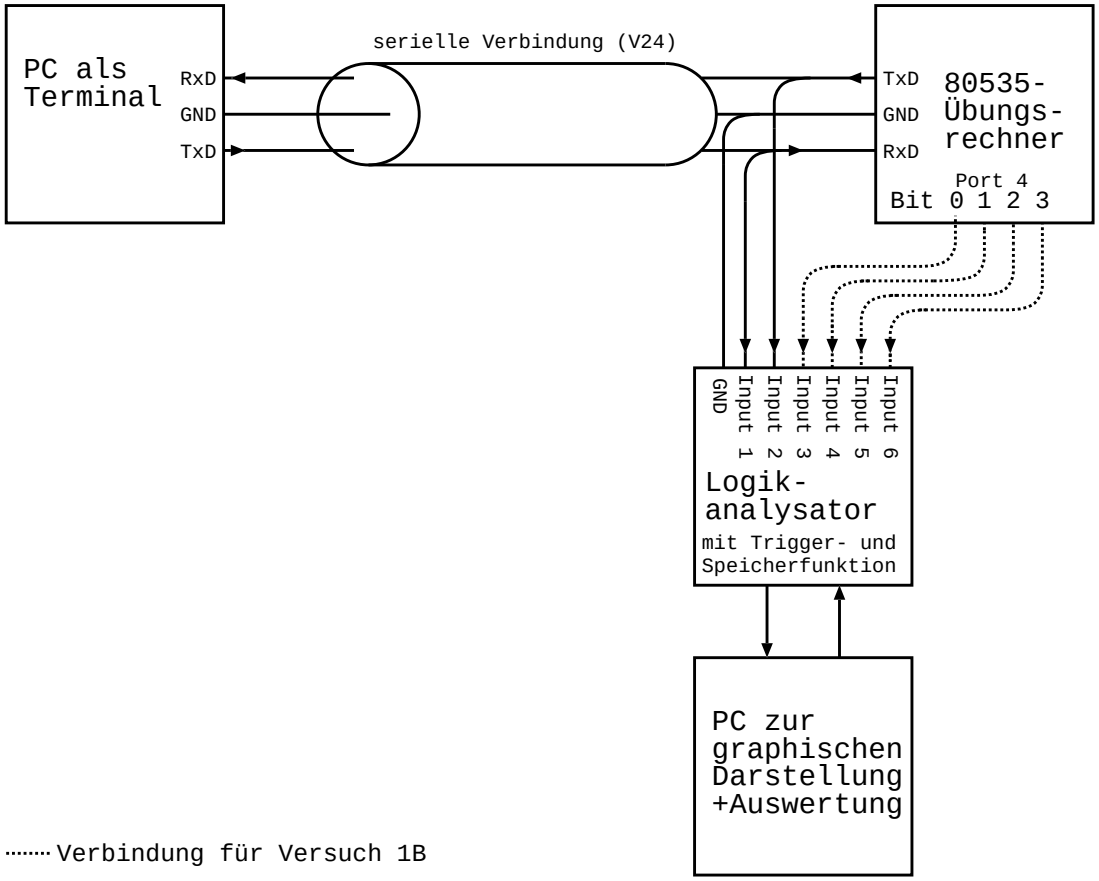

Abbildung 1: Versuchsaufbau

Ein Mikrocontroller des Typs 80C535 (Intel 80C51-kompatibel) ist über eine serielle Verbindung (V24) mit einem PC verbunden. Dieser PC dient als Terminal

(für die serielle Verbindung) und als Programmiereinheit für den Mikrocontroller.

Die serielle Verbindung wird mit Hilfe eines Logikanalysators abgegriffen. Der Logikanalysator überträgt die abgegriffenen Daten zu einem weiteren PC, welcher diese visualisiert. Eine weitere Aufgabe dieses zweiten PCs ist die Konfiguration des Logikanalysators.

3 Vorbetrachtungen

3.1 Die serielle Schnittstelle

Die serielle Schnittstelle, auch bekannt als V24 oder RS232[1], wird im PC-Bereich mehr und mehr durch den Universal Serial Bus (USB) verdrängt. Früher wurde die serielle Schnittstelle hauptsächlich zum Anschluss von Modems und Mäusen an den PC verwendet. Zu erkennen ist die serielle Schnittstelle an dem 25 oder 9-poligen Sub-D Stecker (male am PC). Wobei die 25-polige Variante fast vollständig verschwunden ist und die 9-polige seltener wird.

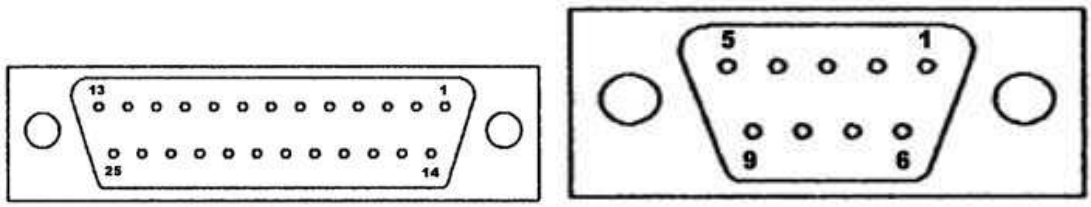

Abbildung 2: links: Sub-D 25 ; rechts: Sub-D 9 (Übernommen von Sephi (1999))

[1]Neue bzw. aktuelle Bezeichnung EIA-232 (de.wikipedia.org (2006a)).

3.1.1 Signale

Signalname		Pin	Rich.
PG Protection Ground	Schutzerde	Schirm	-
SG Signal (Common) Ground	Bezugssignal	5	-
TxD Transmitted Data	Ausgehende Daten	3	OUT
RxD Received Data	Eingehende Daten	2	IN
DTR Data Terminal Ready	Endgerät betriebsbereit	4	OUT
DSR Data Set Ready	Betriebsbereitschaft	6	IN
RTS Request To Send	Sendeteil EIN	7	OUT
CTS Clear To Send	Sendebereitschaft	8	IN
DCD Carrier Detect	Empfangssignalpegel vorhanden	1	IN
RI Ring Indicator	Ankommender Ruf	9	IN

(en.wikipedia.org (2006), Sephi (1999) und Messmer und Dembowski (2003))

Wobei die Spalten "Pin" und "Rich." die Pinnummer und die Richtung des Signals bezogen auf den PC angeben. Die Pinnummer bezieht sich in diesem Fall auf den 9-poligen Sub-D Anschluss.

Die drei wichtigsten Signale sind SG, TxD und RxD, wobei SG ein gemeinsames Bezugspotential definiert, TxD Daten sendet und RxD empfängt. Dies bezieht sich wieder auf die PC-Seite. In einem Modem sind die beiden Anschlüsse TxD und RxD intern vertauscht, so dass vom Modem aus gesehen RxD Daten sendet und TxD Daten empfängt.

PG ist heutzutage meist mit SG verbunden.

3.1.2 Handshaking

Der normale Ablauf einer Datenübertragung verläuft folgendermaßen:

- Der Computer aktiviert die Leitung DTR (Data terminal ready) in Richtung des Modems und wartet auf dessen Rückmeldung durch DSR (Data set ready). Damit besteht lokale Empfangsbereitschaft ohne Aktivierung des Senders.

- Wenn der Computer senden möchte, setzt er den Pin RTS (Request to send) und wartet auf die Sendebereitschaft CTS (Clear to send) des Modems.

Adaptiert von de.wikipedia.org (2006b).

Dieser Vorgang wird allgemeinhin als Hardware-Handshaking bezeichnet. Er dient nach Aufbau der Verbindung auch dazu, den Datentransfer zu unterbrechen, wenn ein Endgerät keine Daten mehr aufnehmen kann. (Wenn zum Beispiel der Puffer voll ist.) Da heutige Geräte wesentlich schneller Daten verarbeiten können, als diese von der seriellen Schnittstelle übertragen werden, sind diese Signale in den meisten Fällen überflüssig.

Um Signale – und somit Adern in der Leitung – zu sparen wurde das Software-Handshaking entwickelt. Bei diesem werden die Steuersignale über die gleichen Leitungen übertragen, wie die Daten: über RxD und TxD. Kann eine Seite keine Daten mehr aufnehmen, signalisiert sie das mit der Übertragung von dem Byte 13_{16} (^S – Control-S). Die Empfangsbereitschaft wird mit der Übertragung eines Bytes 11_{16} (^Q – Control-Q) signalisiert. Die meisten Systeme, die diese Handshake-Verfahren einsetzten, senden das 11_{16} in gewissen Zeitintervallen immer wieder. Dies hat den Sinn, einem neu angeschlossenen Gerät mitzuteilen, dass Empfangsbereitschaft besteht. Diese Verhalten wurde bei dem Laborversuch beobachtet. Das Software-Handshaking hat den Nachteil, dass nicht mehr alle Zeichen für die Datenübertragung genutzt werden können. Die Zeichen 11_{16} und 13_{16} dürfen nämlich nicht im Datenstrom vorkommen, da sie als Handshaking und nicht als Daten interpretiert werden würden.

Jede Standard-UNIX/Linux-Konsole versteht Software-Handshaking. Wird Control-S gedrückt, hält die Konsole die Datenausgabe so lange an, bis Control-Q gedrückt wurde.

Mit Hilfe des Signals DCD teilt das Modem dem Computer mit, dass ein Signalpegel (Eingehende Daten von der Gegenseite, verbundenem Modem) vorhanden ist. Das Signal RI wird genutzt um dem Computer mitzuteilen, dass das lokale Modem angerufen wird.

3.1.3 Signalpegel der RS-232-C-Schnittstelle

Die folgenden Graphiken geben einen Überblick über die Signalpegel und deren logischen Zuordnung der RS-232-C-Schnittstelle.

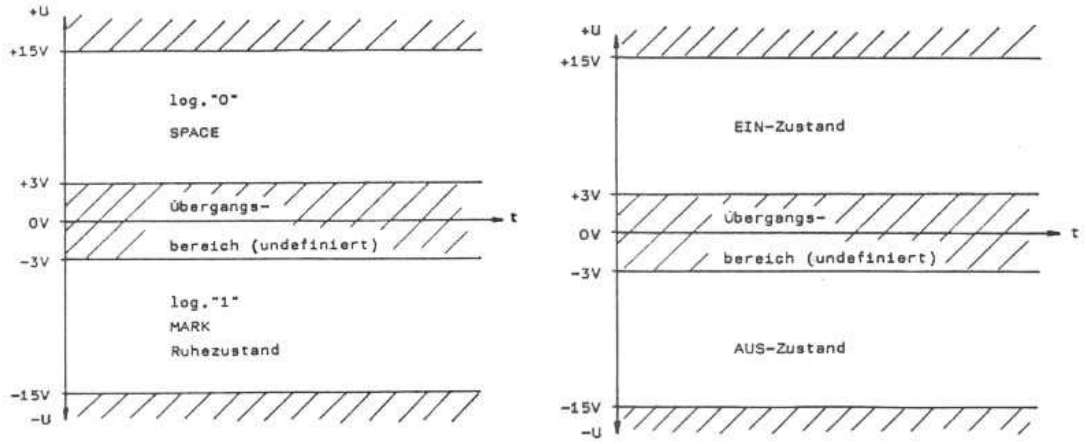

Abbildung 3: Signalpegel der RS-232-C (Übernommen von Sephi (1999))

Die linke Grafik zeigt die logische Zuordnung von Signalpegeln und Datenleitungen (RxD und TxD), die rechte von Signalpegeln und Steuerleitungen (CTS, RTS, DTR, DSR, DCD und RI).

Erlaubte Bereiche sind: von -3V bis -15V (low) und +3 bis +15V (high). Der Bereich von -3V bis +3V ist laut Standard verboten und führt zu nicht definierten Ergebnissen. Wobei zu erwähnen ist, dass moderne serielle Schnittstellen 0V häufig als low (-3V bis -15V) erkennen.

Bei den Datenleitungen gilt, dass der Low-Bereich einer logischen “1” zugeordnet ist und der High-Bereich einer logischen “0”. Der Low-Bereich entspricht dem Ruhezustand.

Ist nicht bekannt, welcher von zwei Drähten der TxD und welcher der RxD ist, kann dies durch einfaches Messen gegenüber SG festgestellt werden. Handelt es sich um TxD würde eine Spannung in einem Bereich von -3V bis -15V gemessen. Bei RxD würde eine Spannung von ca. 0V gemessen. Diese Messung bezieht sich auf Computer. Bei Modems sind RxD und TxD vertauscht.

3.1.4 Verbindung zweier Computer

Sollen zwei Computer über die serielle Schnittstelle verbunden werden, ist darauf zu achten, dass ein "gedrehtes" Kabel verwendet wird. Die folgende Abbildung zeigt links eine volle Kopplung, bei der alle Steuerleitungen verbunden wurden. Da Computer im allgemeinen Daten schneller verarbeiten können, als die serielle Schnittstelle diese überträgt, sind die Steuerleitungen überflüssig. Die Drei-Draht-Kopplung, links abgebildet, kann verwendet werden. Wendet ein Programm auf die Drei-Draht-Kopplung Hardware-Handshaking an, funktioniert dies nicht. Um diesem Problem vorzubeugen, sind die gestrichelt eingezeichneten Verbindungen herzustellen. Dies führt dazu, dass das lokale Hardware-Handshaking ausgetrickst wird. (Das Weglassen der gestrichelten Verbindungen ist eine der häufigsten Ursachen, wieso eine serielle Datenübertragung mit einer Drei-Draht-Verbindung nicht funktioniert.)

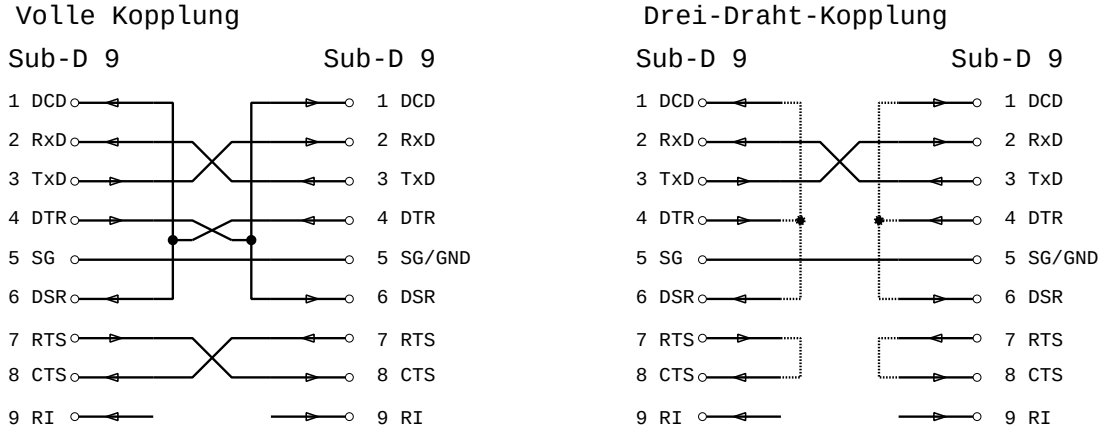

Abbildung 4: Kopplung zweier Computer

3.1.5 Beispiel einer Übertragung

Die folgende Graphik gibt ein Beispiel der Übertragung eines Bytes. Das Zeichen "G" (47_{16}) wurde mit ungerader Parität und einem Stoppbit übertagegen. Das Startbit ist immer "0_2" um sich vom Ruhezustand (= "1_2") abzuheben. Danach folgen die Datenbits, wobei das LSB (least significant bit) zuerst übertragen wird. Dannach wird die Anzahl der Einsen durch das Paritätsbit ungerade gemacht (odd parity). Das heisst, das Paritätsbit wird auf 1_2 gesetzt, wenn die Anzahl der Einsen gerade war. Zum Schluss folgt ein Stoppbit, welches "1_2" ist, um den Ruhezustand einzuleiten oder sich vom nächsten Startbit abzuheben. Der Übergang vom Ruhezustand (oder Stoppbit) zum Startbit wird zu Syncronisationszwecken verwendet.

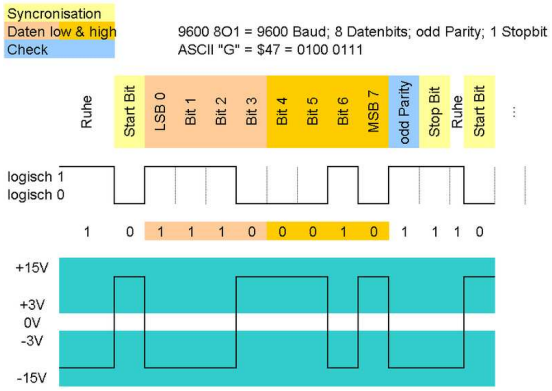

Abbildung 5: Beispiel einer Übertragung (Übernommen von de.wikipedia.org (2006a))

3.1.6 Übertragungsgeschwindigkeit Baud vs. bps

Die Übertragungsgeschwindigkeit wird normalerweise in bps (bit per second) gemessen. Bei der Datenübertragung durch ein Modem wird üblicherweise von Baud gesprochen. Baud ist definiert als Signaländerungen pro Sekunde. Wird pro Signaländerung ein Bit übertragen, so sind 9600baud gleich 9600bps. Dies trifft auf die serielle Schnittstelle zu. Modems sind allerdings in der Lage mehr als ein Bit pro Signalwechsel zu übertragen. In diesem Fall ist die Baudrate mit der Anzahl der pro Signalwechsel übertragenen Bits zu multiplizieren, um auf bps zu kommen.

3.2 Abtastung

3.2.1 Abtastrate

Um eine sichere Abtastung zu gewährleisten, ist mindestens mit vierfacher Geschwindigkeit – bezogen auf die maximale Änderungsgeschwindigkeit des abzutastenden Signals – abzutasten.

Geschwindigkeit: $9600 baud = 9600 bps = 9600 \frac{bit}{s} \Rightarrow \frac{1}{9600 \frac{bit}{s}} \approx 104 * 10^{-6} \frac{s}{bit}$

Um die vierfache Abtastrate zu erreichen, muss die Zeit zwischen den Abtastungen auf $\frac{1}{4}$ zu reduzieren.

Daraus folgt: Abtastgeschwindigkeit: $104 * 10^{-6} \frac{s}{bit} * \frac{1}{4} \approx 25 * 10^{-6} s$

3.2.2 Triggerpegel

Der Triggerpegel am Logikanalysator ist auf 3V einzustellen, da alles über 3V eine "0" darstellt. Alles andere wird als "1" interpretiert. Dabei ist zu beachten, dass das High-Signal eine logische "0" darstellt. Dies führt dazu, dass der Logikanalysator das Signal invertiert darstellt.

4 Versuchsdurchführung

4.1 Versuch 1A

Das Programm `ser_test` wurde übersetzt, auf den Mikrocontroller übertragen und gestartet. Dannach wurden mittels eines Terminalprogramms Zeichen zu dem Mikrocontroller gesendet. Es war zu beobachten, dass die Zeichen um eins erhört an das Terminalprogramm zurückgeschickt wurden. So wurde aus der eingegebenen "7" eine "8". Diese Übermittlung wurde abgefangen und analysiert. Dazu wurde die Triggerung so eingestellt, dass sie durch eine positive Flanke an RxD ausgelöst wurde. Der Pretrigger- und Posttriggerbereich wurden gleich gewählt.

Aufgefangene Sequenz:

1 00010011 0

Diese Sequenz wurde invertiert

0 11101100 1

Die Start und Stopp-Bits wurden entfernt und die Reihenfolge umgekehrt

$00110111_2 = 37_{16}$ = "7"

Die Antwort vom Mikrocontroller wurde etwas versetzt empfangen und lautete:

1 11100011 0

0 00011100 1

$00111000_2 = 38_{16} =$ "8"

Im Timing Diagramm wurde die Zeit zwischen dem Anfang des Startbits und der Anfangszeit des Stopbits abgelesen. Sie Betrug $936.005 * 10^{-6}s$. Dieser Wert wurde folgenderweise überprüft:

$\frac{1}{9600\frac{bit}{s}} * 9bit = 937.5 * 10^{-6}s$

Es ist zu erkennen, dass der abgelesene mit dem errechneten Wert hinreichend übereinstimmt.

Die Aufzeichnungen wurde in Form eines Timing Diagramms ausgedruckt. Siehe Anhang A.

4.2 Versuch 1B

Das Programm für den zweiten Versuch wurde übersetzt, auf den Mikrocontroller übertragen und gestartet.

Der Mikrocontroller sendet permanent "Messwerte" (8-Bit-Werte) über die serielle Schnittstelle. Die Bedeutung des gerade gesendeten Messwertes wird über die unteren 4 Bit von Port 4 signalisiert. An diesem Port treten alle Zahlen zwischen 0 und 15 auf. Mit dem Beginn der Übertragung eines neuen Messwertes steht auch die 4-Bit-Information an P4, und bleibt während der gesamten Sendezeit unverändert.

Es war festzustellen, welcher Messwert seriell übertragen wird, wenn an Port 4 die Gruppennummer – $1 = 0001_2$ – als Dualzahl ansteht.

Es wurden 4 Messleitungen des Logikanalysators an Port 4 angeschlossen. Der Logikanalysators wurde so eingestellt, dass die Triggerung einsetzte, wenn

1. eine positive Flanke an TxD auftrat, und
2. Bit 3 des Ports 4 low (0) ist, und
3. Bit 2 des Ports 4 low (0) ist, und
4. Bit 1 des Ports 4 low (0) ist, und
5. Bit 0 des Ports 4 high (1) ist.

Auch in diesem Fall wurden Pretrigger- und Posttriggerbereich gleich gewählt.

Folgendes Ergebnis wurde ermittelt:

1 01011101 0

invertieren

0 10100010 1

umdrehen, Start- und Stoppbit entfernen

$01000101_2 = 45_{16}$ = "E"

Der Ausdruck des Timing Diagramms ist im Anhang B zu finden.

5 Bibliographie und Referenzen

de.wikipedia.org (2006a) EIA-232 [Online] Available at
`http://de.wikipedia.org/wiki/EIA-232`

de.wikipedia.org (2006b) Datenflusskontrolle [Online] Available at
`http://de.wikipedia.org/wiki/Handshake`

en.wikipedia.org (2006) RS-232 [Online] Available at
`http://en.wikipedia.org/wiki/RS-232`

Messmer, H.-P.; Dembowski, K (2003) PC-Hardwarebuch 7^{th} edition. Munich: Addison-Wesley. ISBN 3-827-32014-3.

Sephi (1999) Die serielle Schnittstelle V.24 [Online] Available at
`http://www.inf.hs-zigr.de/ boehm/rt98/ferstl/v24vortrag.htm`

Anhang A: Timing Diagramm aus Versuch 1A

Anhang B: Timing Diagramm aus Versuch 1B