**ingeniTRON**
GmbH & Co. KG
Mühlstr. 1a
D-64625 Bensheim

Tel.: (06251) 5 82 62 - 40
Fax: (06251) 5 82 62 - 49
eMail: office@ingenitron.de
http://www.ingenitron.de

# RS232-I²C-Umsetzer
# **Typ: RIC-II**

Eigenschaften:

- Als I²C-Bus-Master zur Ankopplung an einem bestehenden I²C-Bus oder als passiver I²C-Busmonitor.
- Zur universellen Konfiguration und Parametrierung von Bausteinen mit I²C-Businterface.
- Monitor-Modus: Überwachung und Analyse der I²C-Kommunikation via PC.
- Terminal-Modus: Einfachste Bedienung über ein Terminalprogramm, mit Editierfunktion.
- ASCII-Modus zur Übertragung von Textdateien direkt aus dem Terminalprogramm heraus.
- Binärmodus zur schnellen Übertragung großer Datenmengen, z.B. zum Beschreiben großer EEPROMS (nicht für Terminalbetrieb geeignet).
- Betriebssystem-unabhängige Anschaltung an jeden PC mit RS232-Schnittstelle.
- Optional zuschaltbare Bus-Widerstände (2kΩ Pull-Up).
- Für beide Standard-Übertragungsgeschwindigkeiten 100kbit/s und 400kbit/s geeignet.
- Durch die I²C-Topologie unproblematische Zusammenarbeit mit dem in der Schaltung vorhandenen Busmaster.
- Zwei LEDs zur Anzeige des Datenverkehrs getrennt für den RS232- und den I²C-Bus.
- Für verschiedene RS232-Baudraten: 4800 / 9600 / 57600 / 115200 Baud.
- Einfache Kontaktierung des I²C-Busses über drei Schraubklemmen (SDA/SCL/GND).
- Großer, verpoltoleranter Versorgungsspannungsbereich: 9V ... 15V ac/dc.
- Kompakter Aufbau.

**Wichtig! Bitte lesen Sie vor Inbetriebnahme dieses Produkts die Bedienungsanleitung!**

Änderungsindex:

| Index | Änderung | Datum |
|---|---|---|
| 0 | neu erstellt | 26.04.2004 |
| | | |
| | | |

**Inhaltsverzeichnis:**



**Zahlenformate:**

Hexadezimale Zahlen werden mit einem führenden „0x...“ gekennzeichnet, z.B. 0x01, 0x02, 0xFF.

Binärwerte werden mit einem führenden „0b...“ gekennzeichnet, z.B.0b11010, 0b01100010.

Zahlen ohne Kennzeichnung sind im Dezimalformat.

**Zeichenerklärung:**
Einzelne Tastatureingaben bzw. Zeichencodes werden mit spitzen Klammern gekennzeichnet, z.B. <Strg+H>, <Enter> (Entertaste, als Zeichencode in diesem Fall nur das CarriageReturn 0x0D, das Linefeed 0x0A wird verworfen), <+>, <#>.

**Abkürzungen, Sonderzeichen**:

| | | |
|---|---|---|
| + (Plus) | ACK | Acknowledge Bit |
| - (Minus) | NAK | Nicht-Acknowledge Bit |
| * (Stern) | | Startsequenz am $I^2C$-Bus |
| # (Raute) | | Stopsequenz am $I^2C$-Bus |

# 1 Technische Daten

| | |
|---|---|
| **Spannungsversorgung** [1] | Über Steckernetzteil mit Standard-Kleinspannungsstecker 2,1/5,5mm (Innen-/ Außendurchmesser) |
| **Betriebsspannung $U_B$** | 9V ... 15Vac, 9V ... 18Vdc |
| Verpoltolerant | ja |
| **Leerlaufstromaufnahme $I_L$** | ≤ 25mA |
| **Betriebsmodi** [2] | Terminal-Modus<br>ASCII-Modus<br>Binär-Modus<br>Monitor-Modus |
| **I²C-Busanschaltung** | Über drei Schraubklemmen SDA/SCL/GND (Leitungsquerschnitt 0...1,5qmm) |
| Sendedaten Baudrate | 100kbit/s / 400kbit/s (peak [3] ) umschaltbar |
| Ausgangsendstufen | NPN Open-Collector |
| I²C-Buswiderstände | 2kΩ Pull-Up, getrennt zuschaltbar für SDA und SCL |
| Kapazitive Last | < 50pF |
| Ausgangs-Impedanz | > 100kΩ (high, logisch 1)<br>< 120Ω (low, logisch 0) |
| **RS232-Busanschaltung** | 9-pol Sub-D (geräteseitig: Buchse/female) |
| Baudrate | 4800/9600/57600/115200 Baud umschaltbar |
| **Einstellungen über DIL-Schalter** | Wahl des Betriebsmodus,<br>I²C-Übertragungsgeschwindigkeit<br>Wahl der RS232-Baudrate<br>Zuschaltbarer Buswiderstände (Pull-Ups) |
| **Anzeigen** | |
| Bereitschaft | LED grün |
| Datenverkehr I²C | LED rot |
| Datenverkehr RS232 | LED rot |

[1] Steckernetzteil nicht im Lieferumfang
[2] Ausführliche Beschreibung siehe folgende Kapitel
[3] Nur innerhalb eines Datenbytes und für Start- und Stopbedingung.
Während und nach einem Acknowledge-Bit sind Pausen > 10µs möglich .

# 2 Einstellmöglichkeiten, Auslieferungszustand

## 2.1 Einstellmöglichkeiten

Das RIC besitzt an der Vorderseite 8 DIL-Schalter, welche die folgende Bedeutung haben:

| DIL-Schalter | Bedeutung |
|---|---|
| 1 | Pull-Up Widerstand an SCL |
| 2 | Pull-Up Widerstand an SDA |
| 3 | --- (nicht belegt) |
| 4<br>5 | RS232-Baudrate: 4800 / 9600 / 57600 / 115200 Baud |
| 6 | $I^2C$-Baudrate: 100kbit/s / 400 kbit/s |
| 7<br>8 | Betriebsart: Terminal-Modus / ASCII-Modus / Binär-Modus / Monitor-Modus |

Mit den DIL-Schaltern #1 und #2 können für SCL und SDA getrennt die Pull-Up-Widerstände zugeschaltet werden. Damit die Buskommunikation funktioniert, müssen diese Widerstände jeweils genau einmal im System vorhanden sein. In der Regel sind sie bereits im Bussystem der Schaltung, mit dem das RIC arbeiten soll, integriert und können abgeschaltet werden.

| Schalter #1 | Schalter #2 | Baudrate |
|---|---|---|
| OFF | x | Pull-Up an SCL abgeschaltet |
| ON | x | Pull-Up an SCL zugeschaltet |
| x | OFF | Pull-Up an SDA abgeschaltet |
| x | ON | Pull-Up an SDA zugeschaltet |

Schalter #3 ist nicht belegt.

Schalter #4 und #5 stellen die RS232-Baudrate des RIC ein:

| Schalter #4 | Schalter #5 | Baudrate |
|---|---|---|
| OFF | OFF | 4800 Baud |
| OFF | ON | 9600 Baud |
| ON | OFF | 57600 Baud |
| ON | ON | 115200 Baud |

Schalter #6 stellt die $I^2C$-Busgeschwindigkeit ein:

| Schalter #6 | Baudrate |
|---|---|
| OFF | 400 kbit/s |
| ON | 100 kbit/s |

Schalter #7 und #8 bestimmen den Betriebsmode des RIC. Siehe hierzu die ausführliche Beschreibung im Kapitel Betriebsarten:

| Schalter #7 | Schalter #8 | Baudrate |
|---|---|---|
| OFF | OFF | nicht definiert |
| OFF | ON | Binär-Modus |
| ON | OFF | ASCII-Modus |
| ON | ON | Terminal-Modus |

## 2.2 Auslieferungszustand

Das Produkt befindet sich im folgenden Auslieferungszustand:

| DIL-Schalter | Stellung | Bedeutung |
|---|---|---|
| 1 | OFF | Pull-Up Widerstand an SCL abgeschaltet |
| 2 | OFF | Pull-Up Widerstand an SDA abgeschaltet |
| 3 | d.c. | --- |
| 4<br>5 | OFF<br>ON | RS232-Baudrate: 9600 Baud |
| 6 | ON | $I^2C$-Baudrate: 100kbit/s |
| 7<br>8 | ON<br>ON | Betriebsart: Terminal-Modus |

# 3 Betriebsarten

## 3.1 Grundsätzliches

Damit das RIC mit dem PC über die RS232-Schnittstelle kommunizieren kann, müssen die Kommunikationsparameter der PC-Schnittstelle wie folgt eingestellt werden:

Baudrate: 4800 Baud, 9600 Baud, 57600 Baud oder 115200 Baud, je nach Einstellung des RIC.
Datenbits: 8.
Parity: kein Parity.
Stopbits: 1.

Damit die $I^2C$-Kommunikation funktioniert, müssen Pull-Up-Widerstände an der Daten-(SDA-) Leitung und der Takt-(SCL-)Leitung vorhanden sein. Sie können dies prüfen, indem Sie im stromlosen Zustand den Widerstand der Leitungen SDA und SCL gegen die 5V-Versorgung messen. Wenn der Widerstand größer als 3kΩ ist, müssen die Pull-Up Widerstände des RIC zugeschaltet werden.

Stellen Sie weiterhin die $I^2C$-Baudrate des RIC gemäß Ihren Schaltungsparametern ein. Wenn Sie sich nicht sicher sind, stellen Sie hier 100kbit/s ein, das funktioniert bei allen $I^2C$-kompatiblen ICs.

## 3.2 Terminal-Modus

Im Terminalmodus kann das RIC über ein einfaches Terminalprogramm bedient werden. Die folgenden Beispiele und Screenshots beziehen sich auf das in jeder Standard-Installation von MS-Windows® enthaltene Terminalprogramm HyperTerm.

Im folgenden wird weiterhin davon ausgegangen, dass sich das RIC im Auslieferungszustand befindet. Siehe hierzu das Kapitel Auslieferungszustand.

### *3.2.1 Kommunikationsbeschreibung*

Im Terminalbetrieb werden alle Befehle und Daten als ASCII-Zeichen gesendet und empfangen.

**Befehls- und Datenbytes** werden immer als 2 Zeichen hexadezimal kodiert, z.B. bedeuten die Zeichenkette „1F“ oder „1f“ die Zahl 0x1F (hexadezimal) bzw. 31 dezimal. Groß- und Kleinschreibung der Hex-Ziffern werden nicht unterschieden. Zahlen kleiner 0x10 werden mit führender 0 geschrieben, z.B. „02“, nicht nur „2“!

**Antwortbytes**, die Sie vom Slave zurückerwarten, müssen Sie mit den Platzhalter-Werten „FF“ oder „ff“ (dezimal 256) kodieren. Diese Bytes werden in der Antwort dann mit dem Wert der Slave-Antwort überschrieben.

**Start- und Stopsequenzen** des I²C-Busses werden mit den Zeichen „*“ (Stern als Startzeichen) und „#“ (Raute als Stopzeichen) kodiert.

Das **Acknowledge** kann vom RIC entweder aktiv gesendet werden (ACK, kodiert durch das Plus-Zeichen „+“) oder bewußt als Nicht-Acknowledge (NAK, kodiert durch ein Minus-Zeichen „-“).

Erwarten sie ein **Acknowledge vom Slave**, dann muss an dieser Stelle ein „-“ (Minus) als Platzhalter für ein vom Slave erwartetes Acknowledge gesendet werden. In der Antwort wird dieser Platzhalter dann durch das ACK oder NAK vom Slave überschrieben.

Nachfolgend eine Übersicht der erlaubten ASCII-Zeichen:

| **Zeichen** | **Bedeutung** |
|---|---|
| * (Stern) | Startsequenz |
| 0...9, A...F oder a...f | Hex-Ziffer, je 2 Stück ergeben einen Bytewert |
| + (Plus) | aktives Acknowledge (ACK) des RIC |
| - (Minus) | Nicht-Acknowledge (NAK) oder Platzhalter für Slave-ACK |
| # (Raute) | Stopsequenz |

Empfängt das RIC andere als diese Zeichen, dann werden sie verworfen.

Ebenso werden Zeichen ignoriert, die an falscher Stelle kommen, z.B. eine zweite Startbedingung direkt hinter der ersten oder ein ACK, obwohl der Bytewert (2 Zeichen) noch nicht vollständig eingegeben wurde.

Ausnahmen sind die Tasten <◄— > (Rück- oder Löschtaste) und <Enter>. Siehe dazu das folgende Unterkapitel.

### *3.2.2 Zeichenpufferung, Editierfunktion*

Das RIC puffert bis zu 1023 eingegebene Zeichen, bevor sie auf dem I²C-Bus gesendet werden. Diese können, z.B. bei Falscheingabe, editiert werden, indem die Rücktaste <◄— > einmal oder mehrmals gedrückt wird. Das oder die letzten Zeichen werden dann aus dem Puffer gelöscht.

Erst nach Drücken der <Enter>-Taste wird der Befehlsstring ausgeführt und die Daten auf dem I²C-Bus gesendet und als Quittung von dort, ggf. mit der Slave-Antwort, zurückgesendet. Der Befehlsstring kann danach fortgesetzt werden, er muss also vorerst nicht mit <#> (Stopzeichen) beendet werden.

**Hinweis:** Die <Enter>-Taste wird nur akzeptiert nach einem ACK, einem NAK oder einem <#> (Stopzeichen). Nur dann können alle bis dahin eingegebenen Zeichen sinnvoll gesendet werden. Ein <Enter> nach einem „halben“ Bytewert macht keinen Sinn und wird ignoriert.

**Hinweis:** Um mit dieser Funktion komfortabel arbeiten zu können, sollte das Terminalprogramm richtig konfiguriert sein. Siehe hierzu das Kapitel Konfiguration des Terminalprogramms.

### *3.2.3 Beispiele*

Nachfolgend wird ein Beispiel gegeben zum Schreiben und Auslesen eines Wertes aus einem EEPROM (Z.B. 24C02, 256 Byte EEPROM).

Das EEPROM habe die Busadresse (7 Bit binär) 0b1010000 [4]. Mit dem Befehlsflag „Lesen“ (Eins-Bit) ergibt sich ein Bytewert 0b10100001 (binär) bzw. 0xA1 (hex). Mit dem Befehlsflag „Schreiben“ (Null-Bit) ergibt sich ein Bytewert 0b10100000 (binär) bzw. 0xA0 (hex).

Tip: Wenn Sie diese Beispiele nachvollziehen wollen, müssen Sie Ihr EEPROM ggf. zunächst mit Daten füllen. Ansonsten erhalten Sie immer 0xFF als Antwort eines unbeschriebenen EEPROMS.

**Hinweis:** Die folgenden Tastatureingaben können ohne Leerzeichen gemacht werden. Die Leerzeichen in diesen Beispielen dienen nur der besseren Lesbarkeit!

**Beispiel 1:** Gelesen werden soll die Speicherzelle 0x12 (dezimal 18). Sie habe den Inhalt 0x34.

Dazu muss zunächst ein Dummy-Write [5] an Adresse 0x12 durchgeführt werden, um den EEPROM-internen Adresszeiger korrekt zu setzen:

Tastatureingabe:
* A1 - 12 - #

Antwort:
* A1 + 12 + #

Jetzt kann gelesen werden:

Tastatureingabe:
* A1 - FF - #

Antwort:
* A1 + 34 - #

[4] Die Busadresse ergibt sich in diesem Falle aus dem Device Type Identifier (1010b) und der Beschaltung der drei Adressleitungen des EEPROMs mit GND (000b).
[5] Beim Lesen wird keine interne Registeradresse übergeben, stattdessen wird diese vorher durch ein Dummy-Write (abgebrochener Schreib-Befehl) eingestellt.

Man sieht hier, wie der Platzhalterwert (0xFF) vom Byteinhalt des EEPROMs überschrieben wird.

**Beispiel 2:** Gelesen werden sollen drei Speicherzellen hintereinander ab Byteadresse 0x12 (dezimal 18). Sie haben die Inhalte 0x34, 0x56 und 0x78.

Dazu muss zunächst wieder ein Dummy-Write an Adresse 0x12 durchgeführt werden, um den EEPROM-internen Adresszeiger korrekt zu setzen:

Tastatureingabe:
* A1 - 12 - #

Antwort:
* A1 + 12 + #

Jetzt kann gelesen werden:

Tastatureingabe:
* A1 - FF + FF + FF - #

Antwort:
* A1 + 34 + 56 + 78 - #

Man sieht hier das NAK am Ende des Befehls („-“) vor der Stopbedingung („#“), welches anzeigt, dass keine weiteren Daten mehr gelesen werden sollen.

**Beispiel 3:** Geschrieben werden sollen drei Speicherzellen hintereinander ab Byteadresse 0x12 (dezimal 18), und zwar mit den Inhalten 0x9A, 0xBC und 0xDE.

Jetzt kann geschrieben werden:

Tastatureingabe:
* A0 - 12 - 9A - BC - DE - #

Antwort:
* A0 + 12 + 9A + BC + DE + #

Man sieht hier, wie die Platzhalter für die ACK-Bits („-“ Zeichen) vom Slave mit ACK („+“ Zeichen) überschrieben werden.

Und jetzt noch eine Kontroll-Lesung, ob die Bytes richtig abgelegt worden sind:

Dazu muss zunächst wieder ein Dummy-Write an Adresse 0x12 durchgeführt werden, um den EEPROM-internen Adresszeiger korrekt zu setzen:

Tastatureingabe:
* A1 - 12 - #

Antwort:
* A1 + 12 + #

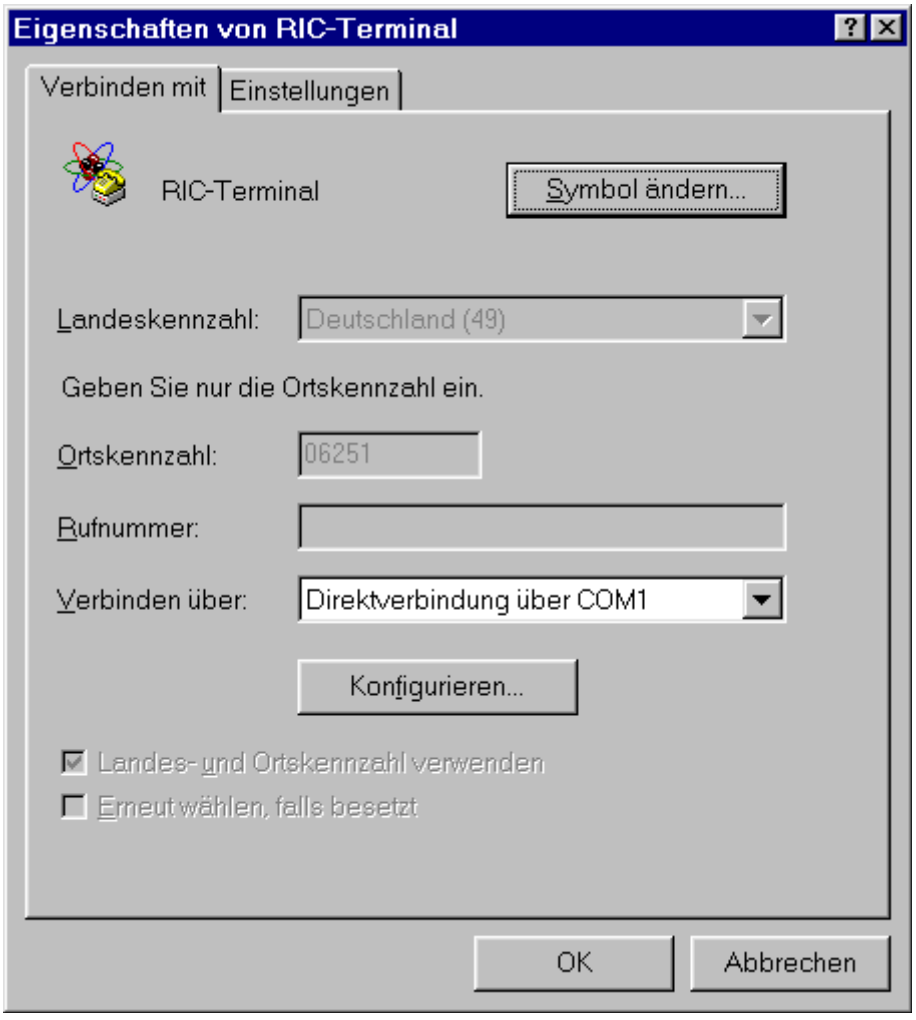
Tastatureingabe:
* A1 - FF + FF + FF - #

Antwort:
* A1 + 9A + BC + DE - #

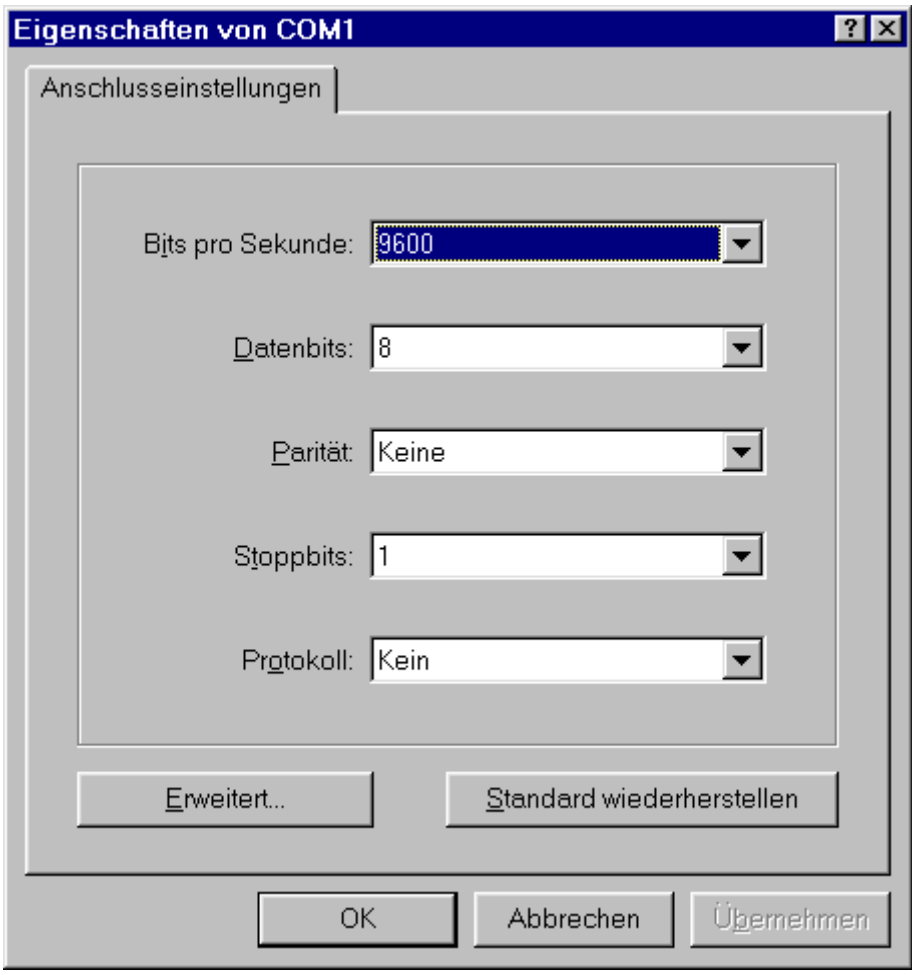
### *3.2.4 Konfiguration des Terminalprogramms*

Zur Konfiguration Ihres Terminalprogramms gehen Sie wie folgt vor:

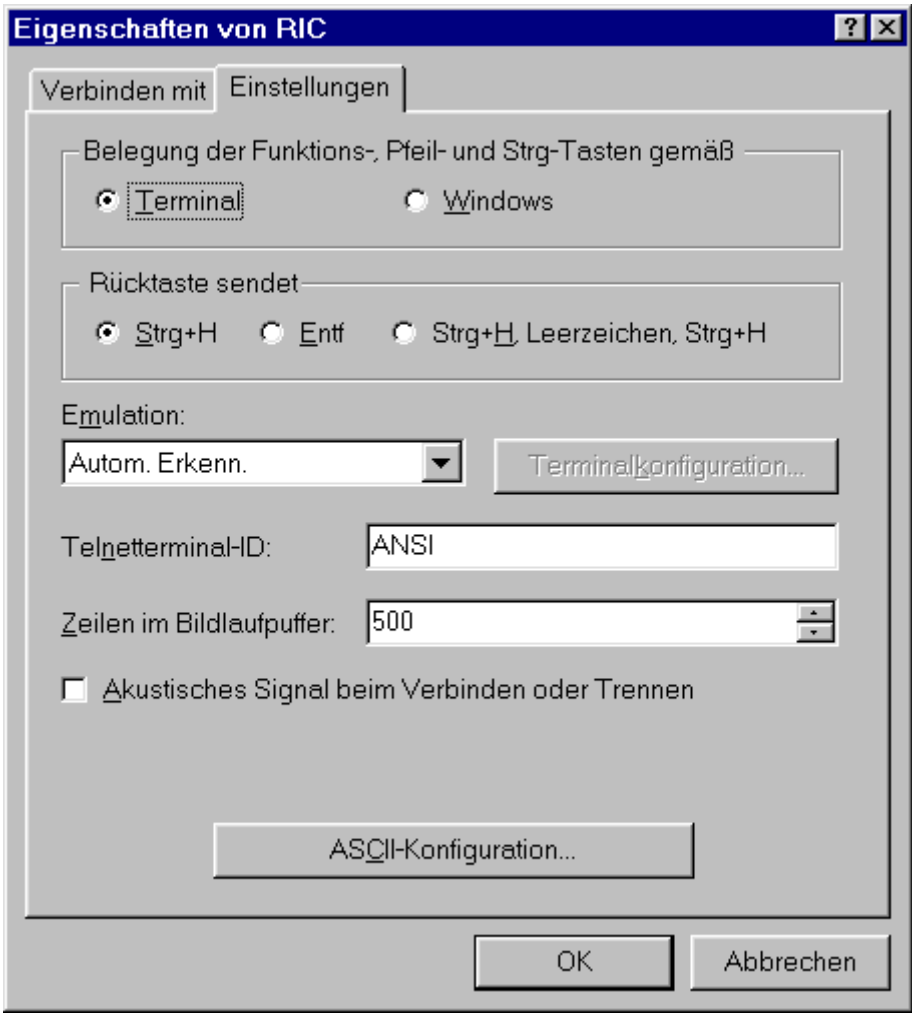
**Schritt 1:** Wählen Sie den Kommunikationsport Ihres PCs, an dem das RIC angeschlossen ist, üblicherweise COM1 oder COM2. In HyperTerm finden Sie diese Einstellung im Menü <Datei><Eigenschaften>:



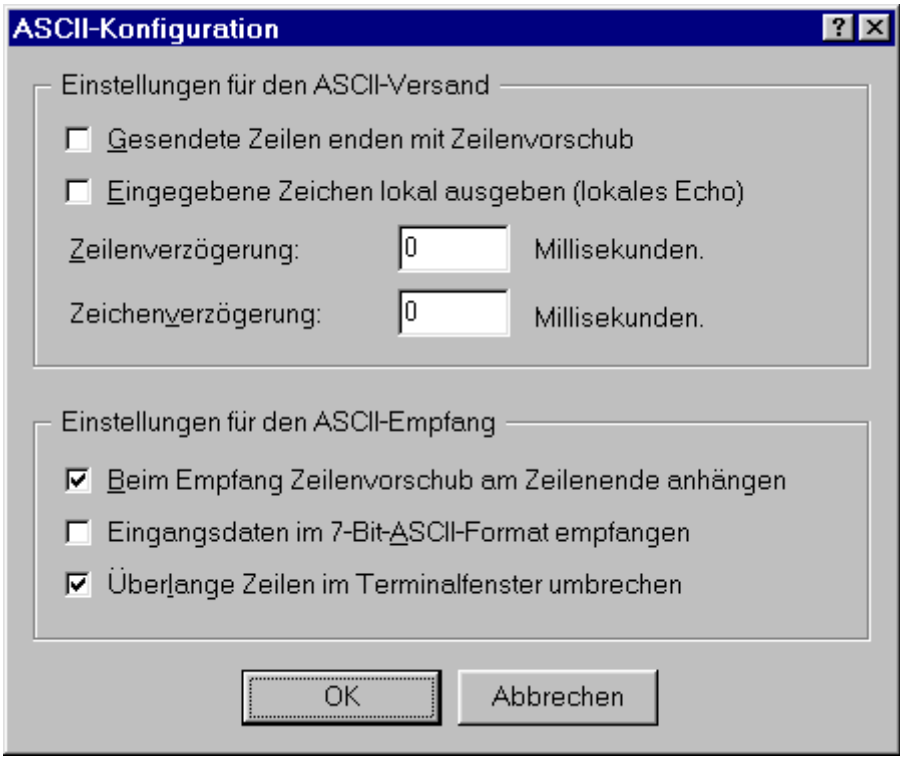
**Schritt 2:** Stellen Sie die Kommunikationsparameter ein. Das RIC kommuniziert mit 9600 Baud (Auslieferungszustand), 8 Datenbits, ohne Parity, einem Stopbit und ohne besonderem Handshake-Protokoll. . In HyperTerm finden Sie diese Einstellung im Menü <Datei><Eigenschaften> nach Drücken der Befehlstaste „Konfigurieren“:



**Schritt 3 :** Stellen Sie die Eigenschaften Ihres Terminals so ein, dass bei Betätigen der Rücktaste das Zeichen <Strg+H> (Backspace, Zeichencode 0x08) gesendet wird. Sie haben damit später die Möglichkeit, Tippfehler zu korrigieren. In HyperTerm finden Sie diese Einstellung im Menü <Datei><Eigenschaften> unter der Registerkarte „Einstellungen“:



**Schritt 4:** Beim Senden von Zeichen muss das lokale Echo ausgeschaltet sein. Die gesendeten Zeichen erscheinen sonst doppelt auf dem Bildschirm, da das RIC jedes empfangene und gültige Zeichen zur Quittierung zurücksendet.
Beim Empfang der Zeichen sollte am Zeilenende ein Zeilenvorschub angehängt werden, da ansonsten Ihre nächste Eingabe am Anfang der selben Zeile dargestellt wird. In HyperTerm finden Sie diese Einstellung im Menü <Datei><Eigenschaften> unter der Registerkarte „Einstellungen“ nach Drücken der Befehlstaste „ASCII-Konfiguration“:




Das Terminalprogramm ist jetzt korrekt konfiguriert.

## 3.3 ASCII-Modus

Der ASCII-Modus eignet sich besonders, um Textdateien mit vorgefertigten Befehlsstrings an den RIC zu senden. Die Zeichenkodierung und der Befehlsaufbau ist nahezu identisch zum Terminal-Modus. Für eine detaillierte Beschreibung sehen Sie bitte dort nach. Im folgenden werden nur die Abweichungen hiervon beschrieben.

### *3.3.1 Abweichungen zum Terminal-Modus*

Die eingegebenen Zeichen werden nicht gepuffert, sondern direkt vom RIC verarbeitet. Ein nachträgliches Editieren von Zeichen ist also nicht mehr möglich.

Dadurch ist es möglich, nahezu uneingeschränkt große Textdateien mit Befehlsstrings an das RIC zu senden.

Weiterhin werden die eingegebenen bzw. vom PC gesendeten Zeichen nicht als Echo zurückgesendet.

Antwortzeichen werden wie gehabt zurückgesendet, allerdings Voll-Duplex, also zeitnah direkt nach dem Empfang.

## 3.4 Binär-Modus

Die ASCII-Zeichenkodierung ist hinsichtlich einer geschwindigkeitsoptimierten Übertragung nicht optimal. Insbesondere bei großen Datenmengen, z.B. dem Beschreiben großer EEPROMS, macht sich das bemerkbar. Außerdem müssen dabei alle Bytewerte des EEPROMs erst ASCII-kodiert werden, was, wenn die Daten bereits als Binärwerte vorliegen, sehr mühsam ist.

Im Binärmodus werden deshalb alle Daten als echte Zahlenwerte übertragen.

Da man jetzt keine Zeichen zur Kodierung von Start- und Stopbedingung mehr zur Verfügung hat, geschieht dies über die RTS-Leitung der RS232-Schnittstelle. Ein Wechsel des Zustandes der RTS-Leitung von 0 nach 1 (aktivierung) gilt als Startbedingung, ein Wechsel von 1 nach 0 als Stopbedingung.

Das Acknowledge wird immer vom RIC aktiv auf dem $I^2C$-Bus gesendet, egal ob dies notwendig ist oder nicht oder ob auch der Slave ein ACK senden will. Durch die $I^2C$-Busphysik (Open-Collector) gibt es dabei zwar keine ernsthaften Datenkollisionen, es entspricht aber auch nicht dem Standard und vor allem ermöglicht es nicht, auf ein evtl. ausbleibendes ACK vom Slave zu reagieren.

**Hinweis:** Um speziell beim Einlesen von Daten vom Slave eine saubere Abbruchbedingung zu bekommen (Ende des Einlesens, normalerweise vom Master durch ein NAK einzuleiten), muss nach einem Einlesevorgang noch einmal eine Start- und eine Stop-Sequenz direkt hintereinander gesendet werden. Der Slave wird dadurch sicher zurückgesetzt.

**Beispiel 1:** Schreiben von 3 Datenbytes eines EEPROMs mit Busadresse 0b1010000 im Binärmodus (Wert 0xA0 incl. Schreib-Flag). Die RTS-Leitung sei inaktiv (0).

<RTS-Leitung aktivieren>
<Busadresse+Schreibflag (0xA0)><Bytewert 1><Bytewert 2><Bytewert 3>
<RTS-Leitung deaktivieren>

Die Antwort sieht folgendermaßen aus:
<Busadresse+Schreibflag (0xA0)><Bytewert 1><Bytewert 2><Bytewert 3>

**Beispiel 2:** Lesen von 3 Datenbytes eines EEPROMs mit Busadresse 0b1010000 im Binärmodus (Wert 0xA1 incl. Lese-Flag). Die RTS-Leitung sei inaktiv (0).

<RTS-Leitung aktivieren>
<Busadresse+Leseflag (0xA1)><0xFF><0xFF><0xFF>
<RTS-Leitung deaktivieren>
<RTS-Leitung aktivieren>
<RTS-Leitung deaktivieren>

Die Antwort sieht folgendermaßen aus:
<Busadresse+Leseflag (0xA1)><Bytewert 1><Bytewert 2><Bytewert 3>

## 3.5 Monitor-Modus

Diese Betriebsart ermöglicht die Analyse des Datenverkehrs auf einem bestehenden $I^2C$-Bus. Das RIC fungiert hierbei nicht, wie in den anderen Betriebsmodi, als Master, sondern nur als passiver Mithörer am Bus. Er greift nicht aktiv in die Kommunikation ein.

### *3.5.1 Kodierung der Kommunikationsdaten und -ereignisse*

Alles, was das RIC am $I^2C$-Bus hört, wird in ASCII-Zeichen übersetzt und an den PC gesendet. Dabei werden zur Kodierung die folgenden Zeichen verwendet:

| **Zeichen** | **Bedeutung** |
|---|---|
| * (Stern) | Startsequenz |
| 0...9, A...F oder a...f | Hex-Ziffer, je 2 Stück ergeben einen Bytewert |
| + (Plus) | aktives Acknowledge (ACK) des RIC |
| - (Minus) | Nicht-Acknowledge (NAK) oder Platzhalter für Slave-ACK |
| # (Raute) | Stopsequenz |
| X | Kennzeichnung eines internen Pufferüberlaufs |

Die Darstellungsweise der Start- und Stop-Sequenzen, der Datenbytes und der Acknowledge-Bits ist demnach identisch mit der Zeichenkodierung des

Terminalmodus. Für weitere Details zur Darstellung der Daten lesen Sie bitte das Kapitel Terminalmodus.

### *3.5.2 Anzeige eines internen Pufferüberlaufs*

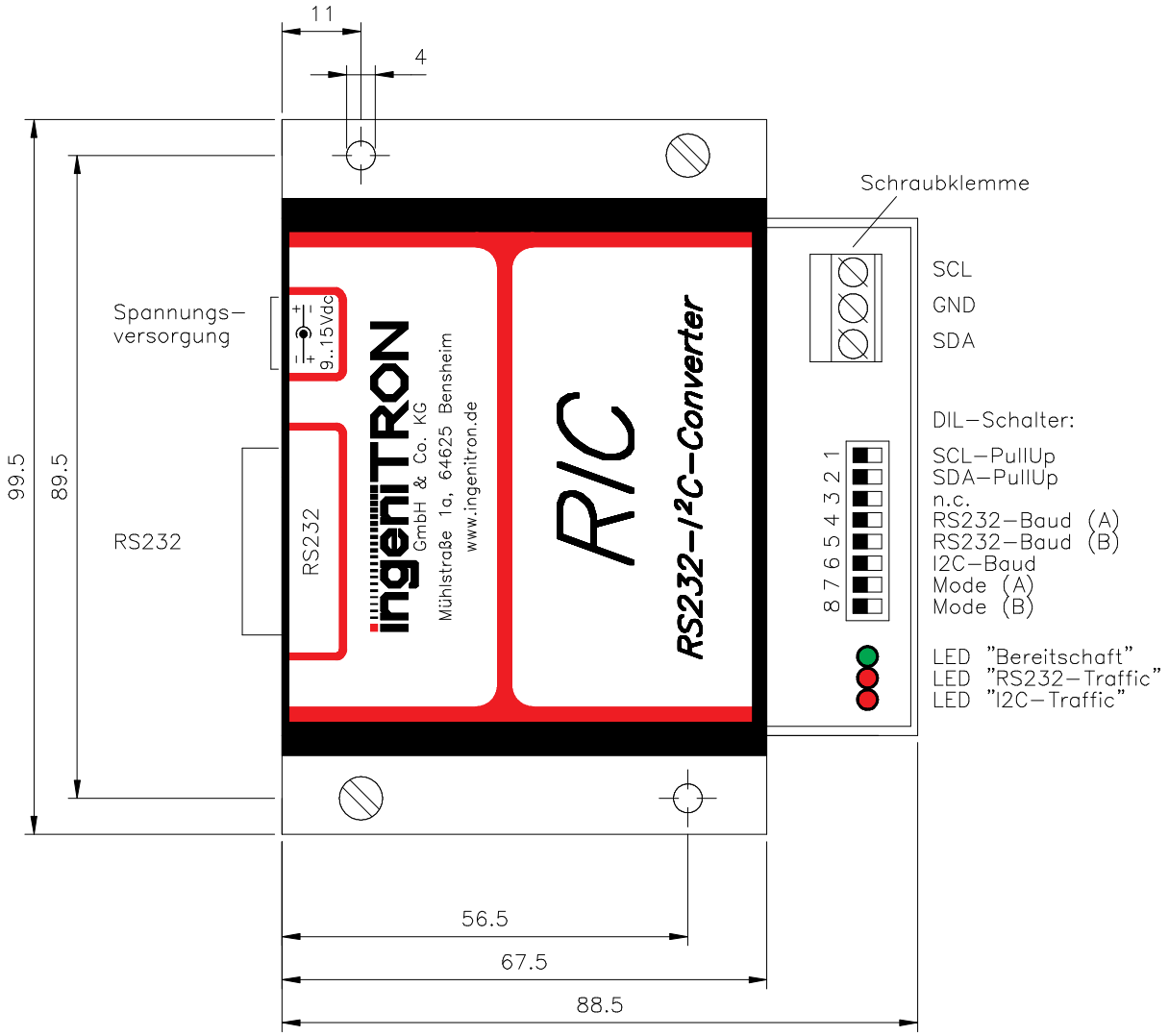
Das RIC puffert intern bis zu 1022 Zeichen an ASCII-Zeichen, die wegen der geringeren RS232-Bandbreite noch nicht zum PC gesendet werden konnten.
Das ist notwendig, da die $I^2C$-Buskommunikation mit einer höheren Geschwindigkeit als der maximalen RS232-Baudrate erfolgt und außerdem durch die ASCII-Kodierung RS232-seitig mehr Zeichen übertragen werden müssen, als auf der $I^2C$-Seite kommuniziert werden.

Der Puffer ist als FiFo-Stack aufgebaut. Er wird bei laufender $I^2C$-Buskommunikation mit den ASCII-kodierten Zeichen gefüllt und gleichzeitig durch die Sendung der Daten an den PC wieder geleert.

Ein Pufferüberlauf kann dadurch auftreten, dass über eine längere Zeit mehr $I^2C$-Daten empfangen und zur Sendung an den PC ASCII-kodiert werden, als zeitgleich via RS232-Bus an den PC gesendet werden können. Übersteigt die Anzahl der zu puffernden Zeichen die Puffergröße, dann wird ein Sonderzeichen („X“) als vorerst letztes Zeichen in den Puffer geschoben.

Die Aufzeichnung von weiteren $I^2C$-Busdaten wird vorerst gestoppt, bis der Puffer vollständig an den PC gesendet und damit geleert worden ist. Das letzte Zeichen des aktuellen Pufferinhalts, das der PC empfängt, ist das Pufferüberlauf-Zeichen „X“. Im Anschluss daran wird die Datenaufzeichnung, Pufferung und Sendung zum PC wie gehabt fortgesetzt.

# 4 Anschlussbild, Gehäuseabmessungen



# 5 Inbetriebnahme und Fehlersuche

Sollten Sie Probleme mit der Kommunikation haben, können Sie anhand folgender Schritte feststellen, auf welcher Seite (RS232 oder $I^2C$) das Problem liegt:

1. Das RIC bleibt vorerst stromlos (ohne Netzteil). Stellen Sie seine DIL-Schalter in den Auslieferungszustand (siehe Kapitel Auslieferungszustand).
2. Verbinden Sie das RIC mit dem PC und starten Sie ein Terminalprogramm. Stellen Sie dabei sicher, dass das Terminalprogramm korrekt eingestellt ist: Übertragungsparameter 9600,N,8,1.
3. Schalten Sie jetzt die Versorgungsspannung des RIC zu. Die grüne Bereitschafts-LED am RIC sollte leuchten.
   **Ist dies nicht der Fall**, funktioniert Ihre Spannungsversorgung nicht korrekt.
4. Das Terminalprogramm sollte nach der RIC-Aktivierung einmalig eine Zeichenkette mit seinem Namen und dem Betriebsmodus empfangen und anzeigen.
   **Ist dies nicht der Fall**, dann ist die Kommunikation zwischen PC und RIC gestört. Gehen Sie dann wie folgt vor:
   a. Senden Sie eine lange Folge von Start- und Stop-Zeichen im Wechsel („ * # * # * # * # ...“)
   b. Die LED zur Anzeige des Datenverkehrs auf der RS232-Seite sollte leuchten/flackern.
      **Ist dies nicht der Fall**, dann empfängt das RIC keine Daten. Überprüfen Sie dann nochmals die Kommunikationsparameter.
      **Ist dies der Fall** und Sie sehen kein Echo Ihrer gesendeten Daten im Terminalprogramm, dann ist die Einstellung Ihres Terminalprogramms fehlerhaft. Das RIC empfängt und sendet korrekt, die Antwort wird Ihnen aber nicht angezeigt.
      **Achtung:** Verwechseln Sie nicht das lokale Echo Ihrer Zeichen mit den zurückgesendeten Zeichen. Wenn Sie „lokales Echo“ in Ihrem Terminalprogramm aktiviert haben, werden Ihre eingegebenen Zeichen immer zuerst lokal ausgegeben. Die RIC-Antwort wird dann, sofern alles funktioniert, im Anschluss daran nochmals angezeigt!
5. Wenn bis zu diesem Schritt alles funktioniert hat, dann versuchen Sie jetzt, ein Byte eines Bausteins am $I^2C$-Bus zu lesen. Senden Sie dazu die Zeichenfolge ´´ <*> <Bausteinadr. + Leseflag (1)> <NAK [1)] > <FF> < NAK [1)] > <#> ´´
6. Sie sollten daraufhin die folgende Antwort erhalten:
   ´´ <*> <Bausteinadr. + Leseflag (1)> <ACK [1)] > <Wert des gelesenen Bytes> < NAK [1)] > <#> ´´
7. **Ist dies nicht der Fall**, dann funktioniert die $I^2C$-Buskommunikation nicht korrekt. Müssen Sie ggf. die Buswiderstände zuschalten oder sind sie vielleicht zugeschaltet und damit doppelt im System vorhanden? Haben Sie SCL und SDA vertauscht? Haben Sie GND mit Ihrer Schaltung verbunden?

[1)] ACK = Acknowledge-Bit (“+”)
NAK = Nicht-Acknowledge-Bit („-“)

# 6 Gewährleistung, Haftungsausschluss

Die Gewährleistung und Haftung für unser Produkt wird nur in solchen Fällen übernommen, in denen ausgeschlossen werden kann, dass der Gewährleistungs- oder Haftungsgrund auf Mängel in anderen, an unser Produkt angeschaltete oder unmittelbar oder mittelbar damit in Verbindung stehenden Produkten oder deren Dokumentationen zurückzuführen ist.

Es kann keine Gewährleistung mehr übernommen werden, falls andere, nicht zur Firma Ingenitron gehörende oder von ihr beauftragte Personen Veränderungen oder Reparaturen an diesem Produkt vornehmen.

Bei Sach- oder Personenschäden, die durch unsachgemäße Handhabung oder Nichtbeachtung dieser Bedienungsanleitung verursacht werden, übernehmen wir keine Haftung. In solchen Fällen erlischt des weiteren jeglicher Garantieanspruch.

Wir weisen weiterhin ausdrücklich darauf hin, dass dem Anwender die Verantwortung für Maßnahmen zur Abwehr von Personen-, Sach- und Vermögensschäden obliegt, welche aus dem Einsatz unseres Produkts resultieren.

Ein dauerhafter Einsatz als Busmaster in einer bestehenden elektronischen Schaltung ist zwar möglich, jedoch nicht vorgesehen. Übertragungsfehler bei der Protokollumsetzung können nicht ausgeschlossen werden.

Insbesondere für Bereiche oder Geräte, die eine Gefahr für Leben und Gesundheit darstellen können, sind vor der Inbetriebnahme unseres Produkts eigenständige, nicht durch unser Produkt zu beeinflussende Sicherungsmaßnahmen vorzusehen.

Die Firma Ingenitron haftet nicht für Schäden, auch nicht für Folgeschäden, die durch ihr Produkt entstehen, es sei denn, dass in Fällen des Vorsatzes oder der groben Fahrlässigkeit zwingend gehaftet wird.