Der Link Interface Chip

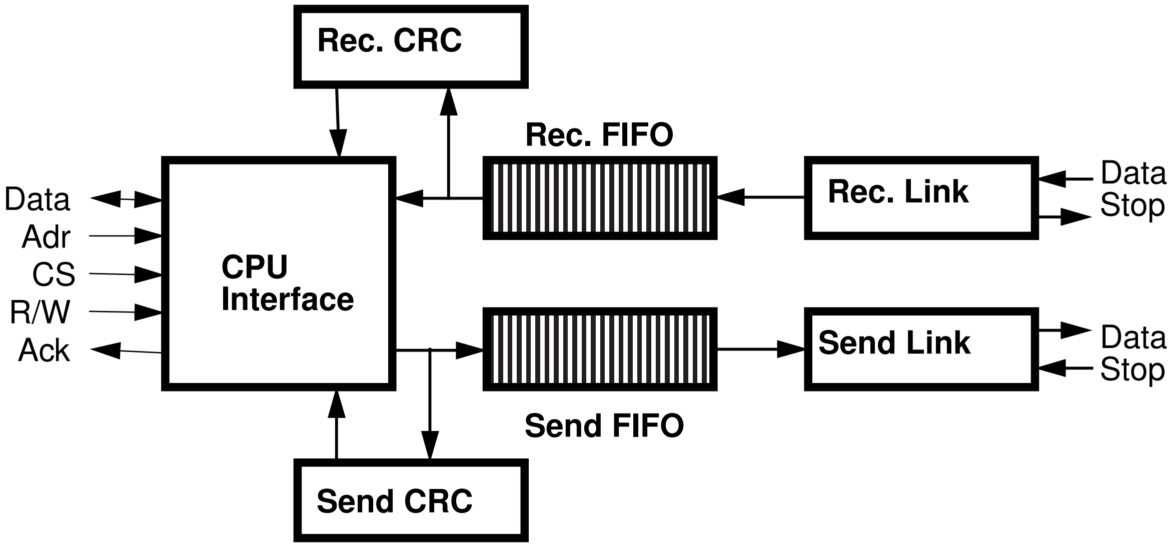
Dieser Interface kann leicht eine 64-Bit CPU oder DMA mit dem MANNA-Link-Protokoll (9 Bit breit),wie im MANNA-Link-Protokoll-Bericht beschrieben, verbinden. Mit diesem Gerät ist es einfach, einen beliebigen Knoten mit einem MANNA-Interconnection Network,das aus zwei bis tausend Knoten, Crossbars und Repeatern bestehen kann, zu verbinden. Außer der Protokoll-Konvertierung und dem Datentransfer kann der Link Interface, um für Zuverlässigkeit zu sorgen, einen CRC-Check-Code für eine kontinuierliche Gruppe von Daten (z.B. eine Nachricht) berechnen.

Das Gerät unterstützt die gesamte Funktionalität des MANNA-Link-Protokolls. Es besitzt je einen Kanal zum Senden und zum Empfangen von Daten (Nachrichten). Beide Kanäle sind voneinander unabhängig und können simultan arbeiten. Jede Richtung (senden und empfangen) hat einen Puffer, der bis zu 32 64-Bit-Wörter (Eingänge) fassen kann. Der Puffer ist nützlich, um beide Seiten des Link Interface (CPU und Manna-Link), die unterschiedliche Datenübertragungsraten besitzen, miteinander zu verbinden. Beide Seiten sind taktsynchron (derselbe Takt) und können pro Takt ein Wort übertragen. Der Unterschied in der Datenübertragungsrate ergibt sich aus der Breite des Datenweges (64 und 8 Bit breit).

Auf der CPU-Seite besitzt der Link Interface einen sehr einfachen Register-Interface, der eine Anforderung pro Taktzyklus im Pipeline-Modus ausführen kann. Die Antwort auf eine Anforderung folgt jeweils zwei Taktzyklen später. Ist genug Platz im Send-Puffer, dann kann der Interface pro Zyklus ein 64-Bit-Wort nehmen, um Daten zu senden. Sind Daten im Receive-Puffer vorhanden, ist der Interface in der Lage, pro Zyklus ein 64-Bit-Wort zum Empfangen von Daten zur Verfügung zu stellen. Die CPU-Seite hat eine maximale Übertragungsrate von 400 Mbytes/Sek und eine Dauertransferrate (sustained transfer rate) von 50 Mbytes/Sek.

Die MANNA-Link-Seite des Interface hat eine (Dauer-) Transferrate von 50 Mbytes/Sek (siehe MANNA-Link-Protokoll-Bericht). Wegen des Geschwindigkeitsunterschiedes kann die Maximumrate auf der CPU-Seite nicht lange eingehalten werden. Darum kann dem Send-Puffer der Platz ausgehen, und der Empfangspuffer (Receive-Puffer) wird geleert. Vor jeder Übertragung oder jedem Übertragungsblock sollte die CPU den Status des Puffers lesen, um herauszufinden, ob genug Daten oder Platz vorhanden sind.

Block Diagramm



Hardware-Überlegungen

Bei jedem Taktzyklus werden alle Signale gesammelt und jeweils einen Zyklus später ausgewertet. Der Link Interface Chip benötigt einen Zyklus, um auf einen CPU- Eingang zu reagieren. Dann kann er pro Taktzyklus eine Anforderung bearbeiten.

CPU-Seite (gemäß der CPU-Spezifikation):

```
input Setup:
       Adresse:       2 ns(-> 0)
       Control:       6 ns
       Data:          1 ns
input Hold
       Adresse:       0.6 ns
       Control:       2.5
       Data:          2.5

CLK-to-output:        1.5 .. 11.5 ns
            z->data:  2.5 .. 11.5
            1/0->0/1: 2.5 .. 11
            data->z:  1.5 .. 11.5
```

Link-Seite (gemäß dem MANNA-Link-Protokoll):

```
input Setup:         -1 ns
input Hold :         3.2 ns
CLK-to-output        3.3 .. 11.5 ns
```

Chip Select (CS#) (Input):

Der Link Interface beginnt auf der CPU-Seite keinen Arbeitsgang, wenn CS nicht aktiv ist (Zustand 1), und alle anderen Kontrollsignale auf der CPU-Seite werden ignoriert. Ist CS aktiv (0), dann klinkt der Link Interface die Adressen-Signale und die W/R-Signale im gleichen Zyklus ein. Zwei Zyklen später bearbeitet er die Anforderung (nimmt oder steuert Daten).

Write/Read# (W/R#) (Input):

Wird die CPU-Seite aktiviert (siehe CS#) und W/R# ist 0 (Read), dann steuert der Link Interface, zwei Zyklen nach der Anforderung, den Datenbus (data-bus) mit den ausgewählten Daten (siehe Addrs). W/R# ist 1 (write), wenn die CPU-Seite aktiviert wird. Zwei Zyklen nach der Anforderung nimmt der Interface die Daten vom Datenbus. Der Link Interface kann pro Taktzyklus eine CPU-Anforderung bearbeiten (nach zwei Pipeline-Stufen).

Adresse 3.5 (adr3, adr4, adr5) (Input):

Durch adr3..5 wird ein internes Register oder ein Daten-Puffer ausgewählt, das geschrieben oder gelesen werden soll (siehe CS# and R/W#).

Acknowledge (ack#) (Output):

Mit dem Signal Ack signalisiert der Link Interface die Beendigung jeder CPU-Anforderung. Ack wird für einen Taktzyklus, zwei Zyklen nachdem CS# entdeckt wurde, aktiviert (0): zur gleichen Zeit, wenn die Daten genommen oder gesteuert werden.

Data 0..73 (in/out) :

Mit diesen Signalen bekommt oder steuert der Link Interface die Daten und Paritäten als Antwort auf jede CPU-Anforderung (siehe CS, RW & adr). Der Link Interface steuert bei jedem Lesevorgang (read operation) alle 64Bit-Daten, obwohl für manche Operationen nur einige von ihnen relevant sind (siehe Register-Modell). Bei jedem Schreibvorgang (write operation) werden alle 64 Bit-Daten und 8 Paritäten genommen, obwohl für manche Operationen nur einige benutzt werden. Der Rest wird verworfen. Die 8 Paritätsbits entsprechen den 8 Daten-Bytes: P0 entspricht Byte 0 (D0..D7), P1 entspricht Byte 1 (D8..D15), ... P7 entspricht Byte 7 (D56..E63). Beim Write-Request kontrolliert der Link Interface die Paritätsbits nicht. Beim Senden von Daten werden sie nur zusammen mit anderen Daten-Bits gesendet. Beim Schreiben von internen Kontroll-Registern werden sie ignoriert.Die mit jedem Daten-Wort empfangene Parität wird nur bei einem Read-Request, der den empfangenen Daten entspricht, an die CPU weitergegeben. Durch die Read-Requests des internen Registers erzeugt der Link Interface für jeden Byte eine gerade Parität.

Link_D_O 0..8, link_D_I 0..8, stop_I, stop_O:

Siehe Manna-Link-Protokoll-Bericht und Manna-Crossbar-Bericht

Halt, Stop Select:

Siehe Manna-Crossbar-Bericht

Takt (Input):

Bis zu 50 MHz Taktsignal

Reset# (Input):

Active Low Power-On Reset-Signal für den gesamten Interface

Status 0..25 (Output):

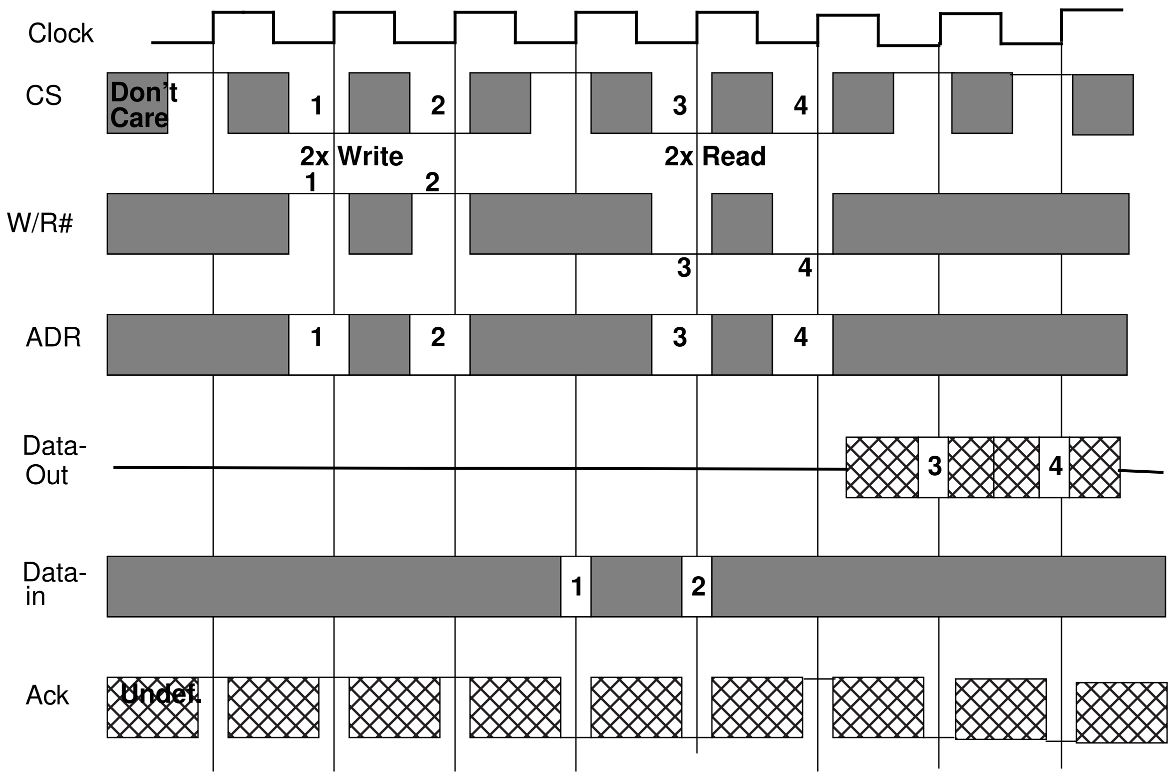
Einige der internen Statussignale des Link Interface werden zu den Pins gesteuert. Die Signale repräsentieren den momentanen Zustand des Interface und können sich nach jedem Taktangeber (clock rising edge) dynamisch verändern. Diese Signale haben immer gültige Daten, die von einem Arbiter oder einer Unterbrechungssteuerung (interrupt controller) benutzt werden können. Das Format dieser Signalgruppe ist noch nicht definiert, aber es existieren zumindest die folgenden Signale

```
6..0 = Receive Status:
          1 Bit 0 >= 8  Einträge im FIFO
          2 Bit 1 >= 16 Einträge im FIFO
          4 Bit 2 >= 20 Einträge im FIFO
          8 Bit 3 >= 24 Einträge im FIFO
         10 Bit 4 = FIFO voll
         20 Bit 5 = FIFO leer
         40 Bit 6 = Stop aktiv
13..7 = Send Status:
          1 Bit 7 >= 8  Einträge im FIFO
          2 Bit 8 >= 16 Einträge im FIFO
          4 Bit 9 >= 20 Einträge im FIFO
          8 Bit 10 >= 24 Einträge im FIFO
         10 Bit 11 = FIFO voll
         20 Bit 12 = FIFO leer
         40 Bit 13 = Stop aktiv

14 = Comand am Kopf der ankommenden Daten
15 = Data am Kopf der ankommenden Daten

20..16 = Belegte Einträge im Receiver Fifo
25..21 = Belegte Einträge im Send Fifo
```

Timing-Beispiel:



Register-Modell:

Das Register-Modell des Link-Interface hat zwei verschiedene Seiten: einen zum Lesen und einen zum Schreiben. Der Interface hat 8Register zum Schreiben und 8 zum Lesen. Sie sind nicht identisch, aber haben dieselbe Adresse. Der Link Interface kann bei jedem Zugriff bis zu 8 Bytes (64 Bits) lesen oder schreiben.

Außer der Register gibt es einen Send-Puffer (Send-Fifo) und einen Receive-Puffer (Rec-Fifo), um Daten (Nachrichten) zum Netz zu senden oder vom Netz zu empfangen (z.B. von anderen Knoten). Jeder dieser Fifos hat 32 64-Bit-Wort-Eingänge, aber nur 31 können zur gleichen Zeit belegt werden (einer muß immer frei bleiben). Zum Senden von Daten und Befehlen benutzt der Link Interface den gleichen Send-Fifo. Das gilt auch für das Empfangen von Daten oder Befehlen. Zum Erkennen bzw. Interpretieren von Daten und Befehlen benutzt er ein Zusatzbit in den Fifos.

Die Daten werden, so wie sie sind, vom Sender-Link-Interface (vom Send-Fifo) zum Receiver-Link-Interface, der sich in einem anderen (oder demselben) Knoten befindet, übertragen. Der Receiver ist in der Lage, vom Receive-Fifo als Daten zu lesen, was der Sender in Form von Daten in den Receive-Fifo schreibt. Der Link-Interface überträgt immer alle 8 Byte-Daten pro Wort. Doch die unbeschriebenen Bytes sind auf der Receiver-Seite nicht definiert. Die Befehle werden anders behandelt. Manche Befehle haben Bedeutung für einige Bausteine im Netz. Sie können von anderen Bausteinen im Verbindungsnetz oder durch den Link Interface (Sender- und Receiver-Link-Interface) interpretiert, ausgeführt, geschaffen oder zerstört werden. Einige Befehle werden vom Netz konsumiert. Sie erreichen den Receiver-Link-Interface nicht. Andere Befehle werden vom Netz erzeugt, so daß der Receiver-Link-Interface Befehle erhalten kann, die nicht vom Sender-Link-Interface, sondern vom Netz gesendet wurden. Aber meistens kommen Befehle, die im Send-Fifo geschrieben sind, im Receive-Fifo (in einem anderen Knoten) an. Die CPU kann 4 oder 8 Befehle (einen pro Byte) in einem Schreibzugriff schreiben (zum Senden von Befehlen). Zum Lesen der empfangenen Befehle, bekommt der CPU jeweils einen pro Zugang, da sie reihenweise angeordnet sind. Das ist wichtig für die Befehle, die vom Netz ausgeführt werden. Byte 0 wird zuerst gesendet; der Befehl, der in Byte 0 steht, wird zuerst ausgeführt, dann Byte 1, 2, 3,...7.

Die definierten Befehle lauten:

Name	Hex	link-sender	Crossbar	repeater	link-rec
FILLER	0xff	NOP & Pass	NOP	NOP & Pass	delete
ROUTE	0xex	NOP & Pass	connect to x & consume	NOP & Pass	NOP & Pass
ABORT	0xdx	NOP & Pass	disconnect & Pass	NOP & Pass	NOP & Pass
CLOSE & Pass	0xcx	NOP & Pass	disconnect	NOP & Pass	NOP & Pass
CRC	0xbx	NOP & Pass	NOP & Pass	NOP & Pass	NOP & Pass
ERROR	0xax	NOP & Pass	NOP & Pass	NOP & Pass	NOP & Pass
USER	0x0x	NOP & Pass	NOP & Pass	NOP & Pass	NOP & Pass
All Others	0x10.. ..0x9f	NOP & Pass	NOP & Pass	NOP & Pass	NOP & Pass

ERROR-Befehle repräsentieren Übertragungsfehler, die durch einen Baustein im Netz festgestellt werden. Erkennt ein Baustein solch einen Fehler, fügt er einen ERROR-Befehl im Datenfluß ein.

FILLER-Befehle haben keinen Arbeitsgang. Sie werden im Netz erzeugt und zerstört, aber erreichen den CPU-Interface nicht.

CRC kann verwendet werden, um die nächsten Bytes für eine Nachricht als CRC zu signalisieren.

Die normale Sequenz von Befehlen und Daten sieht etwa folgendermaßen aus:

Senden:

route* data* [crc data] close

Empfangen:

data*   [crc   data]   close

Transport (im Netz):

(route | filler)* (data | filler)* [crc filler* data filler*] close

Zu Beginn einer Nachricht sollte der Sender pro Crossbar zuerst einen Leitwegbefehl (route command) auf den Weg zum Receiver (0 oder mehr Leitwegbefehle) schicken, dann einige Daten (0 oder mehr Wörter), wahlweise einen CRC-Befehl mit der entsprechenden Summe, und zum Schluß einen Schließbefehl (close command).

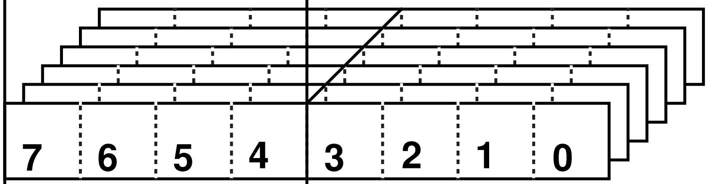
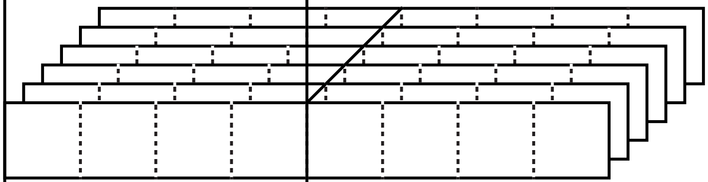
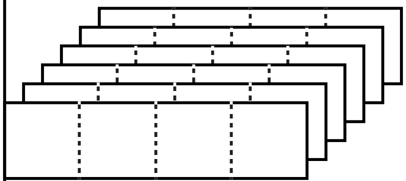
Der Receiver erhält die Leitwegbefehle nicht, da sie vom Crossbar konsumiert werden (einer pro Crossbar). Nur wenn der Sender mehr Routing-Befehle sendet als der Pfad Crossbars hat, erhält der Receiver den Rest.

Der Benutzer kann ein anderes Protokoll mit anderen Befehlen definieren, aber er sollte zumindest die folgenden verwenden:

route*   data*   close

Schreib-Register-Modell:

A2 == A1 == A0 == 0

AAA 6 5 4	**Hex**	**Name**	
0 0 0 64-bit	0x00 DAT	Send- Data	
0 0 1 64-bit	0x10 CMD	Send- 8-CMDs	
0 1 0 32-bit	0x20 4CM	Send 4-CMDs	
0 1 1 0-bit	0x30 NOP	NOP	**No Data**
1 0 0 0-bit	0x40 RST	Reset- All	**No Data**
1 0 1 0-bit	0x50 RSC	Reset- Send CRC	**No Data**
1 1 0 0-bit	0x60 RRC	Reset- Rec CRC	**No Data**
1 1 1 0-bit	0x70 NP1	NOP	**No Data**

Einige Register benötigen 8-Byte-Daten für jeden Schreibvorgang (Double). Einige benötigen nur 4 Bytes (Long) und andere keine Daten. Wird weniger als die erforderliche Datenlänge geschrieben, bleiben einige Bytes nicht definiert. Dies kann zu fehlerhaftem Verhalten führen. Werden mehr als die erforderlichen Daten geschrieben, dann werden nur die niederwertigen Bytes genommen, und der Rest verworfen, z.B. die NO-Data-Register können als Byte oder Wort geschrieben werden. Daten und ihre Länge spielen hier keine Rolle. Alle Schreibadressen sind "alligned" auf 64 Bits (8 Bytes). Eine schlechte Ausrichtung (misalignment) kann zu fehlerhaftem Verhalten führen.

Send-Daten (8 bytes, adr: (double *)0x00):

Diese Adresse ist das Ende des Send-Fifos. Die hier geschriebenen Daten werden am Ende des Send-Fifos gespeichert. Der Link-Interface nimmt die Daten von dort und sendet sie weiter zum Netz. Die CPU kann schneller schreiben als der Link-Interface in der Lage ist, Daten zu übertragen. Deshalb kann es geschehen, daß der Fifo voll wird. Im Fifo ist nur Platz für 31 Einträge (64 Bit Daten oder Befehle). Bei einer Kapazitätsüberschreitung des Fifos (fifo overflow) (beim Schreiben des 32sten Stelle) werden alle Daten im Fifo gelöscht, so daß der Fifo wieder leer ist. Die CPU sollte, um eine Kapazitätsüberschreitung zu vermeiden, vor dem Schreiben eines Blockes, das Zustandsregister des Fifos lesen, um zu bestimmen, wieviel Platz noch vorhanden ist.

Send-8-CMDs (8 bytes, adr: (double *)0x10):

Diese Adresse stellt auch das Ende des Send-Fifos (wie die Send-Daten) dar, aber der Eintrag wird als Befehl markiert (siehe Send-Daten). Der Eintrag wird in Form von 8 Befehlen gesendet, zuerst Byte 0 und zuletzt Byte 7. Wenn weniger als 8 Befehle erforderlich sind, muß die CPU die unbenutzten Bytes mit FILLER (NOP) -Befehlen füllen. Werden weniger als 8 Bytes geschrieben, sind einige von ihnen nicht definiert. Dies kann zu fehlerhaftem Verhalten des Verbindungsnetzes (interconnection network) führen.

Send-4-Cmds (4 bytes, adr: (long *)0x20):

Diese Adresse bedeutet soviel wie "Sende-8-CMDs", aber nur die unteren 4 Bytes (0.. 3) eines Wortes sind relevant, der Rest (4.. 7) wird automatisch mit FILLERn (NOPs) gefüllt.

Reset-all (0 Bytes (just write anything here), adr: (char *)0x40):

Schreibt die CPU etwas in diese Adresse, führt der Link Interface einen Reset wie den Power-On Reset durch. Die Daten und Kontrollinformationen im internen Register und Puffer gehen verloren.

Reset-Send-CRC (0 Bytes, adr: (char *)0x50):

Schreibt die CPU etwas in diese Adresse, löscht der Link Interface den CRC des Senderkanals (z.B. den Beginn der Nachricht).

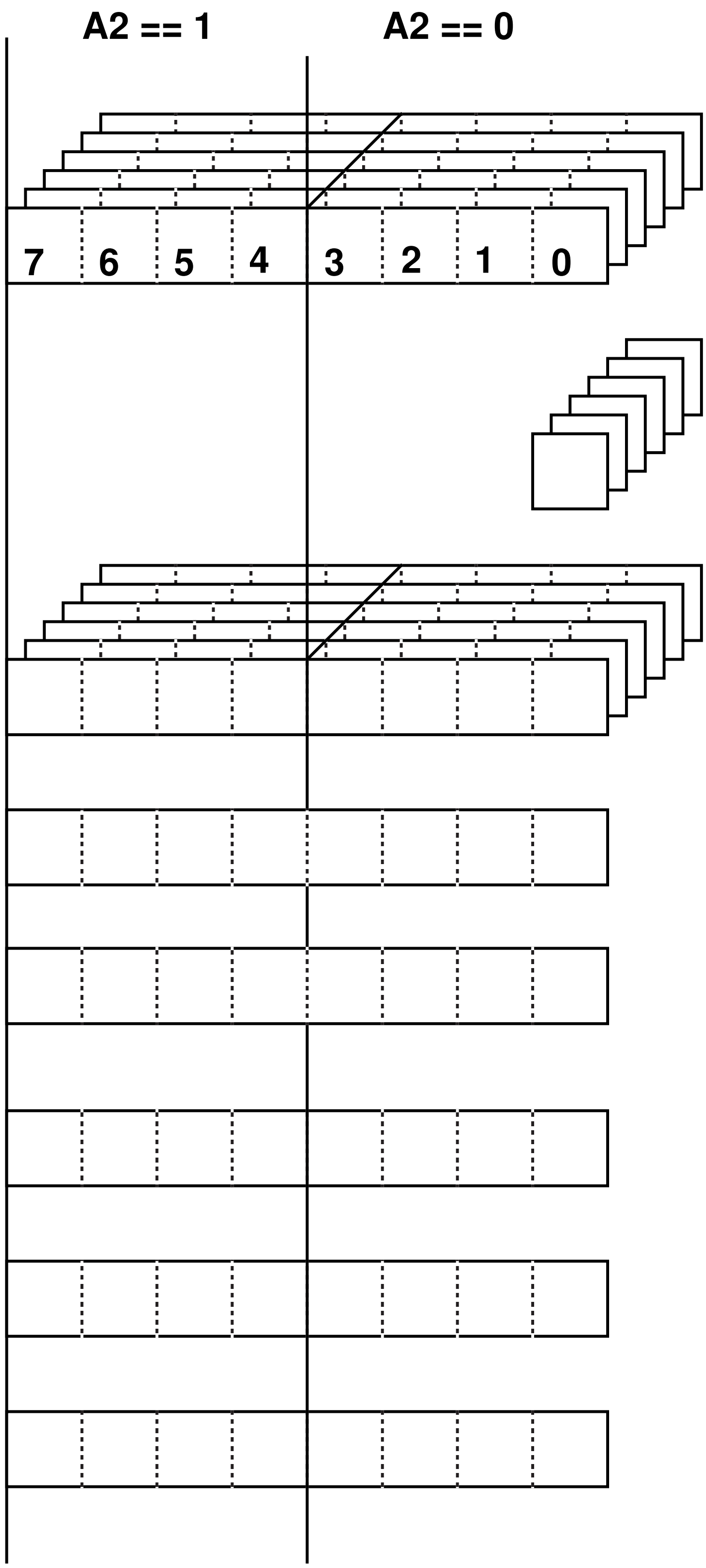
Reset-Rec-CRC (0 Bytes, adr: (char *)0x60):

Schreibt die CPU etwas in diese Adresse, löscht der Link Interface den CRC des Empfangskanals (z.B. bereit zum Empfang einer neuen Nachricht).

Leseregister-Modell:

A1 == A0 == 0

AAA 6 5 4	Hex	Name
0 0 0 64-bit	0x00 DAT	Rec. Data
0 0 1 8-bit	0x10 CMD	Rec CMD
0 1 0 64-bit	0x20 ANY	Rec any Thing
0 1 1 64-bit	0x30 HED	rec fifo HEAD
1 0 0 64-bit	0x40 HD1	rec fifo HEAD
1 0 1 64-bit	0x50 RCR	Rec. CRC
1 1 0 64-bit	0x60 SCR	Send CRC
1 1 1 64-bit	0x70 STS	global status



Bei jedem CPU-Lesezugriff liefert der Link Interface bis zu 8 Byte-Informationen. Die CPU kann alle oder auch nur einige Bytes nehmen. Manche Informationen können nur einmal gelesen werden, (z.B. empfangene Daten und Befehle). Nimmt die CPU nicht alle Bytes, geht der Rest verloren.

Rec-Daten (8 bytes, adr: (double *)0x00):

Diese Adresse ist der Kopf des Receive-Fifos. Die Daten, die vom Link Interface empfangen werden, werden zu 8 Byte-Wörtern gruppiert und in den Receive-Fifo geschrieben. Die CPU kann die Daten in dieser Adresse lesen. Liest die CPU diese Adresse, wenn der Fifo leer ist, bekommt sie alle Bits gesetzt (-1). Um das zu verhindern (ungültige Daten zu bekommen), sollte die CPU erst den Zustand des Fifos lesen, um herauszufinden, wieviele Daten bereit sind, empfangen zu werden. Man beachte, daß derselbe Kopf des Fifos Daten oder Befehle eines Verbindungsnetzes halten kann. Der Link-Interface kontrolliert, was auf dem Kopf des Link-Interfaces ist (Befehl oder Data). Enthält der Kopf des Fifos Daten, nach dem Lesen der Adresse, wird der nächste Eingang beim Receive-Fifo abgeholt. Enthält der Kopf des Fifos einen Befehl, und die CPU liest die Adresse, wird er etwas empfangen (einen Byte-Befehl und etwas anderes), aber der Link-Interface holt den nächsten Eingang beim Fifo nicht ab, bis der Kopf (ein Befehl) der Rec-CMD-Adresse gelesen wurde. Der Link Interface schreibt nur vollständige Wörter (8 Bytes) in den Receive-Fifo. Wird ein unvollständiges Wort vor einem Befehl (z.B. close) empfangen (z.B. wenn einige Bytes im Netz verlorengegangen sind), dann wird das ganze Wort verworfen.

Rec-CMD (1 byte, adr: (char *)0x10):

Diese Adresse entspricht den Rec-Daten (0x00), aber nur ein Byte des Wortes (Byte 0, Addresse 0x08) ist definiert. Enthält der Kopf des Receive-Fifos einen Befehl, und die CPU liest die Adresse, wird der nächste Eingang beim Receive-Fifo abgeholt. Enthält der Kopf des Fifos einen Dateneintrag, und die CPU liest die Adresse, bekommt sie das Datenwort, aber es wird kein weiterer Eingang beim Fifo abgeholt, bis die CPU den Kopf des Fifos mit der Rec-Datenadresse gelesen hat.

Rec-anything (8 bytes, adr: (double *)0x20):

Diese Adresse entspricht der Rec-Daten- und der Rec-CMD-Adresse, aber sie holt, unabhängig vom Informationstyp im Kopf des Fifos, den nächsten Eintrag beim Fifo ab.

Receive FIFO Head (8 bytes, adr: (double *)0x30 and 40):

Diese Adresse entspricht der Rec-Daten- und der Rec-CMD-Adresse, aber sie holt nie den nächsten Eintrag beim Fifo ab.

Receive CRC (8 bytes, adr: (double *)0x50):

Mit dieser Adresse kann die CPU die CRC-Kontrollsumme der angekommenen Daten lesen. Dies ist die Kontrollsumme aller Datenwörter (64 Bits), die die CPU, seit dem letzten Löschbefehl an den Rec-CRC (Schreibe Adresse 0x30), vom Receive-Fifo gelesen hat.

Send CRC (8 bytes, adr: (double *)0x60):

Mit dieser Adresse kann die CPU die CRC-Kontrollsumme der gesendeten Daten lesen. Dies ist die Kontrollsumme aller Datenwörter (64 Bits), die die CPU, seit dem letzten Löschbefehl an den Send-CRC (Schreibe Adresse 0x28), in die Send-Daten-Adresse geschrieben hat. Zu Beginn einer Nachricht löscht die CPU den Send-CRC, sendet alle Daten der Nachricht, sendet einen CRC-Befehl, liest den Send-CRC und sendet die Lesesumme als Daten. Zum Schluß sendet sie einen Schließbefehl. Auf der Empfängerseite empfängt die CPU alle Daten bis der CRC-Befehl ankommt, liest den Rec-CRC, liest den CRC-Befehl, liest das nächste Datenwort des Receive-Fifos und vergleicht es mit dem erwarteten CRC. Dann löscht sie den Rec-CRC und erhält den Schließbefehl. Auch die CPU kann alle Daten erhalten bis der Schließbefehl ankommt. Dann sollte der Rec-CRC 0 sein. Der Schließbefehl sollte nicht gelesen werden bevor der Rec-CRC gelesen und gelöscht wurde, sonst wird das nächste Wort auf dem Fifo (von der nächsten Nachricht) im CRC berechnet (computed).

Global Status (8 bytes, adr (char *)0x70 .. (char *)0x77; oder (double *) 0x70;)

Mit dieser Adresse kann die CPU den internen Zustand der Link Interface lesen:

```
Byte 0 (0x70): Belegte Einträge im Send Fifo
Byte 1 (0x71): Belegte EInträge im Receive Fifo
Byte 2 (0x72): Send Status
         Bit 0 >= 8  Einträge im FIFO
         Bit 1 >= 16 Einträge im FIFO
         Bit 2 >= 20 Einträge im FIFO
         Bit 3 >= 24 Einträge im FIFO
         Bit 4 = FIFO voll
         Bit 5 = FIFO leer
         Bit 6 = Stop aktiv
Byte 3 (0x73) : Receive Status
         Bit 0 >= 8  Einträge im FIFO
         Bit 1 >= 16 Einträge im FIFO
         Bit 2 >= 20 Einträge im FIFO
         Bit 3 >= 24 Einträge im FIFO
         Bit 4 = FIFO voll
         Bit 5 = FIFO leer
         Bit 6 = Stop aktiv
Byte 4 (0x74): Kopf des Receive-Fifos:
          000 = Nichts (nichts zum Lesen)
          101 = Kommando angekommen
          110 = Daten angekommen
                Bit 2 ist ein or von bits 0 und 1.
Byte 5 (0x75) : CRC Status:
          00 = Beide CRC != 0
          01 = CRC der empfangenen Daten == 0
          10 = CRC der gesendeten Daten == 0
          11 = Beide CRC == 0
Byte 6 (0x76): Immer 0
Byte 7 (0x77): Immer 0
```