Zur Startseite
Zur Artikelübersicht

# Der RS485 Bus

## Einleitung

Der RS485 Bus ist eine sehr interessante Schnittstelle. Dieser Artikel erklärt was der RS485 Bus eigentlich ist, wie er funktioniert und wo die Unterschiede und die Vor− und Nachteile gegenüber der RS232 Schnittstelle sind. Auch die Unterschiede zu CAN werden beschrieben. In diesem Artikel sollen weniger technische Details erklärt werden sondern mehr die praktische Anwendung in Verbindung mit Mikrocontrollern. Es gibt zwar noch jede Menge andere Schnittstellen, aber der RS485 Bus ist meiner Meinung nach unschlagbar was Preis, Platzbedarf und Leistung angeht.

## Anwendungsgebiete

Man kann den RS485 Bus überall dort einsetzen wo größere Datenmengen über längere Strecken übertragen werden müssen oder wenn mehrere Geräte miteinander vernetzt werden sollen, besonders dann, wenn die Kosten oder der Platz eine Rolle spielen, da die notwendigen Treiberbausteine kompakt und günstig sind. Er eignet sich besonders dazu um Mikrocontroller miteinander zu verbinden. Da Mikrocontroller meist einen sogenannten UART integriert haben ist der Aufbau eines Busses problemlos möglich.

## Vorteile

Der RS485 Bus hat gegenüber anderen Schnittstellen einige Vorteile.

- Hohe Datenraten. Je nach verwendetem Treiber bis weit über 2 MBit
- Mehrere Geräte an einem Bus. Üblicherweise 32, bei manchen Treibern auch 128
- Hohe Leitungslängen. Je nach Datenrate lassen sich mehrere hundert Meter erreichen
- Niedrige Kosten, da außer dem Treiber keine weiteren Bauteile benötigt werden
- Geringer Platzbedarf
- Unempfindlicher gegen Störungen
- Kein Protokoll Overhead wie z.B. bei CAN

## Nachteile

Die Nachteile des RS485 Bus sollen auch genannt werden, auch wenn es nicht sehr viele sind

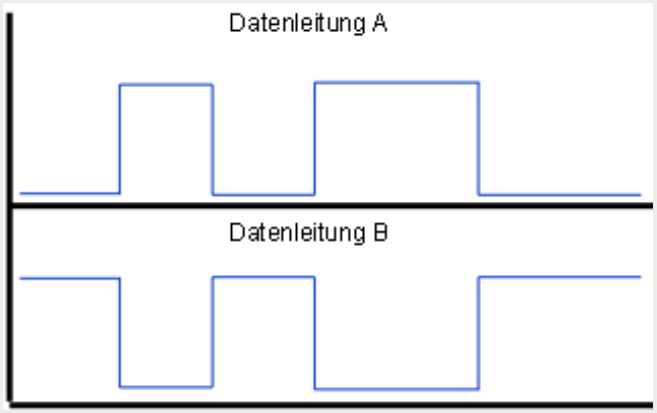
- Benötigt mindestens zwei Leitungen für einen Datenkanal
- Mit zwei Leitungen nur Half Duplex möglich
- Für Full Duplex werden vier Leitungen benötigt
- Für die Steuerung der Datenrichtung wird ein zusätzliches Steuersignal benötigt

## Funktionsweise

Der RS485 Bus ist wie der Name schon sagt nicht nur eine Verbindung zwischen zwei Geräten wie dies bei RS232 der Fall ist sondern es handelt sich um einen Bus, an dem mehrere Geräte gleichzeitig angeschlossen werden können. Er ermöglicht entweder einen Single Master / Multi Slave Betrieb oder einen Multi Master Betrieb. Ein Master ist ein Teilnehmer, der Befehle sowohl senden als auch empfangen kann, ein Slave erhält nur Befehle die er dann ausführt. Für Single Master / Multi Slave bedeutet es das nur ein Teilnehmer, z.B. ein PC, Befehle versenden darf. Alle anderen Teilnehmer, die am Bus angeschlossen sind, empfangen lediglich diese Befehle, führen sie aus und senden eventuell Daten zurück. Ein Slave beginnt aber nie ohne Aufforderung eine Kommunikation.

Im Multi Master Betrieb können mehrere Teilnehmer Befehle versenden. Da es passieren kann das zwei Teilnehmer gleichzeitig einen Befehl senden wollen kann es zu Kollisionen auf dem Bus kommen. Daher ist der Verwaltungsaufwand im Multi Master Betrieb höher. In diesem Artikel wird anfangs nur der Single Master / Multi Slave Betrieb beschrieben.

Bei RS485 handelt es sich um eine sogenannte differenzielle Schnittstelle. Das bedeutet, das für jedes Signal zwei Leitungen benötigt werden. Über die eine Leitung wird das Signal unverändert übertragen, auf der anderen Leitung wird das Signal invertiert. Wenn eine 1 übertragen werden soll hat die eine Leitung +5V, die andere Leitung 0V. Bei einer 0 hat die erste Leitung 0V, die andere +5V. Der Empfänger vergleicht die beiden Signale miteinander und gibt das original Signal aus. Durch dieses Verfahren werden Störungen nahezu eliminiert und man erreicht große Leitungslängen oder hohe Datenraten.




Die Verbindung zwischen den Teilnehmern kann entweder durch eine Zweidraht− oder durch eine Vierdrahtverbindung erfolgen. Verwendet man nur zwei Leitungen ist nur ein Half Duplex Betrieb möglich. Das bedeutet , das Daten nicht gleichzeitig in beide Richtungen übertragen werden können. Eine Übertragung erfolgt immer abwechselnd in die eine oder in die andere Richtung. Soll der Bus Full Duplex fähig sein, also Daten in beide Richtungen gleichzeitig Senden können, muss man für beide Datenrichtungen eigene Leitungen und Treiber verwenden. Für den Full Duplex Betrieb benötigt man also 4 Leitungen.

## RS485 vs. RS232

Die RS232 Schnittstelle war dazu gedacht um externe Geräte mit möglichst wenig Aufwand z.B. an einen PC anzuschließen. Man denke hier nur an die Maus oder das Modem. Die RS232 Schnittstelle kann nur zwei Geräte miteinander verbinden. Moderne PCs erreichen Datenraten von bis zu 256kBit über die RS232 Schnittstelle. Mit RS485 sind 2,5Mbit und mehr möglich, je nach verwendetem Treiber. RS232 kann maximal 12m überbrücken, mit RS485 kommt man mehrere 100 Meter weit. Da RS232 mit Pegeln von +12V und ™2V arbeitet muss der Treiber meist erst aufwändig diese Spannungen erzeugen. Viele der RS232 Treiber benötigen dafür externe Kondensatoren, was Platz kostet. Ein RS485 Treiber ist in einem 8 poligen Gehäuse untergebracht und begnügt sich mit +5V.

## RS485 vs. CAN

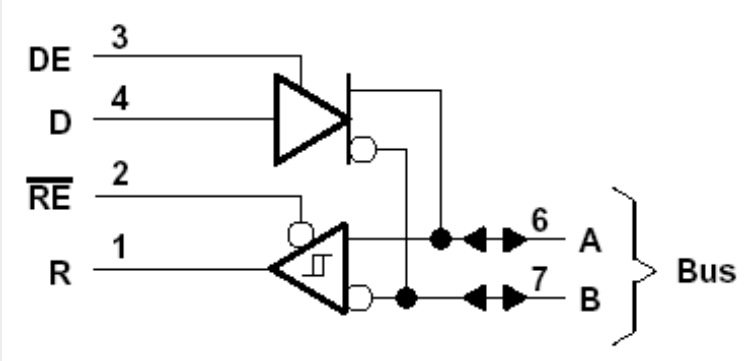
Der Vergleich zwischen RS485 und CAN sieht da schon etwas anders aus da die Hardware von CAN fast mit RS485 identisch ist. Auch bei CAN handelt es sich um einen differentiellen Bus mit hohen Datenraten, großen Leitungslängen und der Möglichkeit, mehrere Geräte mit dem Bus zu verbinden. Der wesentliche Unterschied zu RS485 ist das Protokoll. RS485 ist lediglich eine Definition wie die Hardware aussieht. Das Protokoll ist

frei definierbar. Dagegen ist bei CAN sowohl die Hardware als auch das Protokoll fest vorgegeben. Daraus resultieren große und teure Treiberbausteine und ein nicht unerheblicher Datenoverhead auf den Leitungen. In vielen Fällen ist dieser Overhead aber nicht notwendig, der Preis spielt eine Rolle oder der Platz für einen CAN Treiber ist nicht vorhanden. Es gibt auch Controller mit integrierter CAN Schnittstelle. Diese sind unter Umständen aber schwer zu bekommen und relativ teuer.

## Treiber

Wie auch für RS232 gibt es für RS485 eine große Anzahl an Treibernbausteinen. Diese unterscheiden sich in der Zahl der Sender und Empfänger, in der maximalen Datenrate oder der Anzahl der maximal möglichen Busteilnehmer. An dieser Stelle beschränke ich mich auf die Treiberbausteine mit nur jeweils einem Sender und Empfänger.

Die beiden Busleitungen werden mit A und B bezeichnet, wobei A nicht invertiert und B invertiert wird. Beide Leitungen sind sowohl mit dem Sender als auch dem Empfänger verbunden. Sowohl Sender als auch Empfänger haben einen Enable Eingang. Ist dieser Eingang deaktiviert befindet sich der entsprechende Treiber im Tri State Modus und belastet den Bus nicht. Je nach dem ob man senden oder empfangen will muss man den entsprechenden Eingang aktivieren. Da der Enable Eingang des Empfängers invertiert ist kann man ihn mit den Enable Eingang des Senders verbinden und mit einer einzigen Portleitung eines Controllers steuern. Dabei ist aber immer einer der beiden Treiber aktiv und kann nicht abgeschaltet werden.

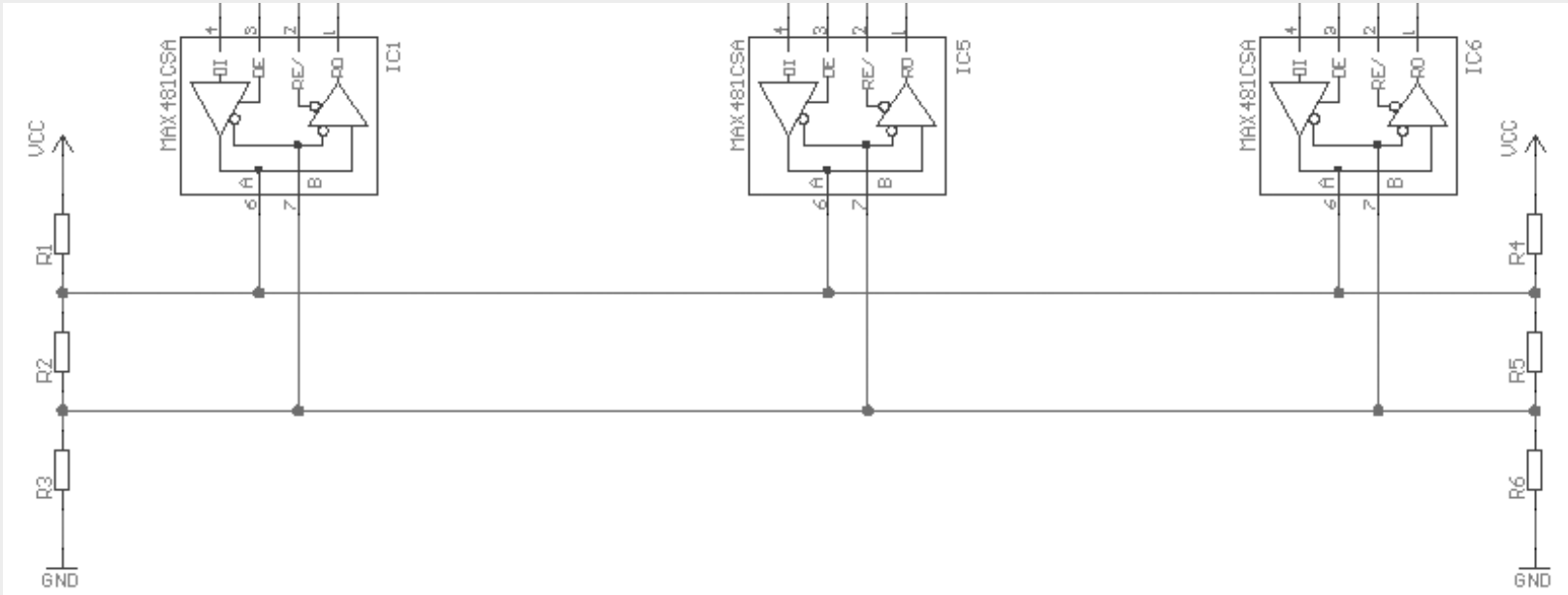



In dieser Darstellung sieht man den schematischen Aufbau eines RS485 Treibers. DE und RE sind die Enable Leitungen für Sender und Empfänger.
D ist der Datenausgang vom Controller. Man sollte D mit dem TxD Anschluß des UART's verbinden.
R ist der Dateneingang zum Controller. Er wird mit dem RxD Anschluß des UART's verbunden.
A und B sind die Verbindungen zum Bus.
Die Kreise deuten an welche der Anschlüsse invertiert werden

Die am häufigsten verwendeten RS485 Treiber sind der MAX485 von Maxim, der LTC485 von Linear Technology und der SN75176 von Texas Instruments
Alle diese Treiber sind zueinander Pin− und Funktionskompatibel, unterscheiden sich jedoch in der maximalen Datenrate und der Anzahl der Teilnehmer auf einem Bus. Je nach Ausführung kann z.B. der MAX485 mit bis zu 2,5MBit arbeiten, der MAX483 dagegen nur mit 250kBit. Verwendet man den MAX487 kann man bis zu 128 Teilnehmer an einen Bus hängen, mit dem MAX485 sind dagegen nur 32 Teilnehmer möglich.
Die hier vorgestellten Treiber sind alle in einem DIP 8 oder SO08 Gehäuse erhältlich.

## Der Bus

Die folgende Grafik zeigt wie der RS485 Bus aufgebaut ist.




Man sieht hier die beiden Leitungen des RS485 Bus. Eine Leitung verbindet jeweils die A−Anschlüsse der Treiber, die zweite Leitung verbindet die B−Anschlüsse.
Die Widerstände R1 bis R6 sind Abschlußwiderstände und müssen am Ende der Leitungen montiert werden. Bei geringen Entfernungen < 20m

werden folgende Widerstände eingesetzt:
R1, R3, R4, R6: 100kOhm
R2, R5: 1kOhm
Je nach Datenrate und Leitungslänge kann es notwendig sein andere Werte einzusetzen.

## Protokolle

Wie bereits erwähnt ist bei RS485 kein festes Protokoll definiert. Deshalb soll hier das am häufigsten verwendete Protokoll erklärt werden. Das hier beschriebene Protokoll ist nur eine von vielen Möglichkeiten und kann den entsprechenden Bedürfnissen angepasst werden.

Ein einfaches Protokoll für die Kommunikation enthält drei bis vier Komponenten und wird als Frame bezeichnet. Als erstes wird eine Adresse übertragen. In kleinen Systemen ist diese meist 8 Bit lang. Damit lassen sich bis zu 256 Geräte adressieren. Darauf folgt die Länge des Frames, entweder 8 Bit oder 16 Bit. Wird für die Framelänge 8 Bit definiert können maximal 256 Byte Daten übertragen werden, bei 16 Bit kann ein Frame 64 kByte enthalten. Danach folgen die eigentlichen Daten. Es müssen immer so viele Daten übertragen werden wie in der Länge angegeben ist. Nach den Daten kann noch eine Checksumme folgen. Diese ermöglicht es Übertragungsfehler festzustellen.

Dieses Diagramm Zeigt den Aufbau eines solchen Frames. Als erstes das Adressbyte mit gesetztem 9. Bit, anschließend die Paketlänge, in diesem Fall 8 Byte, danach die 8 Datenbytes und zum Schluß die Schecksumme. Bei allen Bytes werden 9 Bit übertragen aber nur bei der Adresse ist dieses Bit gesetzt.

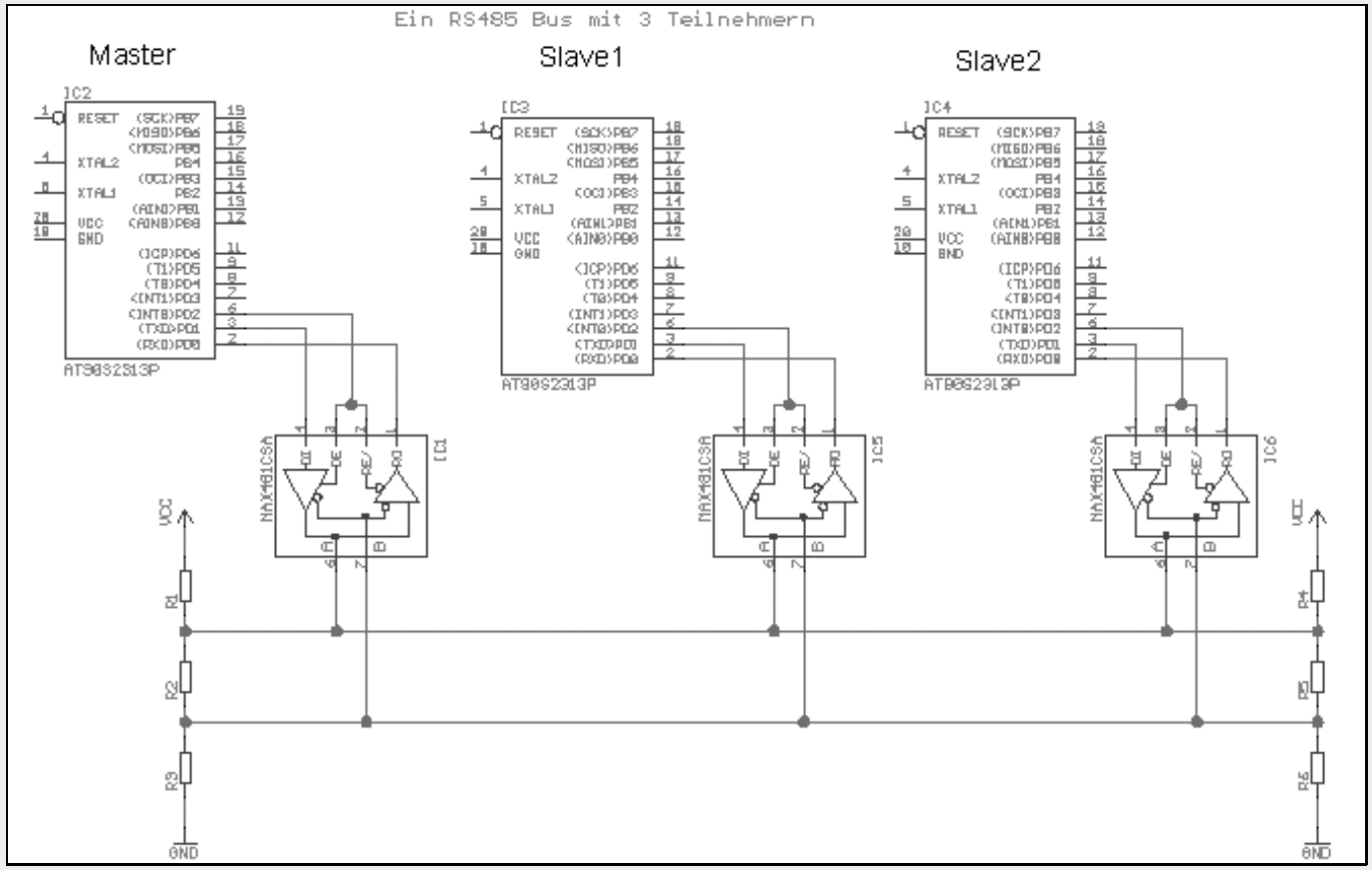
| Adressbyte | | | | | | | | | | Paletlänge | Byte1 | Byte2 | Byte3 | Byte4 | Byte5 | Byte6 | Byte7 | Byte8 | Checksumme |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Bit8 | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 | $08 | $48 | $61 | $6C | $6C | $6F | $20 | $3A | $29 | $5F |

Für die Übertragung der Daten werden nicht wie üblich 8 sondern 9 Datenbits übertragen. Das neunte Datenbit signalisiert ob es sich bei den übrigen 8 Bit um eine Adresse oder um Daten handelt. In einem RS485 Bus−System hat jeder Teilnehmer eine eigene Adresse. Diese ist im Normalfall 8 Bit lang. Möchte der Master eine Kommunikation zu einem Slave aufbauen sendet er als erstes die Adresse. Hierzu setzt er das neunte Bit um dem Slave zu signalisieren das es sich um eine Adresse handelt. Alle weiteren Bits werden ohne gesetztes neuntes Bit übertragen. Nach der Adresse wird die Anzahl der zu übertragenden Datenbytes übermittelt. Anschließend folgen die Daten

## Beispiel

Im folgenden Abschnitt wird anhand eines Beispieles die Funktionsweise des RS485 Bus erklärt. In diesem Beispiel wird als Controller der AT90S2313 von Atmel und als Treiber der MAX485 verwendet. Der MAX485 kann jedoch durch einen der oben genannten Treiber oder einen kompatiblen ersetzt werden. Als Controller kommt jeder in Frage der über einen sogenannten UART verfügt.

Dieser Schaltplan zeigt wie die Treiber an den Controller anzuschließen sind. Auf andere Schaltungsteile wie Quarz oder Spannungsversorgung wurde verzichtet um die Übersichtlichkeit zu erhöhen. Dieses Beispiel beschränkt sich auf den Single Master/Multi Slave Bus und auf den 2 Wire Bus.




In diesem Beispiel haben wir einen Master und zwei Slaves. Die Slaves haben die Adressen $01 und $02. Sie muß jedem Slave vorgegeben werden und muß einmalig auf dem Bus sein. Die Adresse $00 darf nicht verwendet werden. Sie hat eine Sonderfunktion. Der Master in einem Single Master

System hat keine Adresse. Im Ruhezustand sind alle Treiber, auch der des Masters, als Empfänger geschaltet. PD2 steuert die Datenrichtung der Treiber. Ist PD2=0 arbeiten die Treiber als Empfänger, wenn PD2=1 ist sind die Treiber als Sender geschaltet.

Möchte der Master einen Befehl senden schaltet er als erstes seinen Treiber auf Senden (PD2=1). Soll der Befehl an Slave´2 gehen sendet er als erstes die Adresse $02, und zwar mit gesetztem 9. Bit. Diese Adresse wird nun von beiden Slaves empfangen. Als erstes überprüfen die Slaves ob das 9. Bit gesetzt ist. Ist dies der Fall handelt es sich um eine Adresse und die Slaves vergleichen sie mit dem jeweils vorgegebenen Wert. Stimmt er überein erwartet der Slave die Länge des folgenden Datenpaketes. Alle Daten die nach der Adresse kommen dürfen das 9. Bit nicht mehr gesetzt haben.
Wenn der Master 8 Byte senden will überträgt er jetzt $08 als Längenangabe. Als nächstes überträgt der Master seine Daten. Zum Schluß berechnet der Master noch die Checksumme, die sich aus den übertragenen Daten berechnet, und übermittelt sie an den Slave. Danach schaltet er den Treiber wieder auf Empfang. Der Slave errechnet aus den empfangenen Daten nach der gleichen Formel die Checksumme und vergleicht sie mit der die er vom Master erhalten hat. Stimmen beide Werte überein war die Übertragung erfolgreich.

Damit der Master weiß das die Übertragung funktioniert hat muß der Slave nun eine Bestätigung schicken. Hierzu schaltet der Slave seinen Treiber auf Senden und überträgt zunächst seine eigene Adresse, wieder mit gesetztem 9. Bit. Danach sendet er einfach die Checksumme, die er selbst berechnet hat, ohne das 9. Bit zu setzen. Dann schaltet der Slave seinen Treiber wieder auf Empfang und wartet auf die nächste Übertragung.

War die Übertragung erfolgreich kann das nächste Paket gesendet werden. Ist dabei jedoch ein Fehler aufgetreten muß dieses Paket erneut gesendet werden. Da der Slave durch die Berechnung der Checksumme weiß das das empfangene Paket fehlerhaft war geht er davon aus das der Master als nächstes das gleiche Paket erneut sendet. Der Master vergleicht die vom Slave zurückgesendete Checksumme mit der von ihm berechneten und kann dadurch feststellen ob die Übertragung erfolgreich war. Im Fehlerfall wiederholt sich der Vorgang und der Master sendet das Paket erneut an den Slave.

Die Adresse $00 hat eine Sonderfunktion. Sie ermöglicht es dem Master einen Befehl an alle Teilnehmer zu senden. Auf diesen Befehl folgt aber keine Antwort der Slaves, da dies Kollisionen auf dem Bus zur Folge hätte.

Um zusätzliche Sicherheit in dieses Protokoll einzubauen kann man es natürlich noch erweitern. Wenn der Master z.B. Daten an eine Adresse versendet weiß er erst ob der Slave überhaupt gehört hat wenn dieser nach der Übertragung antwortet. Desshalb kann der Slave nach dem Empfang der Adresse eine Bestätigung, ein sogenanntes Acknoledge (kurz ACK), an den Master zurücksenden. Das kann ein bestimmer Wert oder seine eigene Adresse sein.