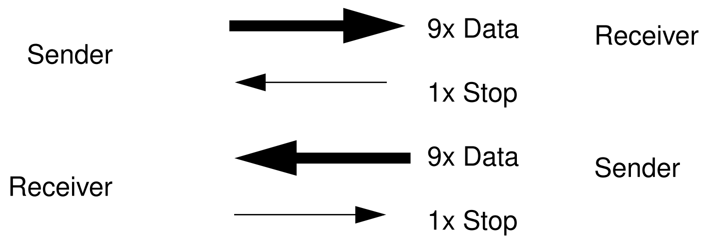
Das MANNA-Link-Protokoll

Der MANNA-Link ist eine Punkt-zu-Punkt-Verbindung. Alle MANNA-Geräte haben den gleichen Link Interface. Mit einem physikalischen Link können nur zwei Geräte verbunden werden. Zum Zusammenschalten von mehreren Geräten innerhalb eines Systems, sind Crossbars mit mehreren Link-Verbindungen vorhanden. Sie können intern zwei beliebige Ports zu einem Paar zusammenschalten. Es ist möglich, jedes Gerät mit einem beliebigen anderen Gerät zusammenzuschalten. Das System kann auf jede mögliche Weise verkoppelt werden.

Durch den MANNA-Link sind die MANNA-Geräte unabhängig vom Netz, Struktur und Größe, und umgekehrt. Die physikalischen Link-Protokolle sind taktsynchron und so einfach und allgemein wie möglich.

Alle Links sind in der Lage, zur gleichen Zeit Daten und Kontrollinformationen in beide Richtungen zu übertragen (bidirectional und full-duplex). Alle Geräte können simultan senden und empfangen. Alle Richtungen eines Ports besitzen einen 9 Bit breiten Kanal für die Daten und Kontrollinformationen zwischen Sender und Empfänger und einen1 Bit breiten Kanal (“stop” Signal) für eine weiche Flußregelung (soft flow control) in die entgegengesetzte Richtung. Auf der Empfängerseite eines Links ist ein Puffer. Solange genug Platz im Puffer vorhanden ist, kann der Sender mit voller Geschwindigkeit senden (ein 9-Bit Wort pro Taktzyklus). Wird der Platz im freien Puffer knapp, setzt der Empfänger das Signal “stop” in Gang (0). Dann muß der Sender so schnell wie möglich aufhören zu senden. Bei aktueller Implementierung muß der Sender innerhalb von vier Zyklen aufhören. Die Signale auf dem physikalischen Link lauten:



9x Daten: Output des Senders -> Input des Empfängers

Die Signale übertragen Daten und Kontrollinformationen. Das neunte Kontrollbit benötigt man, um Daten und Kontrollinformationen gleichzeitig mit den anderen 8 Signalen senden zu können. Ist “control” eingestellt (1), übertragen die Datensignale Routing- oder Kontrollinformationen. Wenn nicht, dann haben die Datensignale ein Byte Data (a byte data), das an den Endempfänger weitergegeben werden muß.

Die Befehle und Daten in den 9 Datensignalen sind wie folgt kodiert:

```
8 7 6 5 4 3 2 1 0       (Hex)
---------------------------------------------------------------------
0 a b c d e f g h       0x0xx       Daten Byte (abcdefgh)
1 1 1 1 1 1 1 1 1       0x1ff       FILLER (keine Daten,keine Kontrolle zum

Senden)
1 1 1 1 0 a b c d       0x1ex       Routing zum Port (abcd: 0..15)
1 1 1 0 1 x x x x       0x1dx       Abort
```

```
1 1 1 0 0 x x x x       0x1cx       Close
1 1 0 1 1 x x x x       0x1bx       CRC Kode (a = 1 -> OK, a = 1 -> Error)
1 1 0 1 0 a b c d       0x1ax       Übertragungsfehlerkode (abcd)
1 0 0 0 0 a b c d       0x10x       Anwender   (Software)   Befehle   (abcd:
0..15)
-alle anderen Kodes-                noch nicht definiert,wie Daten behandelt
```

Nicht jedes Gerät führt alle Befehle aus. Führt ein Gerät einen Befehl nicht aus, sendet es ihn zum nächsten (wie Daten) und führt einen NOP aus.

Füllbefehle werden erzeugt, wenn der Sender nichts zu senden hat oder vor dem Senden noch warten muß. Füllfelder können durch den Empfänger zerstört werden. Sie werden nicht gepuffert und nicht in den Speicher geschrieben.

Datenwörter werden an den Endempfänger weitergegeben. Kein Transportgerät schaut sich die Datenwörter an oder modifiziert sie. Datenwörter haben nur für die Terminalgeräte eine Bedeutung (Link Interface des CPU). Sie können nur für andere Daten Daten oder Kontrollinfomationen sein. Beim Link-Protokoll kommt es nicht darauf an, was sie sind.

Routing-Befehle können nur von Crossbars ausgewertet werden. Alle anderen Geräte können sie nur weiterleiten. Hat der Port noch keine logische Verbindung, nimmt er den nächsten Routing-Befehl und benutzt ihn, um zu einem Port zu steuern 0..15. Der erste Leitwegbefehl eines Crossbars wird vernichtet. Alle anderen Befehle oder Daten werden, bis ein Schließbefehl ankommt, weitergeleitet. Jeder Crossbar, in einer logischen Verbindung, vernichtet einen Routing-Befehl. Benötigt eine logische Verbindung n Crossbars, müssen n Routing-Befehle am Anfangsetikett der Nachrichten vorhanden sein.

Schließbefehle braucht man, um logische Verbindungen zu beenden (geöffnet durch Leitwegbefehle). Schließbefehle werden nicht vernichtet. Ein Schließbefehl reicht aus, um mit n Crossbars eine gesamte logische Verbindung zu schließen. Der Schließbefehl wird an den Endempfänger weitergeleitet.

Abort-Befehle funktionieren wie Schließbefehle. Nur beim Software gibt es da einen Unterschied. Das Link-Protokoll behandelt beide Befehle geichermaßen.

CRC (Cycle redundant Code - check-) -Befehle sind nur für Terminalgeräte von Bedeutung, alle anderen Geräte leiten diese Befehle nur weiter. Stößt der CPU beim Sender-Endgerät (ein CPU Interface) einen CRC-Befehl aus, sendet der Interface zuerst den Befehl und dann den errechneten CRC - als Daten - als aktuelle Nachricht. Erhält das Empfänger-Endgerät (ein anderer CPU Interface) einen CRC-Befehl, speichert es die nächsten n Wörter (Größe des CRC-Kodes) und vergleicht, um eventuelle Fehler feststellen zu können, den Wert mit dem errechneten CRC der aktuellen Nachricht. Der CPU kann den Stand des Vergleiches ablesen. Die Übertragungsfehlerbefehle werden von einem Transportiergerät, das einen Übertragungsfehler festgestellt hat, generiert. Die Fehlerart wird in den 4 unteren Bits kodiert. Alle anderen Geräte leiten die Befehle nur weiter, bis sie den Endempfänger erreicht haben. Der CPU kann den Fehler-Kode erhalten und auswerten.

Die User Commands werden nur vom sendenden Endgerät an das empfangende Endgerät ausgewertet. Sie sind ohne Bedeutung für das Link-Protokoll, wie alle anderen Befehlen, die noch nicht definiert sind.

STOP#: Input des Senders <<- Output des Receivers

STOP wird zur Durchführung einer weichen Datenflußsteuerung benutzt. Ist STOP# inaktiv (1), kann der Sender bei jedem Taktzyklus Daten oder Kontrolle (data oder control) senden. Wird STOP# aktiv(0), muß der Sender so schnell wie möglich aufhören, aber nicht unbedingt sofort (soft handshake), Daten zu senden. Wird STOP# aktiv (0), ist der Empfänger-Puffer (receiver buffer) fast voll, aber er hat noch Platz für mindestens 4 weitere Bytes. Bei einigen Devices ist das STOP#-Signal konfigurierbar. Die Reichweite beträgt zwischen 4 und 24 freien Bytes im Input-Puffer.

Hardware Hinweise

Die Ports arbeiten synchron, mit einer maximalen Frequenz von 50 MHz (50 MByte/s). Die Ports aller Geräte müssen den gleichen Takt haben - Frequenz und Phase -. Es gibt Geräte, mit denen man das System an verschiedene Takte koppeln kann. Sie arbeiten mit asynchronen FIFOs und wahlfreiem Serientransfer. Alle Signale können ihren Zustand nur nach einer aufsteigenden Taktflanke (clock rising edge) verändern und müssen bis zur nächsten Taktperiode stabil bleiben:

```
Setup-time:             min. 3 ns
Hold-time:              min. 4 ns
clock-to-out:           4 to 17 ns
```