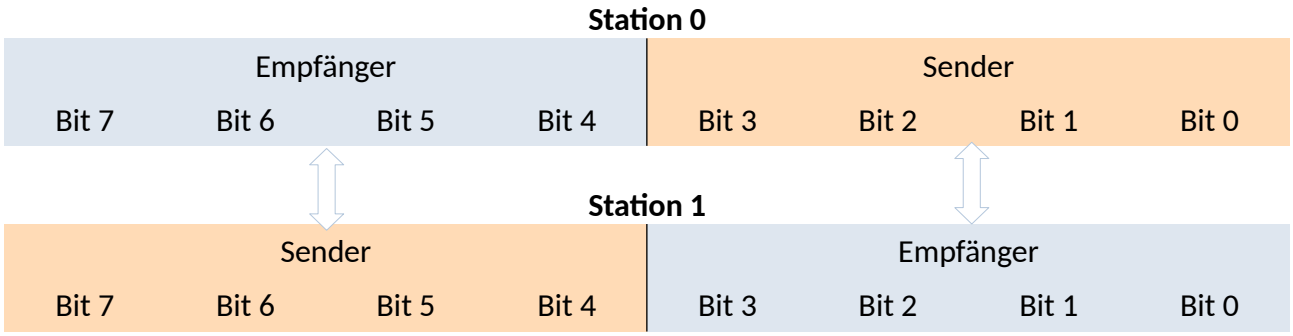
Protokoll V7

Tristan Sopauschke, Hans Schreiter

Aufteilung der Leitungen

Uns stehen 8 Leitungen und somit theoretisch 8 Bits zur Verfügung. Um eine gleichzeitige Vollduplex-Übertragung zu ermöglichen werden diese zunächst aufgeteilt, sodass jeder Station vier Leitungen zum Senden, und vier Leitungen zum Empfangen bereitstehen. Die beiden Stationen spiegeln sich dabei natürlich so, sodass Station 0 auf den Leitungen empfängt, auf denen Station 1 sendet, und umgekehrt:



So entstehen zwei identische Kanäle, die jeweils über 4 Leitungen/Bits verfügen.
Diesen 4 Bits werden dann noch jeweils eine spezifische Funktionalität zugewiesen, die im folgenden weiter erläutert werden:

Bit	3	2	1	0
Funktion	Taktleitung	ACK-Leitung	Datenbits	

Taktleitung: Taktgenerierung und -rückgewinnung

Auf der dedizierten Taktleitung wird kontinuierlich ein Taktsignal gesendet. Immer wenn zwei neue Datenbits bereitstehen, wird der Wert der Pegel der Taktleitung invertiert (also nach Muster 101010 etc). So weiß der Empfänger ob bereits neue Bits bereitstehen oder nicht. Es ist dadurch dann auch kein Problem, wenn der Empfänger die Leitungen häufiger pollen sollte als der Sender neue Daten bereithält. Diese Methode ist sehr robust und folgt dem guten alten KISS-Prinzp, ein Nachteil ist allerdings der durch die dedizierte Taktleitung entstehende Overhead.

ACK-Leitung: Empfang bestätigen oder Resends anfordern

Das ACK-Bit ermöglicht es dem Empfänger, dem Sender Rückmeldung zu geben, ob die gesendeten Daten korrekt empfangen wurden. Dies spielt später bei der Fehlerkorrektur eine entscheidende Rolle. Für diese Funktionalität reservieren wir eine dedizierte Leitung, da auf dieser Leitung eben genau in die umgekehrte Richtung zurückgesendet werden muss als auf den anderen 3 Bits.

Datenbits

Hier können zwei Bits beliebige Daten pro Takt übergeben werden. Über 4 Takte hinweg wird dann also ein ganzes Byte gesendet/empfangen, dass dann den darüberliegenden Schichten zur Verfügung gestellt wird.

Sicherungsschicht

Nun haben wir also erstmal die Möglichkeit, Bytes robust und im Vollduplex zu übertragen. Darauf wird nun eine vernünftige Verfahrensweise aufgebaut, um eine sichere Übertragung größerer Datenmengen zu erlauben.

Rahmenfindung und -erkennung

Die Daten werden stückweise in Form von Rahmen theoretisch beliebiger Größe übertragen. Zur Rahmenerkennung nutzen wir Steuerzeichen, welche jeweils 1 Byte groß sind und in den Datenstrom eingefügt werden.

Steuerzeichen	Wert	Funktion
BEGIN	0x01	Signalisiert Beginn eines Rahmens
END	0x02	Signalisiert Ende eines Rahmens
ESCAPE	0x03	Ignoriere folgende ESC Sequenz

Ein Rahmen sieht in der Regel dann so aus:
`BEGIN | (CHECKSUM) | Daten … | END`

Sollten in den Datenbytes selbst Bytes vorhanden sein, die zufällig dem Wert eines Steuerzeichens entsprechen, wird diesen beim Packen des Rahmens ein ESCAPE-Steuerzeichen vorangestellt, um diese als herkömmliche Daten kenntlich zu machen.

Fehlererkennung und -korrektur

Wie oben zu sehen, wird in jedem Rahmen zusätzlich direkt nach dem BEGIN-Zeichen eine Checksum eingefügt. Dabei handelt es sich um eine einfache, 1 Byte große Verdichtung aller im Datenbereich des Rahmens vorhandenen Bytes, die vom Sender erstellt wird. Der Empfänger erstellt dann nach vollständigem Empfang des Rahmens nach dem selben Verfahren eine eigene Checksum über den empfangenen Datenbytes und vergleicht diese mit der vom Sender gelieferten Checksum. Stimmen beide überein, wurde der Rahmen korrekt empfangen, sonst eben nicht.

Zur Fehlerkorrektur wird nun die ACK-Leitung und das Stop-And-Wait-Verfahren angewendet: Nachdem der Sender den Rahmen vollständig abgesendet hat, wartet dieser einen gewissen Zeitraum und liest die ACK-Leitung. Der Empfänger führt währenddessen den Checksum-Test durch und legt auf die ACK-Leitung entsprechend eine 1 oder 0; und verwirft außerdem inkorrekte Frames.
Erhält der Sender nun vor Ablauf des Zeitraumes eine 1 auf der ACK-Leitung, so ist gewiss, dass der Rahmen korrekt versendet wurde, und der Sender bereitet das Senden des nächsten Rahmens vor. Andernfalls wird der zuletzt versendete Rahmen erneut gesendet.
Die Fehlerkorrektur hat auch ihre Grenzen: Störungen der Taktleitung können eventuell zum Verlust der Synchronität von Sender und Empfänger führen.

Ende der Übertragung

Für das Ende einer Übertragung gibt es kein eigenes Steuerzeichen, stattdessen wird das Ende einer Übertragung jeweils vom Sender/Empfänger jeweils unabhängig voneinander daran erkannt, dass ein leerer Rahmen gesendet/empfangen wurde.

Konkretes zur Implementierung

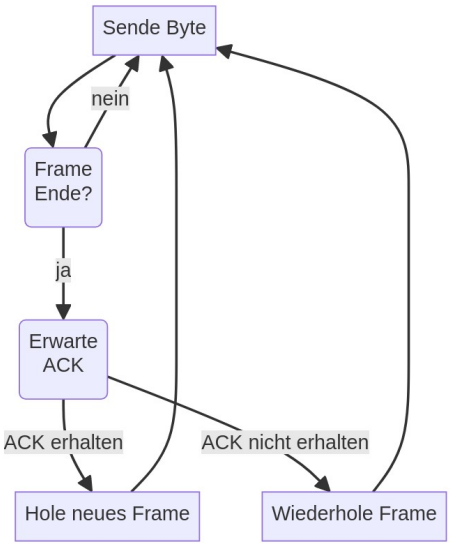
Das alles haben wir in einem einzelnen Programm implementiert, dass auf beiden Stationen ausgeführt wird, wobei lediglich jeweils festgelegt werden muss, auf welchem Kanal (also obere oder untere 4 Bits) gesendet wird. Es liest aus STDIN und schreibt nach STDOUT. Zusätzlich kann die geforderte Rahmengröße und Verzögerung pro Takt per Kommandozeile festgelegt werden.

Für den Sender und Empfänger sind jeweils eine Klasse implementiert, die als Automaten fungieren, also gemäß ihres Zustandes und des Inputs einen entsprechenden Output und Zustandsübergang finden.
Als Input akzeptieren diese Automaten jeweils den Zustand des Kanals und bedienen mit ihrem Output dann eben denselbigen.
In der Main-Funktion müssen dann nur noch die beiden 4-Bit-Kanäle aus den acht Leitungen extrahiert werden und an die Automaten übergeben werden. Deren Ausgabe jeweils 4-Bit große Ausgabe wird dann wieder zusammengeführt und auf die Leitungen geschrieben.

Die Automaten werden im virtuellen Unit-Test geprüft und ihre einwandfreie Funktionsweise so theoretisch garantiert. Auch das Packen und Entpacken von Rahmen ist durch automatische Tests abgedeckt.

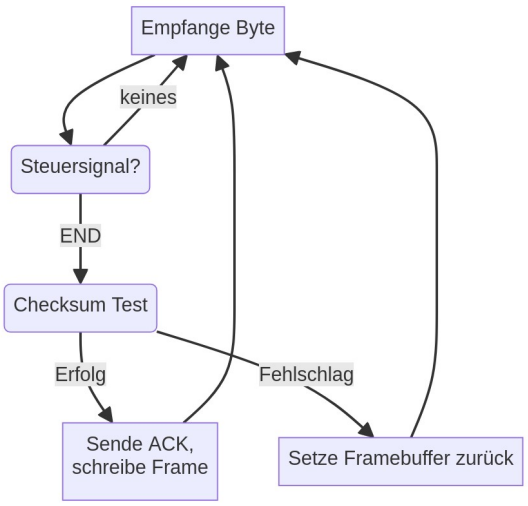
Der Senderautomat

Der Senderautomat sendet kontinuierlich Frames. Empfängt er anschließend kein ACK, versendet er den selben erneut. Sonst wird das nächste Frame gesendet.



Der Empfängerautomat

Der Empfängerautomat empfängt kontinuierlich Frames, schreibt aber nur jene in den Output, deren Korrektheit er mittels Checksum überprüfen kann.



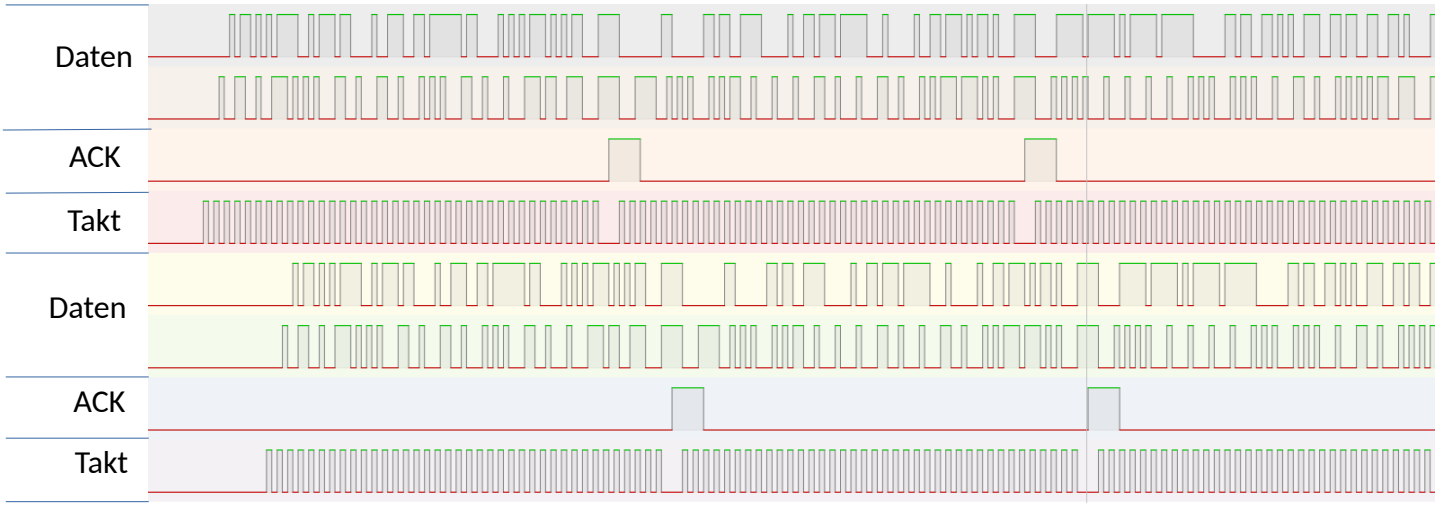
Übertragungsgeschwindigkeit

Grundsätzlich übertragen wir 2 Bits Daten pro Tick, das heißt 0,25 Bytes/Tick. Das Übertragen von 16 Bytes dauert etwa 2,5 Sekunden, was 6,4 B/s entspricht, inklusive Overhead durch Steuerzeichen.

Pro Rahmen fallen fest 3 Bytes Overhead durch BEGIN/END/CHECKSUM Steuerzeichen an. Zusätzlich müssen vor Datenbytes deren Wert mit dem von Steuerzeichen übereinstimmt, mit einem ESCAPE-Steuerzeichen versehen werden. Es gibt drei Steuerzeichen und somit drei Bytewerte die mit Escapes versehen werden müssen; gehen wir im folgenden von komplett zufälligen Datenbytes und nehmen an dass $3/256$ aller Datenbytes mit ESCAPES versehen werden müssen. Die Formel für den Overhead pro Rahmen lautet also $3+\frac{3}{256}*n_{framesize}$.
Desto höher die Größe der Frames gewählt wird, desto geringer also der Overhead pro Datenbyte.
Der Overhead pro Datenbyte ist somit $\frac{3+\frac{3}{256}*n_{framesize}}{n_{framsize}}$. Die Rahmengröße ist pro Übertragung und Überträgungsrichtung frei wählbar, in der Regel haben wir bis jetzt eine Größe von meist 32 Bytes gewählt. Somit ergibt sich ein konkreter Overhead von $\frac{3+\frac{3}{256}*32}{32} \approx 0.105$, also ungefähr 10 Prozent.

Der tatsächliche Datendurchsatz beträgt also etwa $6{,}4-6{,}4*0.1 \approx 5{,}78\,\mathrm{B/s}$.

Sicht im Logic Analyzer



Hier ist eine Vollduplex-Übertragung zu sehen, die in beide Richtungen die selbe Datei (mit gleicher Framegröße = 16) versendet.