(19) **Bundesrepublik Deutschland**
Deutsches Patent- und Markenamt

(10) **DE 691 33 518 T2** 2006.11.23

(12) # Übersetzung der europäischen Patentschrift

(97) **EP 0 971 501 B1**
(21) Deutsches Aktenzeichen: **691 33 518.4**
(96) Europäisches Aktenzeichen: **99 203 396.9**
(96) Europäischer Anmeldetag: **22.05.1991**
(97) Erstveröffentlichung durch das EPA: **12.01.2000**
(97) Veröffentlichungstag
der Patenterteilung beim EPA: **15.03.2006**
(47) Veröffentlichungstag im Patentblatt: **23.11.2006**

(51) Int Cl.[8]: ***G06F 13/42*** (2006.01)

(30) Unionspriorität:
9011700 **25.05.1990** **GB**

(73) Patentinhaber:
STMicroelectronics Ltd., Almondsbury, Bristol, GB

(74) Vertreter:
Samson & Partner, Patentanwälte, 80538 München

(84) Benannte Vertragsstaaten:
DE, FR, GB, IT

(72) Erfinder:
Simpson,Robert J., Redland Bristol BS6 6QD, GB

(54) Bezeichnung: **Kommunikationsschnittstelle**

Beschreibung

[0001] Die Erfindung betrifft Kommunikationsschnittstellen und ist insbesondere auf Kommunikationsschnittstellen zur Verwendung mit einer Computervorrichtung und der Übertragung von Nachrichten zwischen Computern oder Computern und anderen damit verbundenen Vorrichtungen anwendbar.

[0002] Computervorrichtungen und andere integrierte Schaltkreisvorrichtungen können es erfordern, Nachrichten zu übertragen oder Nachrichten von anderen Vorrichtungen zu empfangen. In einigen Fällen kann die Nachrichtenübertragung zwischen zwei miteinander verbundenen Vorrichtungen oder zwischen einer großen Anzahl an Vorrichtungen erfolgen, die ein Netzwerk bilden. In einem solchen Netzwerk kann eine oder mehrere der Vorrichtungen in der Form einer Computervorrichtung sein und andere Vorrichtungen können eine Anzahl an peripherem Gerät umfassen. Solche Netzwerke können Routing-Schalter umfassen, um einen großen Umfang an Verbindungen in dem Netzwerk zuzulassen. Die Übertragung von Daten zwischen derart verbundenen Vorrichtungen kann erfolgen, indem parallele Datenbusse oder serielle Kommunikationsleitungen verwendet werden. Schwierigkeiten treten beim Steuern von Bussen in solchen Netzwerkverbindungen auf und ferner ist die Arbeitsgeschwindigkeit insbesondere eingeschränkt, wenn eine Mehrzahl an Vorrichtungen in dem Netzwerk verbunden ist.

[0003] US-A-3,889,236 beschreibt, wie ein seriell arbeitender Schnittstellenadapter an verschiedenen Stellen in Computersystemen oder Datenübertragungssystemen verwendet werden kann, um die Anzahl von Übertragungsleitungen zu verringern. Der Adapter verwendet einen speziellen Modulator und Demodulator und zugeordnete Kodiertechnik, wobei der Modulator eine Einrichtung umfasst, um Bit-Paare zu erzeugen, die einen geraden und ungeraden Zustand haben, und bei dem die Paare von Signalelementen von abwechselnder Parität sind. Der Demodulator ist ausgelegt, um 4-Bit-Paritäts-Kodesignale zurückzugewinnen und zu detektieren, um Start- und Stoppsignale einer Blockinformation wiederherzustellen.

[0004] US-A-4,369,516 behandelt ein Datenübertragungssystem, bei dem Datensignale zwischen einem Datensender und einer Mehrzahl von Datenempfängern in selbsttaktenden Bit-Strömen, die auf Echtdaten- und Komplementdatensignalleitungen übertragen werden, und einem Bit-Strom ohne Rückkehr nach Null (NRZ; engl.: non-return-to-zero) auf einer Signalleitung zur Rückgabe von Daten (engl.: return date signal line) bidirektional übertragen werden.

[0005] Es ist eine Aufgabe der vorliegenden Erfindung, eine einfache Hochgeschwindigkeitsschnittstelle zwischen solchen kommunizierenden Vorrichtungen bereitzustellen. Es ist eine Aufgabe, einen Hochgeschwindigkeitsbetrieb ohne Verlust von Daten zu ermöglichen und zu ermöglichen, Nachrichten variabler Länge in Form mehrerer Pakete zu übertragen. Dies ist vorteilhaft, um es einer beliebigen Schnittstelle zu ermöglichen, Pakete unterschiedlicher Nachrichten in Folge vor Abschluss einer Übertragung aller Pakete einer Nachricht zu handhaben.

[0006] Die vorliegende Erfindung stellt eine Kommunikationsschnittstelle für die Verwendung in einem Kommunikationssystem bereit, das einen Computer mit wenigstens einer anderen Vorrichtung verbindet, wobei die Schnittstelle eine Ausgabeschaltkreisanordnung zur Ausgabe von Nachrichten und eine Eingabeschaltkreisanordnung zur Eingabe von Nachrichten umfasst, wobei die Ausgabeschaltkreisanordnung eine Steuerschaltkreisanordnung und eine Kodierschaltkreisanordnung aufweist, um zwei parallele Ausgaben bereitzustellen, eine in Form eines Datensignals mit einem seriellen Bitmuster, das wenigstens einen Teil der Ausgabenachricht bildet, und die andere in Form eines Strobe-Signals, das, wenn Daten in das Datensignal ausgegeben werden, Signalübergänge nur an Bitgrenzen aufweist, wo es keinen Übergang auf dem parallelen Datensignal gibt, und die Eingangsschaltkreisanordnung eine Dekodierschaltkreisanordnung mit zwei Eingängen aufweist, einen ersten Eingang zum Empfang eines Datensignals und einen zweiten Eingang zum Empfang eines Strobe-Signals, wobei die Dekodierschaltkreisanordnung ausgelegt ist, sowohl auf beide der Daten- und Strobe-Signale anzusprechen, um Daten, die in dem Datensignal kodiert sind, zu dekodieren, dadurch gekennzeichnet, dass die Ausgabeschaltkreisanordnung ferner eine Synchronisations-Token erzeugende Schaltkreisanordnung umfasst, eine Ausgabe eines Synchronisations-Token zu bewirken, das in das Datensignal auszugeben ist, und wobei die Ausgabeschaltkreisanordnung ausgelegt ist, zu bewirken, dass die beiden parallelen Ausgaben, simultane Übergänge haben, wenn ein Synchronisations-Token übertragen wird, und die Eingangsschaltkreisanordnung ferner eine Synchronisations-Schaltkreisanordnung aufweist, die auf das Synchronisations-Token anspricht, um eine Synchronisation von Signalen an den beiden Eingängen zu bewirken.

[0007] Vorzugsweise ist die Steuerschaltkreisanordnung ausgelegt, Daten in dem Datensignal in Token mit vorbestimmter Länge auszugeben.

[0008] Vorzugsweise ist die Steuerschaltkreisanordnung betriebsfähig, Token von mehr als einer vorbestimmten Bit-Länge auszugeben.

[0009] Vorzugsweise weist die Steuerschaltkreisanordnung einen Paritätsbit-Generator auf, um ein Paritätsbit zur Implikation in jedes Token erzeugen.

[0010] Vorzugsweise weist die Steuerschaltkreisanordnung einen Flag-Bit-Generator auf, um ein Flag-Bit zur Implikation in jedes Token zu erzeugen, um jedes Token als Daten-Token oder Steuer-Token zu identifizieren.

[0011] Vorzugsweise stellt das Flag-Bit eine Angabe einer Token-Länge in dem genannten vorherigen Token bis zu dem Flag-Bit des genannten einen Token und schließlich desselben bereit.

[0012] Vorzugsweise ist die Steuerschaltkreisanordnung ausgelegt, Steuer-Token und Daten-Token bereitzustellen, die jeweils eine entsprechende vorbestimmte Bit-Länge haben, wobei jedes Daten-Token eine größere Bit-Länge als ein Steuer-Token hat.

[0013] Vorzugsweise weist die Eingabeschaltkreisanordnung eine Verzögerungsschaltkreisanordnung auf, die zwischen jedem der zwei Eingänge und dem Dekodierer angeschlossen ist, und eine Einrichtung auf, um vor dem Dekodieren die Verzögerung von einem oder von beiden der Eingänge zu variieren.

[0014] Vorzugsweise weist die Ausgabeschaltkreisanordnung eine Fluss-Steuereinrichtung zum Erzeugen eines Fluss-Steuer-Token zur Ausgabe an eine angeschlossene Kommunikationsschnittstelle auf, und weist die Eingangsschaltkreisanordnung eine Einrichtung auf, die auf eine Eingabe eines Fluss-Steuer-Token anspricht, um den Betrieb der Ausgabeschaltkreisanordnung beim Ausgeben weiterer Datensignale zu steuern.

[0015] Vorzugsweise weist die Eingabeschaltkreisanordnung eine Speichereinrichtung auf, um eine Mehrzahl an Datensignalen zu speichern, und spricht die Fluss-Steuer-Einrichtung auf die Inhalte der Registereinrichtung an.

[0016] Vorzugsweise ist die Kodierschaltkreisanordnung ausgelegt, für jedes kodiertes Token einen Header bereitzustellen, wobei der Header sowohl das Paritätsbit als auch das Flag-Bit umfasst.

[0017] Vorzugsweise ist der Paritätsbit-Generator mit dem Kodierer verbunden, um auf die Anzahl von Bits ansprechen, die in einem ersten Token nach einem Flag-Bit in dem ersten Token kodiert sind, und um ein Paritätsbit an einer ersten Stelle eines zweiten Token zuzuführen, das die Summe der Anzahl an Bits von den ersten Token zusammen mit den Paritäts- und Flag-Bits des zweiten Token angibt.

[0018] Die Erfindung stellt auch ein Verfahren bereit, um eine bidirektionale Kommunikation zwischen wenigstens zwei miteinander verbundenen Vorrichtungen zu bewirken, wobei mindestens eine der Vorrichtungen einen Computer umfasst, wobei das Verfahren umfasst, parallele Datensignal- und Strobe-Signal-Kommunikationswege zwischen zwei Verbindungsschnittstellen aufzubauen, die jeweils mit einer entsprechenden der beiden Vorrichtungen verbunden sind, ein serielles Bitmuster zu kodieren, um ein Datensignal als wenigstens ein Teil einer Ausgabenachricht zu bilden, und das Datensignal auf den Datensignalweg und ausgehend von der einen Verbindungsschnittstelle ein Strobe-Signal auf den Strobe-Signalweg auszugeben, das, wenn Daten in das Datensignal ausgegeben werden, Signalübergänge nur an Bitgrenzen aufweist, wo es keinen Übergang auf dem parallelen Datensignal gibt, das Datensignal und das Strobe-Signal an der anderen der Verbindungsschnittstellen parallel einzugeben, und auf beide der Daten- und Strobe-Signale anzusprechen, um Daten, die in dem Datensignal kodiert sind, zu dekodieren, und, gekennzeichnet durch, ferner ein Synchronisations-Token bereit zu stellen, das in das Datensignal auszugeben ist, und Übergänge gleichzeitig zu bewirken, die in die Daten- und Strobe-Signalwege auszugeben sind, wenn das Synchronisations-Token auf dem Datenweg übertragen wird, und die gleichzeitigen Übergänge zu verwenden, um eine Synchronisation der Signale auf den Daten- und Strobe-Wegen zu bewirken, wenn sie eine Verbindungsschnittstelle eingeben werden.

[0019] Vorzugsweise umfasst das Verfahren ferner, ein Daten-Token bereit zu stellen, bei dem eine Datenausgabe auf dem Datensignalweg vorliegt und das Strobe-Signal die Signalübergänge nur an den Bitgrenzen aufweist, wo es keinen Übergang auf dem parallelen Datensignal gibt.

[0020] Vorzugsweise schließt das Verfahren ferner ein, vier unidirektionale Kommunikationswege zwischen jedem Paar von Verbindungsschnittstellen aufzubauen, wobei die vier Wege ein erstes paralleles Paar an Daten- und Signalwegen in einer Richtung und eine zweites paralleles Paar an Daten- und Signalwegen in der

entgegengesetzten Richtung aufweisen.

[0021] Vorzugsweise werden Daten von einer Verbindungsschnittstelle in Token einer vorbestimmten Bitlänge ausgegeben.

[0022] Vorzugsweise werden Daten durch eine Verbindungsschnittstelle in Token von mehr als einer vorbestimmten Bit-Länge ausgegeben.

[0023] Vorzugsweise umfasst das Verfahren ein Flag-Bit zu Implikation in jedes Token zu erzeugen, um das Token als Daten-Token oder Steuer-Token zu identifizieren.

[0024] Vorzugsweise sind das Paritätsbit und das Flag-Bit an ersten bzw. zweiten Bit-Positionen in jedem Token angeordnet.

[0025] Vorzugsweise umfasst das Verfahren zwei sukzessive Steuer-Token, um ein zusammengesetztes Token zu bilden, wobei das erste der Steuer-Token ein Bitmuster aufweist, das angibt, dass ein weiteres Token erforderlich ist, um die durch das zusammengesetzte Token angegebene Steuerung festzulegen.

[0026] Vorzugsweise werden Nachrichten zwischen verbundenen Verbindungsschnittstellen in Paketen variabler Länge übertragen, wobei jedes Paket eine mehrfache Anzahl an Token umfasst, wobei jedes Token eine vorbestimmte Bit-Länge hat und am Ende jedes Pakets ein Ende-des-Pakets-Token bereitstellt.

[0027] Vorzugsweise wird ein Ende-der-Nachricht-Token an dem Ende eines letzten Pakets in einer Nachricht eingebunden.

[0028] Vorzugsweise umfasst das Verfahren, ein Fluss-Steuer-Token zur Ausgabe durch eine Verbindungsschnittstelle zu bilden, Daten ausgehend von einer ersten Schnittsellte an eine zweite Schnittstelle auszugeben, ein Fluss-Steuer-Token ausgehend von der zweiten Schnittstelle an die erste Schnittstelle auszugeben, um der ersten Schnittstelle anzuzeigen, dass weitere Daten-Token an die zweite Schnittstelle ausgegeben werden können.

[0029] Vorzugsweise umfasst das Verfahren, eine Zählung aufrechtzuerhalten, die durch Ausgabe von Token durch eine ausgebende Verbindungsschnittstelle angepasst wird, eine weitere Ausgabe von Daten-Token ausgehend von der genannten Verbindungsschnittstelle zu sperren, wenn die Zählung eine vorbestimmte Zahl erreicht, und die Zählung in Antwort auf eine Eingabe eines Fluss-Steuer-Token ausgehend von einer Verbindungsschnittstelle anzupassen, um die Ausgabe von weiteren Daten-Token zu ermöglichen.

[0030] Vorzugsweise ist jede Verbindungsschnittstelle ausgelegt, Daten-Token einer vorbestimmten Bit-Länge auszugeben, die größer als eine zweite vorbestimmte Bit-Länge für Steuer-Token ist.

[0031] Vorzugsweise ist jede Verbindungsschnittstelle ausgelegt, Steuer-Token von zwei Typen auszugeben, einen ersten Typ zum Steuern des Betriebs einer angeschlossenen Verbindungsschnittstelle und einen zweiten Typ zur Verwendung durch die Vorrichtung, die mit der zweiten Verbindungsschnittstelle verbunden ist.

[0032] Vorzugsweise ist jede Verbindungsschnittstelle ausgelegt, in ihrer Eingabeschaltkreisanordnung, Daten-Token und Steuer-Token des zweiten Typs zu speichern.

[0033] Vorzugsweise umfasst das Verfahren, Daten-Token und Steuer-Token des zweiten Typs ausgehend von einem Speicher in der Verbindungsschnittstelle zu der Vorrichtung, die mit der Verbindungsschnittstelle verbunden ist, mittels Verwendung eines synchronisierten Hand-Shakes zu übertragen.

[0034] Vorzugsweise ist eine Verbindungsschnittstelle ausgelegt, ein Paket auszugeben, das ein oder mehrere Daten-Token einer Adresse eines Kommunikationskanals aufweist, der in der Vorrichtung zu verwenden ist, die mit einer Verbindungsschnittstelle verbunden ist, die ausgelegt ist, das Paket zu empfangen.

[0035] Das Verfahren kann umfassen, eine bidirektionale Kommunikation zwischen einer Mehrzahl von Vorrichtungen zu bewirken, die in einem Netzwerk enthalten sind, das eine Mehrzahl an Mikrocomputern und wenigstens einen Routing-Schalter aufweist.

[0036] Eine Ausführungsform der Erfindung wird nun als Beispiel und unter Bezugnahme auf die beigefügten

Zeichnungen beschrieben, in denen:

[0037] Fig. 1 ein Blockdiagramm eines Netzwerks ist, das eine Mehrzahl an Verbindungsschnittstellen gemäß der vorliegenden Erfindung aufweist,

[0038] Fig. 2 eine schematische Ansicht einer Verbindungsschnittstelle gemäß der Erfindung ist,

[0039] Fig. 3 detaillierter den Verbindungsausgang der in Fig. 2 gezeigten Anordnung zeigt,

[0040] Fig. 4 detaillierter den Verbindungseingang der in Fig. 2 gezeigten Anordnung zeigt,

[0041] Fig. 5 das Bitmuster auf Daten- und Strobe-Leitungen für zwei aufeinanderfolgende Token zeigt, die durch Verwendung der in Fig. 2 gezeigten Vorrichtung erhalten werden,

[0042] Fig. 6 eine Ausgabe auf den Daten- und Strobe-Leitungen während der letzen vier Bit eines Daten-Synchronisations-Token zeigt,

[0043] Fig. 7 eine Eingabe auf die Daten- und Strobe-Leitungen nach einer Eingabe, wie in Fig. 6 gezeigt, zeigt, und

[0044] Fig. 8 eine alternative Eingabe auf die Daten- und Strobe-Leitungen nach einer Eingabe, wie in Fig. 6 gezeigt, zeigt.

[0045] Das in Fig. 1 gezeigte Netzwerk umfasst eine Mehrzahl an Mikrocomputer **11**, **12**, **13** und **14**, die jeweils einen einzelnen Mikrocomputer mit integriertem Schaltkreis umfassen können, wie zum Beispiel der, der in unserem US-Patent 4680698 gezeigt ist. Das Netzwerk kann auch weitere Ausrüstung aufweisen, wie zum Beispiel einen Mikroprozessor **15**, periphere Einheiten **16** und **17** und eine Festplattensteuervorrichtung **18**, die eine Festplatte **19** steuert. Das Netzwerk weist auch einen Routing-Schalter **20** auf, der wie in unserer europäischen Patentanmeldung Publikations-Nr. 04059990 beschrieben sein kann. Jeder der Mikrocomputer **11**–**14** weist eine Mehrzahl an Verbindungseinheiten **21** auf, die jeweils eine Kommunikationsschnittstelle bereitstellen, die mit vier undirektionalen Signaldrähten **23** verbunden sind, was eine bidirektionale Kommunikation mit einer angeschlossenen Verbindungseinheit **21** an einer anderen Vorrichtung in dem Netzwerk bildet. Bei dem gezeigten Beispiel ist der Mikroprozessor **15** mit einem Bus **22** verbunden, der mit einer Verbindungseinheit **21** verbunden ist. Einige der Vorrichtungen, die zum Beispiel der Mikrocomputer **12** und die Festplattensteuervorrichtung **18**, sind unmittelbar durch vier Signaldrähte **23** angeschlossen, wohingegen andere Vorrichtungen über den Routing-Schalter **20** angeschlossen sind. Jeder der Mikrocomputer **11**, **12**, **13**, **14**, der Mikroprozessor **15** und die peripheren Einheiten **16** und **17** und die Disk-Steuereinrichtung **18** arbeiten als Hauptrechnervorrichtung im Verhältnis zu der Verbindungseinheit **21**, die eine Eingabe zu und eine Ausgabe ausgehend von der Hauptrechnervorrichtung ermöglicht. Jede der Verbindungseinheiten **21** ist vergleichbar und deren Aufbau und Betrieb ist unten detaillierter beschrieben. Jede ist ausgelegt, um eine bidirektionale Kommunikation zwischen Paaren von Vorrichtungen in dem Netzwerk zu ermöglichen. Die Kommunikation ist derart, um Nachrichten variabler Länge zu übertragen, die in eine Folge von Paketen unterteilt werden können. Jede Gruppe von Signaldrähten **23** weist ein erstes paralleles Paar an Drähten **25** und **26** auf, die einen Datensignalweg bzw. einen parallelen Ausblendimpulssignalweg in einer Richtung zwischen einem Paar von verbundenen Verbindungseinheiten **21** bilden. Das zweite Paar an Drähten bildet einen Datensignalweg und einen parallelen Ausblendimpulssignalweg in der entgegengesetzten Richtung zwischen dem gleichen Paar von Verbindungseinheiten **21**. Nachrichten werden zwischen verbundenen Verbindungseinheiten **21** durch serielle Bit-Strings übertragen, die auf den Datensignalwegen **25** kodiert sind, und parallele Signale werden auf dem Ausblendimpulssignalweg **26** beim Dekodieren der Nachrichten verwendet, die auf dem Datensignalweg **25** empfangen werden. Die Bit-Strings werden in Token übertragen, wobei jedes Token von einer vorbestimmten Bit-Länge ist. Bei diesem speziellen Beispiel werden zwei unterschiedliche Token-Längen verwendet. Die als erste bezeichneten Daten-Token sind jeweils 10 Bit lang und der zweite Typ sind Steuer-Token, jeweils 4 Bit lang. Beispiele von jedem sind in Fig. 5 gezeichnet, wo ein Token N ein Beispiel eines Daten-Token ist und das nächste nachfolgende Token N + 1 ein Beispiel eines Steuer-Token ist. Für beide Typen von Token ist die erste Bit-Stelle, die an der linken Seite des Token in Fig. 5 gezeigt ist, ein Paritätsbit für das Token. Die zweite Bit-Stelle ist ein Flag, um anzugeben, ob das Token ein Daten-Token oder ein Steuer-Token ist. In dem Fall eines Daten-Token können die nächsten 8 Bit-Stellen jegliche in der Nachricht benötigten Daten enthalten. In dem Fall von Steuer-Token sind die ersten 2 Bit die gleichen, wie bereits für das Daten-Token beschrieben, und die letzten 2 Bit enthalten eine Steuer-Angabe. Das Strobe-Signal, das parallel zu dem Datensignal übertragen wird, ist ebenfalls in Fig. 5 gezeigt, und dieses ist so ausgelegt, dass, wenn Token auf der Datenleitung

25 übertragen werden, die einen Teil einer Nachricht bilden, die Strobe-Leitung **26** Signalübergänge nur an Bit-Grenzen aufweist, wo es keinen Übergang zu dem parallelen Datensignal gibt. Auch wenn beide Typen von Token jeweils von einer vorbestimmten Bit-Länge sind, kann jede gewünschte Anzahl von Token in einer Folge übertragen werden, um ein einzelnes Paket zu bilden. Das Ende eines Pakets kann durch eine Übertragung eines Steuer-Token markiert sein, dass das Ende des Pakets angibt. Es kann in einigen Fällen für eine Verbindung oder einen Weiterleitungsschalter wünschenswert sein, Pakete einer anderen Nachricht (die möglicherweise zwischen einem anderen Paar Vorrichtungen übertragen wird) zu bearbeiten, bevor eine Übertragung einer ersten Nachricht abgeschlossen wird. Durch Auslegung für die in Paketen zu übertragende Nachricht ist es möglich, eine Übertragung einer Nachricht an dem Ende eines Pakets anzuhalten und die Übertragung dieser Nachricht zu einem späteren Zeitpunkt durch ein oder mehrere nachfolgende Pakete wieder aufzunehmen, die mit einem Ende eines Nachrichten-Token enden, wenn die Nachricht vollständig ist. Dies erlaubt es, eine Mehrzahl von Nachrichten über eine einzelne Verbindung zu multiplexen.

[0046] Das für die Ausgabe von Token auf dem Datensignalweg **25** gemäß diesem Beispiel verwendete Protokoll ist wie folgt:

Funktion	Abkürzungen	Bitmuster
Daten-Byte		POXXXXXXXX
Fluss-Steuer-Token	FCT	P100
Ende des Pakets	EOP	P101
Ende der Nachricht	EOM	P110
Codeumschaltung (engl.: escape)	ESC	P111
Datensynchronisation	DAT	ESC PO11
Null	NULL	ESC P100
Reserve		ESC P1xx

[0047] Die obige Tabelle gibt die Bitmuster für ein Daten-Token zum Übertragen eines Daten-Byte und nachfolgender vier Steuer-Token in der Form eines Fluss-Steuer-Token, eines Ende-des-Pakets-Token, eines Ende-der-Nachricht-Token und eines Codeumschaltungs-Token. P gibt ein Paritätsbit in jedem Token an. Das Daten-Token weist die zweite Bit-Stelle mit der Flag auf 1 gesetzt auf, wohingegen die Steuer-Token das zweite Bit-Flag auf 1 gesetzt aufweisen. Der Zweck des Codeumschaltungs-Token besteht darin, ein zusammengesetztes Steuer-Token zu bilden, das aus zwei aufeinanderfolgenden Token mit vier Bit besteht. Das Codeumschaltungs-Token ist durch sein Flag klar als Steuer-Token markiert und die dritten und vierten Bit-Stellen des Codeumschaltungs-Token geben an, dass das Token, welches folgt, ein Token mit vier Bit ist, das für Steuerzwecke verwendet wird, und keine Daten. Das nachfolgende Token, das verwendet wird, um ein zusammengesetztes Token zusammen mit den Codeumschaltungs-Token zu bilden, kann entweder ein Datensynchronisations-Token oder ein Null-Token oder ein Reserve-Token sein. In der obigen Tabelle bilden die Steuer-Token zwei unterschiedliche Typen. Abschluss-Steuer-Token und zusammengesetzte Token, die durch Verwendung des Codeumschaltungs-Token gebildet werden, bilden einen ersten Typ. Diese Steuer-Token des ersten Typs werden nur durch die Verbindungsschnittstellen selbst für Steuerzwecke verwendet. Das Abschluss-Steuer-Token wird verwendet, um die Rate einer Ausgabe von Token durch eine Verbindung zu steuern, um zu gewährleisten, dass ein Speicher in einer empfangenen Verbindung nicht überfüllt wird. Die Daten-Synchronisations-Token werden verwendet, um die Synchronisation der Daten und Strobe-Signale, wenn von einer Verbindung eingegeben, anzupassen. Null-Token werden normalerweise übertragen, wenn keine andere Token gesendet werden. Steuer-Token des zweiten Typs bestehen aus dem Ende-des-Pakets-Token und im Ende-der-Nachricht-Token und diese werden zusammen mit den Daten-Token von der Hauptrechnervorrichtung benötigt, die mit der Verbindungsschnittstelle verbunden ist, und diese werden häufig in einem Speicher in der Verbindung gespeichert, bis sie mit einem synchronisierten Hand-Shake-System zu dem Hauptrechner übertragen werden. Auf vergleichbare Weise werden Steuer-Token des ersten Typs durch die Verbindungsschnittstelle selbst erzeugt, wohingegen Daten-Token und Steuer-Token des zweiten Typs von der Hauptrechnervorrichtung erzeugt und über ein synchronisiertes Hand-Shake-System zu der Verbindung übertragen werden.

[0048] Wenn eines der Token (außer dem DAT-Token), die in der obigen Tabelle aufgeführt sind, auf den Datensignalweg **25** ausgegeben werden, gibt der Strobe-Signalweg **26** ein Signal aus, das aus Signalübergängen an jeder Bit-Grenze in dem Datensignal besteht, wo es keine Änderung im Signalpegel in dem Datensignal gibt. Wenn ein DAT-Token ausgegeben wird, folgt das Strobe-Signal der normalen Prozedur für Bit-Grenzen

nach Bit-Positionen 1, 2, 3, 4 und 5 des DAT-Token, aber an den Bit-Grenzen nach den Bit-Positionen 6 und 7 folgt es den Inversen des üblichen dahingehend, dass es gleichzeitige Übergänge **140a**, **140b** auf beiden der Daten- und Strobe-Wege nach dem 6. Bit und keinen Übergang in einem Weg nach dem 7. Bit bewirkt. Diesen einzelnen identifizierbaren Rand auf beiden Signalwegen nach dem Bit 6 bereit, der zur Synchronisation verwendet wird. Dies ist in Fig. 6 gezeigt, die die letzten vier Bit-Positionen eines verwundenen Token veranschaulicht, das ein DAT-Token bildet.

[0049] Die Anordnung der Verbindungsschnittstellen wird nun unter Bezugnahme auf die Fig. 2, Fig. 3 und Fig. 4 beschrieben.

[0050] Jede Verbindungseinheit **21** weist eine Ausgabe **30**, eine Eingabe **31**, eine Fluss-Steuer-Einheit **32** und eine Steuerschaltkreisanordnung **33** auf. Die Ausgabeeinheit **30** ist detaillierter in Fig. 3 gezeigt. Eine Hauptrechnerschnittstelle **34** ist vorgesehen, um die Ausgabeeinheit mit der Hauptrechnervorrichtung zu verbinden, wie zum Beispiel der Mikrocomputer **11** in Fig. 1. Die Schnittstelle **34** ist mit dem Host durch einen Datenbus **35** verbunden, der ausgelegt ist, acht parallele Bit an Daten bereitzustellen. Um eine synchronisierte Hand-Shake-Kommunikation mit dem Hauptrechner zu ermöglichen, weist die Schnittstelle **34** auch einen Eingang **36**, um ein Datengültigkeitssignal von dem Hauptrechner zu empfangen, und einen Ausgang **37** auf, um ein Bestätigungssignal bereitzustellen, wenn sie ein Byte an Daten von dem Hauptrechner erhalten hat. Die Schnittstelle **34** weist einen Taktsignaleingang **38** und einen Rücksetzeingang **39** auf. Die Schnittstelle **34** weist auch einen Eingang **40** ausgehend von der Fluss-Steuer-Einheit **32** auf, um eine Datenausgabe zu unterbinden, wenn ausreichend Token bereits ausgegeben worden sind, ohne dabei ein Fluss-Steuer-Token an dem entsprechenden Eingang empfangen zu haben. Die Schnittstelle **34** stellt auch einen Token-Gesendet-Ausgang **41** zu der Fluss-Steuer-Einheit **32** bereit. Daten werden der Schnittstelle **34** zusammen mit einem Datengültigkeitssignal ausgehend von dem Hauptrechner angeboten. Wenn die Ausgabelogik **30** bereit ist, diese Daten anzunehmen, signalisiert sie unter Verwendung eines Bestätigungssignals auf einer Leitung **37** zu dem Hauptrechner. Wenn der Hauptrechner und die Ausgabeeinheit **30** in unterschiedlichen Taktbetriebsbedingungen arbeiten, ist es erforderlich, das Datengültigkeitssignal **36** und das Bestätigungssignal **37** zu synchronisieren. Von der Schnittstelle **34** empfangene Daten werden in ein paralleles Register **42** geladen, und es ist verständlich, dass die Daten in diesem Register entweder ein Daten-Token des zuvor beschriebenen Typs oder ein Steuer-Token EOP oder EOM sind. Um ein Token auszugeben, muss einer einer Mehrzahl von Token-Anforderungs-Zwischenspeichern **43** gesetzt sein, und um eines der Token, die über die Schnittstelle **34** empfangen wurden, eine Eingabe auf einer Leitung **44** als Eingabe zu den Token-Anforderungs-Zwischenspeichern **43** vorgesehen. Die Zwischenspeicher **43** stellen auf einer Leitung **43a** der Schnittstelle **34** auch eine Ausgabe bereit, um anzugeben, wann ein Daten-Token gesendet wurde. Die Token-Anforderungs-Zwischenspeicher **43** weisen auch drei Eingänge **45**, **46** und **47** auf, um entsprechend ein DAT-, FCT- und NULL-Token zu senden. Die Eingänge **45** und **47** kommen von der Steuerschaltkreisanordnung **33** her, wohingegen der Eingang **46** von der Fluss-Steuer-Einheit **32** kommt. Die Zwischenspeicher **43** stellen der Fluss-Steuer-Einheit **32** auch eine Ausgabe **48** bereit, um anzunehmen, wann ein FCT gesendet worden ist. Um ein Token zu senden, wird der entsprechende Eingang zu den Zwischenspeichern **43** angesteuert, und, wenn das Signal an abfallenden Flanken des Taktsignals gültig ist, welches ebenfalls den Zwischenspeichern zugeführt wird, wird dann der geeignete Zwischenspeicher gesetzt und wird durch einen Eingang **49** zurückgesetzt, wenn das Token gesendet worden ist. Die NULL-Anfrage **47** wird während eines normalen Betriebs hoch beibehalten, so dass ein NULL-Token gesendet wird, wenn kein anderes Token gesendet werden muss. Die Zwischenspeicherschaltkreisanordnung **43** ist durch acht Leitungen (vier Token-Anfragen zu der Prioritätszuordnungseinrichtung **50** und vier Token-Priorisiert zu den Zwischenspeichern **43**) mit einer Token- Prioritätszuordnungseinrichtung **50** verbunden und sie stellt auf einer Leitung **51** eine Token-Sortiereinrichtung **52** auch ein Signal bereit, wann immer ein Zwischenspeicher gesetzt ist. Die Token-Prioritätszuordnungseinrichtung **50** erhält die Ausgaben von den Token-Anfrage-Zwischenspeichern **43** und enthält eine Logikschaltkreisanordnung, um die Anfragen von den Zwischenspeichern **43** in der Reihenfolge DAT, FCT, DATA, NULL zu priorisieren. Dies erlaubt eine Steuerung, wenn mehr als eine Zwischenspeicheranfrage zum gleichen Zeitpunkt gemacht wird. Die Token-Prioritätszuordnungseinrichtung **50** stellt vier separate Ausgaben (Token-Prioritätszuordnungseinrichtung) in Abhängigkeit davon bereit, welcher Zwischenspeicher gesetzt wurde, und diese sind mit einem Steuerkode-ROM **53** verbunden, das ebenfalls als Datenmultiplexer dient. Der ROM **53** ist mit Bitmustern für die Steuercodes ESC, DAT, FCT und NULL programmiert. Diese sind die Steuer-Token, die lediglich innerhalb der Verbindungslogik erzeugt werden, während alle anderen Token über die Schnittstelle **34** ausgehend von dem Hauptrechner eingegeben werden. Wenn der Zwischenspeicher **43** gesetzt worden ist, um anzugeben, dass ein Daten-Token (dies ist ein durch die Schnittstelle **34** zugeführtes Token) zu senden ist, dann werden die Inhalte des Datenregisters **43** zu dem Ausgang des ROM-Schaltkreises **53** durchgelassen, der mit einem Ausgabeschieberegister **54** verbunden ist. Die Ausgabe besteht aus 8 Bit (nur 2 davon werden für die Token EOP und EOM verwendet) und einem Steuer- oder Daten-Flag-Signal auf einer Leitung **55**. Das Register **54** emp-

fängt die Daten parallel und gibt sie in serieller Form mit dem Steuer- oder Daten-Flag aus, das dem 8 Datenbit vorhergeht. Die Ausgabe wird auf einer Leitung **58** dem Paritäts-Generator **59** zugeführt. Der Paritäts-Generator enthält einen zurücksetzbaren Zwischenspeicher mit einem Ausgang, der über ein exklusives OR-Gatter zu dem Eingang zurückgeführt ist. Der Paritäts-Generator **59** weist einen Paritätsrücksetzeingang **60** ausgehend von der Sortiereinrichtung **57** und einem Eingang zur Freigabe einer Ausgabe einer Parität auf einer Leitung **61** ausgehend von der Sortiereinrichtung **57** auf. Dies hat die Wirkung, dass auf jedes Bit in einem Token angesprochen wird, dass durch den Generator **59** ausgehend von dem Register **54** zugeführt wird. Das Paritätssignal wird ausgegeben, nachdem das Steuer- oder Daten-Flag des nächsten Token in den Paritäts-Generator eingegeben wird. Das Paritätsbit stellt dadurch eine Angabe der Anzahl von O's oder I's bereit, die seit dem letzten Steuer- oder Daten-Flag bis zu und einschließlich dem bzw. des nächsten Steuer- oder Daten-Flag ausgegeben worden sind.

[0051] Es ist ersichtlich, dass jedes Paritätsbit eine Überprüfung für die Bit bereitstellt, die einen Steuer- oder Daten-Flag in einem Token bis zu und einschließlich den bzw. des Steuer- oder Daten-Flag des nächsten Token folgen. Bei dem beschriebenen System, das Token variabler Länge verwendet, ist das Steuer- oder Daten-Flag das Mittel, die Bit-Länge des Token anzugeben. Es ist daher wichtig, eine Paritätsüberprüfung vorzunehmen, die einer Überprüfung des Steuer- oder Daten-Flag selbst umfasst, ohne dabei Daten-Bit einzuschließen, die dem Steuer- oder Daten-Flag folgen. Auf diese Weise wird, wenn die Paritätsüberprüfung des Bit-String einschließlich des Steuer- oder Daten-Flag keinen Fehler angibt, das Steuer- oder Daten-Flag als die Bit-Länge des nächsten Token korrekt angebend akzeptiert. Dies führt wiederum zu der richtigen Anzahl an Bit, die beim Durchführen der nächsten Paritätsüberprüfung berücksichtigt werden. Mit anderen Worten, das Bit, das die Token-Länge angibt, muss den Daten-Bit dieses Token überprüft werden, so dass keine Unsicherheit darüber besteht, wie viele Bit beim Durchführen der nächsten Paritätsüberprüfung beurteilt werden sollten. Wenn die Paritätsüberprüfung durchgeführt ist, ist es wichtig, dass das Paritätsbit an einer bekannte Position in jedem Token angeordnet ist. Die bekannte Position muss unabhängig von der Länge des Token konstant bleiben und muss daher innerhalb der kürzesten Bit-Länge eines beliebigen verwendeten Token liegen. Aus diesem Grund ordnet das beschriebene Beispiel das Paritätsbit an der ersten Bit-Position jedes Token an, auch wenn es eine Paritätsüberprüfung des Steuer- oder Daten-Flag, dass sich an der zweiten Bit-Stelle dieses Token befindet, und aller Bit in dem vorhergehenden Token vorsieht, das dem Steuer- oder Daten-Flag des vorhergehenden Token folgt.

[0052] Die Ausgabe des Paritäts-Generators **59** seit dem letzten Steuer- oder Daten-Flag und das Steuer- oder Daten-Flag des nächsten Token einschließend. Der Paritäts-Generator wird durch ein Signal auf einer Leitung **61** eingeschaltet, um eine Ausgabe mit der Paritätsbit-Ausgabe nach dem letzten Daten-Bit bereitzustellen. Dieses Paritätsbit bildet dadurch das erste Bit eines nachfolgenden Token. Die Ausgabe des Paritäts-Generators **59** wird einem Daten/Strobe-Kodierer **62** zugefügt. Dies ist eine Zustandsmaschine, die den folgenden Gleichungen folgt:
On reset Data (0) = 0, Strobe (0) = 0
Strobe (n + 1) = Strobe(n) EXOR NOT (Data(n + 1) EXOR Data(n) EXOR
InvertStrobe)

[0053] Das InvertStrobe-Signal ist normalerweise logisch 0. Es wird nur während des speziellen Bit in dem Datensynchronisations-Token gesetzt.

[0054] Der Kodierer **62** wird durch eine Eingabe auf einer Leitung **63** ausgehend von der Sortiereinrichtung **57** eingeschaltet und er erhält ausgehend von einem DAT-Generator **65** eine invertierte Strobe-Eingabe **64**, wenn ein DAT-Token zu senden ist. Im normalen Betrieb spricht der Kodierer **62** auf den Bit-String an, der ausgehend von dem Paritätsgenerator **59** erhalten wird, und erzeugt eine Strobe-Signalausgabe **26** parallel zu der Datensignalausgabe **25**. Diese Ausgaben werden entsprechend durch Ausgangstreiber **67** und **68** geführt, die durch den Taktgeber der Ausgangsverbindung getaktet sind. Die Ausgangstreiber **67** und **68** sind identisch und gleichzeitig getaktet, um einen Versatz zwischen den Daten- und Strobe-Signalen zu minimieren. Die Wirkung des Kodierers **62** besteht darin, ein Strobe-Signal parallel zu den Datensignalen bereitzustellen, so dass, wenn andere Token als DAT-Token ausgegeben werden, Signalübergänge auf der Strobe-Leitung **26** nur an Bit-Grenzen vorkommen, wenn es auf dem parallelen Datensignal **25** keinen Übergang gibt. Eine typische Daten- und Strobe-Signalkette ist **25** keinen Übergang gibt. Eine typische Daten- und Strobe-Signalkette ist in Fig. 5 veranschaulicht. Die Sortiereinrichtung **57** ist eine Zustandsmaschine, die Strobe-Signale zu dem Ausgabeschieberegister **54**, dem Paritäts-Generator **59**, dem Kodierer **62** und der Token-Sortiereinrichtung **52** zuführt. Wenn durch ein Stabsignal auf einem Eingang **70** ausgehend von der Token-Sortiereinrichtung **52** in Antwort darauf, dass ein Zwischenspeicher **43** gesetzt ist, eingeleitet, wird bewirkt, dass ein Token in das Ausgabeschieberegister **54** durch ein Eingangssignal auf einer Leitung **74** zwischengespeichert wird. Nachdem das

Token in dem Register zwischengespeichert ist, stellt die Sortiereinrichtung auf einer Leitung **72** eine Eingabe bereit, um zu bewirken, dass das Register **54** die Flag- und Daten-Bits seriell entlang einer Leitung **58** durch den Paritäts-Generator **59** schiebt. Die Sortiereinrichtung erhält auf einer Leitung **56** eine Eingabe, die angibt, ob das Token ein mit vierzehn Bit ist, um die Anzahl an Bits zu steuern, die aus dem Register **54** verschoben werden. Wenn die geeignete Anzahl an Bits gesendet worden ist, stellt die Soriereinrichtung **57** auf einer Leitung **94** der Token-Sortiereinrichtung **52** eine Ausgabe bereit, um anzugeben, dass nun ein weiteres Token gesendet werden kann. Die Token-Sortiereinrichtung **52** stellt ein Freigabesignal auf einer Leitung **75** der Token-Prioritätszuordnungseinrichtung **50** nach einem Signal auf der Leitung **74** bereit, so dass die Token-Prioritätszuordnungseinrichtung **50** dann das Token mit der höchsten Priorität berechnet, und gleichzeitig geben die Token-Anfrage-Zwischenspeicher (**43**) der Token-Sortiereinrichtung über eine Leitung (**51**) an, dass es ein zu sendendes Token gibt. Dies bewirkt, dass die Token-Sortiereinrichtung **52** der Token-Prioritätszuordnungseinrichtung auf einer Leitung **75** (Freigabe) signalisiert, seine Ausgaben anzuhalten, damit das Token gesendet werden kann. Die Sortiereinrichtung **52** ist eine Zustandsmaschine, die das Senden jedes Token steuert. Ein Betriebszyklus für die Sortiereinrichtung **52** wird durch eine Eingabe auf der Leitung **51** ausgehend von den Zwischenspeichern **43** eingeleitet. Wenn die Token-Prioritätszuordnungseinrichtung **50** angibt, dass der gesetzte Zwischenspeicher ein ESC-Token fordert, dann wird auf einer Leitung **76** ein Signal der Token-Sortiereinrichtung **52** bereitgestellt, die den ROM **53** auf einer Leitung **77** eine Ausgabe bereitstellt. Die Token-Sortiereinrichtung **52** signalisiert dann der Sortiereinrichtung **57** auf der Leitung **70**, die das Codeumschaltungs-Token sendet. Die Sortiereinrichtung **57** signalisiert dann der Token-Sortiereinrichtung **52** auf der Leitung **74** zurück, dass das Token gesendet worden ist. Die Token-Sortiereinrichtung entfernt dann das Sende-ESC-Token-Signal **77** und signalisiert der Sortiereinrichtung **57**, ein zweites Token mit vier Bit zu senden. Die Sortiereinrichtung **57** sendet dann zweite Token mit vier Bit und signalisiert dann zurück zu der Token-Sortiereinrichtung. Die Token-Sortiereinrichtung **52** setzt dann den Zwischenspeicher durch ein Signal auf der Leitung **49** zurück. Um eine Synchronisation der Daten- und Strobe-Signale an einem Verbindungseingang zu ermöglichen, können DAT-Token gesendet werden, und diese erzeugen spezielle Ausgaben auf den Leitungen **25** und **26** unter Verwendung des DAT-Generators **65**. Die Sortiereinrichtung **57** empfängt, wenn das zu sendende Token ein DAT-Token ist, ausgehend von dem ROM **53** eine Eingabe **79**. Das Signal auf einer Leitung **79** wird von der Sortiereinrichtung **57** verwendet, um DAT-Token von anderen Token zu unterscheiden. Die Sortiereinrichtung **57** sendet für DATs statt einem Bit-Token ein Vier-Bit-Token, wenn das C/D-Flag **56** Null ist, und signalisiert auf der Leitung **73** beim Start des zweiten Token mit vier Bit zu dem DAT-Generator **65**. Die Sortiereinrichtung **57** wartet dann auf eine Antwort von dem DAT-Generator auf einer Leitung **80**, bevor mit dem nächsten Token weiter vorgegangen wird. Der DAT-Generator **65** invertiert dann das Strobe-Signal während des dritten Bit des Token, das dem ESC-Token folgt, als Ergebnis eines invertierten Strobe-Signals auf der Leitung **54**, das zu dem Kodierer **62** geführt wird. Ferner stellt der Generator **65** auf der Leitung **80** Sortiereinrichtung **57** ein WAIT-Signal bereit, um einen Betrieb der Sortiereinrichtung **57** für eine Anzahl an Bit-Perioden zu verhindern, die dem Token folgen, so dass die Ausgabe auf den Daten- und Strobe-Leitungen in dem konstanten Zustand gehalten wird.

[0055] Wann immer ein Daten-Token, ein EOP-Token oder ein EOM-Token von der Verbindung ausgegeben wird, wird auf der Leitung **41** über eine Einheit **81** zum Dividieren durch acht ein Signal zu einem Ausgangs-Token-Zähler **82** in der Fluss-Steuer-Einheit **32** zugeführt. Der Zähler **82** hat die Funktion, die Anzahl an Token zu begrenzen, die durch eine Verbindung ausgegeben werden können, bis ein FCT-Token von dem Verbindungseingang ausgehend von der Verbindungsschnittstelle empfangen worden ist, dass die Ausgabe erhält. Dies verhindert, dass ein Speicher in der empfangenden Verbindungsschnittstelle aufgrund eines Empfangs von zu vielen ausgegebenen Token überläuft. Immer wenn ein FCT-Token an dem Verbindungseingang empfangen wird, wird der Zähler **82** erhöht und macht acht von der Verbindungsausgabe gesendeten Token bewirkt das Signal auf der Leitung **41** eine Verringerung der Zählung in dem Zähler **82**. Immer wenn der Zähler Null erreicht, wird über eine Synchronisiereinrichtung **83** eine Ausgabe auf der Leitung **40** bereitgestellt, um eine weitere Ausgabe von Daten ausgehend von der Schnittstelle **34** zu unterbinden. Auf diese Weise stellt der Zähler **82** eine Angabe des Puffer-Platzes bereit, der in einem Verbindungseingang an dem anderen Ende der Verbindung verfügbar ist, und die Anzahl von Räumen wird mit jedem Token, das ausgegeben wird, heruntergezählt, und beim Empfang eines FCT-Token um acht erhöht.

[0056] In dem in Fig. 3 gezeigten Verbindungsausgang haben Rücksetzeingänge und empfangene Taktpulse von einem Taktgeber, der für die Ausgangsschaltkreisanordnung der Verbindungsschnittstelle vorgesehen ist.

[0057] Die Eingangsschaltkreisanordnung und die Fluss-Steuer-Einheit **32** sind vollständiger in Fig. 4 gezeigt. Verbindungseingang **31** ist vollständig durch die Daten- und Strobe-Signale auf den Leitungen **25** und **26** getaktet. Die Signale werden jeweils über Verzögerungseinheiten **90** und **91** eingegeben, deren Verzögerung einen Verzögerungseinstellschaltkreis **92** eingestellt werden kann. Die Daten- und Strobe-Signale werden

dann durch die Verwendung eines Detektors **95** für ansteigende Datenflanken, eines Detektors **96** für abfallende Datenflanken, eines Detektors **97** für ansteigende Strobe-Flanken, eines Detektors **98** für abfallende Strobe-Flanken detektiert. Wenn Flanken detektiert werden, werden Ausgaben auf Leitungen, die mit einem Verteilschaltkreis **99** verbunden sind, ausgegeben. Die Detektoren werden durch vier Signale zurückgesetzt, die ausgehend von einem Bit-Verzögerungsschaltkreis **100** den Detektoren entsprechend zugeführt werden. Der Bit-Verzögerungsschaltkreis spricht auf Ausgaben von dem Verteilschaltkreis **99** an und setzt die Flankendetektoren nach einer geforderten Bit-Verzögerung zurück. Der Verteiler **99** gibt die Ausgabe von den vier Flankendetektoren ein und reiht die Eingaben so, um diese nacheinander auszugeben. Der Verteiler **99** weist vier Ausgänge auf, die zu einer Daten/Takt-Abfrageeinrichtung **101** führen, und die vier Ausgänge werden so gesteuert, dass zu einem Zeitpunkt nur ein Ausgang auf einem hohen Pegel ist, dies der ersten Eingabe entspricht, die angelegt wurde. Die Abfrageeinrichtung **101** in Verbindung mit den Verteilern **99** und Flankendetektoren arbeiten, um die Signale auf den Daten- und Strobe-Leitungen **25** und **26** zu vergleichen und die Daten zu dekodieren, um eine Datenausgabe **102** und ein Taktsignal **103** bereitzustellen. Die Abfrageeinrichtung **101** weist einen Zwischenspeicher auf, der durch die ansteigenden und abfallenden ?? von dem Verteiler **99** zurückgesetzt wird. Die Ausgabe des Zwischenspeichers ist zurück gewonnene Datensignal und, um das Zurücksetzen der Eingangslogik zu takten, arbeitet das Taktsignal auf **103** bei der halben Bitrate, die durch das Dekodieren zurückgewonnen wird. Jedes Mal, wenn von den Verteileinrichtungen **99** eine neue Ausgabe erhalten wird, wird die Ausgabe eines Zwischenspeichers in der Abfrageeinrichtung **101** gekippt (engl.: flipped). Die Verzögerungsleitungen **90** und **91** sind vorgesehen, damit der Dekodierer korrekt arbeitet und um es zu ermöglichen, die Daten- und Strobe-Signale innerhalb einer Bit-Periode zu synchronisieren. Um diese Synchronisation zu ermöglichen, werden DAT-Drucken von dem Ausgabeschaltkreis einer angeschlossenen Verbindungsschnittstelle gesendet. Diese bewirken gleichzeitige Übergänge auf beiden der Daten- und Strobe-Leitungen **25** und **26**. Die Token werden durch den Eingang dekodiert und erzeugen eines von zwei Ergebnissen, abhängig davon, ob das Datensignal vor oder hinter dem Strobe-Signal liegt. Das Daten-Token wird als P011 ausgegeben, aber jegliche relative Verzögerung zwischen den Daten- und Strobe-Signalen kann bewirken, dass das Token als P011 eingegeben wird, wenn das Strobe(-Signal) vor den Daten ist (Verzögerung Strobe) oder als P001 eingegeben wird, wenn die Daten vor dem Strobe(-Signal) sind (Verzögerung Daten). Fig. 7 zeigt die Position für die letzten vier Bit eines Daten-Token, wo das Datensignal relativ zu dem Strobe-Signal spät eingegeben wird und so die Flanke **140b** die Flanke **140a** führt. Dies wird als P011 dekodiert, indem der Signalpegel auf der Datenleitung nach Übergängen auf jeder Leitung ermittelt wird. Fig. 8 zeigt die äquivalente Position, wenn das Strobe-Signal relativ zu dem Datensignal spät eingegeben wird. Die Flanke **140b** folgt der Flanke **140a**. Dies wird als P001 dekodiert, indem der Signalpegel auf der Datenleitung nach Übergängen auf jeder Leitung ermittelt wird. Beim Zurücksetzen der Verzögerungseinheit **90** und **91** werden beide auf eine minimale Verzögerung eingestellt. Der Empfang eines Verzögere-Strobe-Signal-Token bewirkt, dass die Verzögerung **91** in dem Strobe-Signalweg um einen geringen festgelegten Wert kleiner wird, es sei denn sie ist bereits Null, in welchem Fall die Verzögerung in dem Datensignalweg **90** erhöht wird. Ein Empfang eines Verzögere-Daten-Signals verursacht die entgegengesetzte Wirkung. Auf diese Weise befindet sich zu allen Zeiten wenigstens eine der Verzögerungsleitungen auf einer minimalen Verzögerung.

[0058] Die Daten- und Taktsignale **102** und **103** sind jeweils mit einer Paritätsüberprüfungseinrichtung **105** einer Token-Synchronisationseinrichtung **106** und einem Eingangsschieberegister **107** verbunden. Das Register **107** besteht aus zwei Reihen von Master-Slave-Zwischenspeichern. Eine Kette ist an der ansteigenden Flanke des Takts getaktet und die andere an der abfallenden Flanke. Die Token-Synchronisationseinrichtung **106** ??extrahiert das Steuer/Daten-Flag an der zweiten Bit-Position jedes Token und zählt die geeignete Anzahl an Bit für dieses Token, die entweder vier oder zehn ist, abhängig davon, ob das Token ein Raten- oder Steuer-Token ist. Dies sorgt für Strobe-Signale auf der Leitung **108a**, um 4 Bit-Token ausgehend von dem Register **107** in einem Token-Eingangs-Register **109** zwischenzuspeichern, und auf einer Leitung **108b**, um 10 Bit-Token ausgehend von dem Register **107** dem Token-Eingangs-Register **109** zwischenzuspeichern, wobei auch auf der Leitung **110** der Paritätsüberprüfungseinrichtung **105** ein Strobe-Signal bereitgestellt wird, so dass die Paritätsüberprüfungseinrichtung **105** das Paritätsbit an dem Anfang jedes Token identifiziert und dieses mit der Anzahl an Bits vergleicht, die auf der Leitung **102** ausgehend von dem letzten Steuer- oder Daten-Flag in dem vorherigen Token übertragen worden sind. Wenn ein Paritätsfehler auftritt, wird auf einer Leitung **111** eine Ausgabe bereitgestellt. Diese wird über einen Eingang **112** zurück zu der Token-Synchronisationseinrichtung **106** zurückgeführt, um die Eingabe anzuhalten. Eine Detektion eines Paritätsfehlers kann verwendet werden, um ein Warnlicht einzuschalten oder um das System anzuhalten oder um das System zu aktivieren, um sich von dem Fehler durch erneutes Starten oder auf andere Weise zu erholen.

[0059] Wenn ein vollständiges Token in das Register **107** verschoben worden ist, hat die Token-Synchronisationseinrichtung die richtige Anzahl an Bit für dieses Token gezählt und speichert dann die Inhalte in das Register **109** hinein zwischen. Ein Token-gültig-Signal wird dann auf einer Leitung **112** ausgehend von der To-

ken-Synchronisationseinrichtung **106** einem Steuerkodedetektor **113** bereitgestellt, der das Token ausgehend von dem Register **109** erhält und das Bitmuster derjenigen Token identifiziert, die Steuer-Token zur Verwendung innerhalb der Verbindung sind. Diese Token sind in NULL, FCT, DAT und ESC. Andere Token, wie zum Beispiel ein Daten-Token EOP oder EOM werden ausgehend von dem Detektor **113** einer Leitung **115** ausgegeben und in einen FIFO **116** unter Steuerung eines Signals **113a** ausgehend von dem Detektor **113** geschrieben. Ein Signal **113b** ausgehend von dem Detektor **113** zu der Token-Synchronisationseinrichtung **106** gibt an, wann ein ESC-Token empfangen worden ist. Dies von der Token-Synchronisationseinrichtung gefordert, um die Länge des folgenden Token zu ermitteln. Wenn der Detektor **113** ein FCT-Token detektiert, stellt er auf einer Leitung **117** dem Ausgabe-Token-Zähler **82** eine Ausgabe bereit. Dies gibt an, dass die empfangene Verbindung nun bereit ist, mehr Token zu empfangen (bei diesem Beispiel gibt jedes Steuer-Token an, dass die empfangene Verbindung nun weitere acht Token aufnehmen kann). Das Signal auf der Leitung **117** erhöht daher den Zähler **82**, um zu gewährleisten, dass auf der Leitung **40** keine Ausgabe unterbunden wird.

[0060] Der FIFO **116** ist ein Speicher, der bei diesem Beispiel die Pufferung von acht Token ermöglicht. Um die Bandbreite zu verbessern, kann diese Pufferung bei diesem Beispiel auf sechzehn Token vergrößert werden. Der FIFO **116** stellt eine Schnittstelle zu dem Hauptrechner bereit, der die Nachricht erhält, und gibt auf einem Bus **124** Daten zu dem Hauptrechner sowie auf einer Leitung **125** ein Daten-gültig-Signal aus. Eine Übertragung von Daten ausgehend von dem FIFO **116** zu dem Hauptrechner wird in einem synchronisierten Hand-Shake-Betrieb bewirkt und auf einer Leitung **126** wird dem FIFO **116** ein Bestätigungssignal bereitgestellt, wenn deren Hauptrechner die Daten empfangen hat. Wenn der Hauptrechner den Empfang eines Token auf der Leitung **116** bestätigt, bestätigt dies, dass der FIFO **116** nun weiteren Raum aufgrund der Entfernung dieses Token aufweist und es wird auf einer Leitung **127** ein Signal der Fluss-Steuer **132** bereitgestellt. Das Signal auf Leitung **127** geht durch eine Einheit **128** zum Teilen durch acht hindurch und wird einem Eingabe-Token-Zähler **129** zugeführt. Dieser Zähler **129** zählt Token, wenn sie ausgehend von der Verbindung in den Hauptrechner umgeben werden. Wenn acht Token von der Einheit **128** gezählt worden sind, wird der Zähler **129** erhöht. Dieser Zähler **129** weist einen Zählung-entspricht-Null-Detektor auf, der mit dem Verbindungsausgangstaktgeber über eine Einheit **130** synchronisiert ist und auf der Leitung **46** ein Signal zuführt, das auffordert, dass ein FCT-Token zu senden ist. Der Zähler **129** empfängt auf der Leitung **41** ausgehend von der Ausgangsschnittstelle **34** auch eine Eingabe, um zu bestätigen, dass ein FCT-Token gesendet worden ist, wodurch die Zählung in dem Zähler **129** verringert wird.

[0061] Das Signal auf der Leitung **125** von dem FIFO **116** wird verwendet, um anzugeben, dass der FIFO nicht leer ist, und dieses Signal wird durch einen Synchronisationsschaltkreis **131** mit dem Hauptrechnertaktgeber synchronisiert. Das Bestätigungssignal auf der Leitung **126** muss nicht mit dem Verbindungseingang **31** synchronisiert werden.

[0062] Wenn der Detektor **113** in DAT-Token als P001 detektiert, was der Fall ist, wenn die Daten vor dem Strobe-Signal liegen, stellt der Detektor **113** der Paritätsüberprüfungseinrichtung **105** auf der Leitung **104** ein invertiertes Paritätssignal bereit. Dies ist der Fall, weil die Änderung bei der Eingabe des DAT-Token einen Paritätsfehler relativ zu der Ausgabe des Token verursachen würde, und diese Paritätsinversion ermöglicht weiterhin, dass die Paritätsüberprüfung gültig ist.

[0063] Es ist verständlich, dass die obige Anordnung eine einfache Hochgeschwindigkeitsschnittstelle zwischen kommunizierenden Vorrichtungen in einem Netzwerk bereitstellt. Sie erlaubt die Übertragung von Nachrichten in Paketen variabler Länge und hat die Fähigkeit, Steuer-Token sowie Daten-Token zu senden, wobei diese Steuer-Token verwendet werden, um einen Fluss zwischen angeschlossenen Verbindungen zu steuern, als auch den Betrieb innerhalb der Verbindungen selbst zu steuern. Die Verwendung von Daten- und Strobe-Signalen, bei denen das Strobe-Signal Übergänge nur an Bit-Grenzen aufweist, wo kein Übergang auf dem Datensignal vorkommt, sorgt für einen verbesserten Betrieb bei hohen Bit-Frequenzen ohne Verlust von Daten, wenn die Token dekodiert werden. Eine automatische Synchronisation der Daten- und Strobe-Signale kann durch die Verwendung von DAT-Token erreicht werden, und um große Fehlanpassungen, die größer als eine Bit-Periode sind, zu berücksichtigen, kann das System durch Senden einer Anzahl von Synchronisations-Token bei einer geringeren Geschwindigkeit und nach einem vorläufigen Synchronisationsschaltbetrieb zu der Betriebsgeschwindigkeit und Senden weiterer DAT-Token betrieben werden, um eine Synchronisation bei der hohen Betriebsgeschwindigkeit zu bewirken. Ferner ist das Bereitstellen eines Paritätsbit in jedem Token im Speziellen beim Bereitstellen adäquater Überprüfungen vorteilhaft, wenn bei hoher Geschwindigkeit gearbeitet wird. Jedes Paritätsbit befindet sich an einer festgelegten Position in jedem Token, wie bei diesem Beispiel das erste Bit in jedem Token ist. Das Paritätsbit in jedem Token sorgt für eine Überprüfung des vorherigen Token dahingehend, dass es eine Angabe über die Anzahl von Bit bereitstellt, die seit dem letzten Daten- oder Steuer-Flag und einschließlich desselben übertragen worden sind.

[0064] Die Erfindung ist nicht auf die Einzelheiten des vorherigen Beispiels eingeschränkt. Bei dem obigen Beispiel ist jedes Daten- oder Steuer-Flag ein einzelnes Bit, das eine Bit-Längen-Angabe für die variable Token-Länge bildet, aber jedes Token kann mehr als ein Bit aufweisen, um die Token-Länge anzugeben.

[0065] Die oben beschriebene Schnittstelle kann in einem Computernetzwerk mit adressierbaren Kommunikationskanälen einschließlich virtueller Kanäle verwendet werden, wie in unserer gleichzeitig anhängigen UK-Patentanmeldung Nr. 8915136.9 beschrieben. In solchen Fällen kann ein Paket in einem oder mehreren Daten-Token die Adresse eines Kanals oder eines virtuellen Kanals aufweisen, der beim Bewirken der Kommunikation zu verwenden ist.

[0066] Die Zustandstabellen, die die Übergangszustände der Zustandsmaschinen oben angeben, sind wie folgt:
In den folgenden Zustandstabellen werden die folgenden Konventionen verwendet.

[0067] Ausgaben sind, sofern nicht explizit anders festgestellt (z. B. Strobe = InvertStrobe), eine Funktion des Zustandes und nicht von Eingaben.

[0068] Wenn ein Zustand keine gültigen Eingaben aufweist, bleibt die Zustandsmaschine in dem aktuellen Zustand.

[0069] Wo ein Zustand als "Any" spezifiziert ist und es eine gültige Eingabe gibt, dann hebt der spezifizierte Zustandsübergang alle anderen Übergänge auf, z. B. wenn in der vorliegenden Tabelle "Reset" ausgegeben wird, vollzieht die Zustandsmaschine einen Übergang zu dem Zustand "00" unabhängig der anderen Eingaben.

Schlüssel

~	NICHT
∧	UND
v	ODER
=	Zuordnung. z. B. A = B bedeutet, dass die Ausgabe A den Wert der Eingabe B annimmt; A = 0 bedeutet, dass die Ausgabe A auf 0 gesetzt wird.

Taktgebung

[0070] Bei diesem Beispiel sind alle Ausgabezustandsmaschinen synchron. Der Verbindungseingang enthält synchrone und asynchrone Zustandsmaschinen. Die synchronen Zustandsmaschinen in den Verbindungseingang sprechen auf beide Flanken des Takts an.

Daten/Strobe-Signal-Kodierer (62)

Zustand	Ausgaben	Eingaben	Nächster Zustand
Jeder	Jeder	Zurücksetzen	00
00	Daten = 0 Aktivierung = invertierte Aktivierung	~ Freigabe	00
		Freigabe ∧ ~ Dateneingabe	01
		Freigabe ∧ Dateneingabe	10
01	Daten = 0 Aktivierung = ~ invertierte Aktivierung	~ Freigabe	01
		Freigabe ∧ ~ Dateneingabe	00
		Freigabe ∧ Dateneingabe	11
10	Daten = 1 Aktivierung = ~ invertierte Aktivierung	~ Freigabe	10
		Freigabe ∧ ~ Dateneingabe	00
		Freigabe ∧ Dateneingabe	11
11	Daten = 1 Aktivierung = ~invertierte Aktivierung	~ Freigabe	11
		Freigabe ∧ ~ Dateneingabe	01
		Freigabe ∧ Dateneingabe	10

Token-Sortiereinrichtung (52)

Zustand	Ausgaben	Eingaben	Nächster Zustand
Jeder	Jeder	Zurücksetzen	Warten auf Token
Warten auf Token	Aktiviere Prioritätszuordnungseinrichtung	Jede Anfrage	Token erhalten
		~ Jede Anfrage	Warten auf Token
Token erhalten	Rücksetzen der Anfrage	Escape-Steuerbefehl	Escape-Token
		~ Escape-Steuerbefehl	Startzustand
Escape-Token	Sende ESC-Token	Nächstes Token	Startzustand
		~ Nächstes Token	Escape-Token
Startzustand	Start	~ Nächstes Token	Startzustand
		Nächstes Token	Warten auf Token

Datensynchronisations-Token-Generator (65)

Zustand	Ausgaben	Eingaben	Nächster Zustand
Jeder	Jeder	Zurücksetzen	Warten auf Beginn von DAT
Warten auf Beginn von DAT		Start DAT	1
1			2
2	Warten		3
3	Warten Invertierte Aktivierung		4
4	Warten		5
5	Warten		6
6	Warten		Warten auf Beginn von DAT

Paritäts-Generator (59)

Zustand	Ausgaben	Eingaben	Nächster Zustand
Any	Next	Reset	P1D0
P0D0	0	ResetParity∧0	P1D0
		~ResetParity∧0	P0D0
		ResetParity∧1	P0D1
		~ResetParity∧1	P1D1
P0D1	~EnableParity	ResetParity∧0	P1D0
		~ResetParity∧0	P0D0
		ResetParity∧1	P0D1
		~ResetParity∧1	P1D1
P1D0	EnableParity	ResetParity∧0	P1D0
		~ResetParity∧0	P1D0
		ResetParity∧1	P0D1
		~ResetParity∧1	P0D1
P1D1	1	ResetParity∧0	P1D0
		~ResetParity∧0	P1D0
		ResetParity∧1	P0D1
		~ResetParity∧1	P0D1

Tokenanforderungs-Zwischenspeicher (**43**)

[0071] Dies sind tatsächlich 4 identische, separate Zustandsmaschinen, eine für jeden von DAT, FCT, Data und Null. Die Zustandstabelle für eine dieser ist unten gezeigt. "Send" entspricht Send-DAT, SendFCT etc., TokenRequest entspricht DATRequest etc. (Eingaben an Token-Prioritätszuordnungseinrichtung), TokenSent entspricht FCTSent etc. und TokenPrioritised entspricht DATPrioritised, FCTPrioritied etc. (Ausgaben von Token-Prioritätszuordnungseinrichtung). Es gibt eine zusätzliche Logik, um das "AnyRequest"-Signal zu erzeugen.

Zustand	Ausgaben	Eingaben	Nächster Zustand
Any		Reset	NoRequest
NoRequest		Send	Request
Request	TokenRequest	~(ResetRequest∧TokenPrioritised)	Request
		ResetRequest ∧TokenPrioritised	Acknowledge
Acknowledge	TokenSent		NoRequest

AnyRequest = NullRequestvFCTRequestvDATRequestvDataFequest

Sortiereinrichtung (57)

Zustand	Ausgaben	Eingaben	Nächster Zustand
Any	Any	Reset	WaitingForStart
WaitingForStart		Start	Start1
		~Start	WaitingForStart
Start1	Load, NextToken	DAT	StartDAT2
		~DAT∧Control	StartControl2
		~DAT∧~Control	StartData2
1	Load, EnableOut, NextToken	DAT	DAT2
		~DAT∧Control	Control2
		~DAT∧~Control	Data2
StartData2			Data3
Data2	EnableOut		Data3
Data3	EnableParityOut, Shift, EnableOut		Data4
Data4	Shift, EnableOut, ResetParity		Data5
Data5	Shift, EnableOut		Data6
Data6	Shift, EnableOut		Data7
Data7	Shift, EnableOut		Data8
Data8	Shift, EnableOut		Data9
Data9	Shift, EnableOut		Data10
Data10	Shift, EnableOut	Start	1
		~Start	Data11
Data11	Shift, EnableOut		Data12
Data12	EnableOut		WaitingForStart
StartControl2			Control3
Control2	EnableOut		Control3
Control3	EnableParityOut, Shift, EnableOut		Control4
Control4	Shift, EnableOut, ResetParity	Start	1
		~Start	Control5
Control5	Shift, EnableOut		Control6
Control6	EnableOut		WaitingForStart
StartDAT2	EnableParityOut		DAT3
DAT2	EnableOut		DAT3
DAT3	EnableParityOut, Shift, EnableOut, StartDAT		DAT4
DAT4	Shift, EnableOut, ResetParity		DAT5
DAT5	Shift, EnableOut		DAT6
DAT6	EnableOut		DAT7
DAT7		Wait	DAT7
		~Wait	WaitingForStart

Hauptrechnerschnittstelle (34)

Zustand	Ausgaben	Eingaben	Nächster Zustand
Any	Any	Reset	Empty
Empty		DataValid ∧ ~ InhibitDataOutput	Write
Write	WriteDataRegister, SendDataToken		Acknowledge
Acknowledge	DataAcknowledge		WaitingToSend
WaitingToSend		DataTokenSent	Empty
		~ DataTokenSent	WaitingToSend

Steuerkode-ROM & Multiplexer (53)

Zustand	Ausgaben	Eingaben	Nächster Zustand
Any	Any	SendESC	ESC
		~ SendESC∧Data	Data
		~ SendESC∧DAT	DAT
		~ SendESC∧FCT	FCT
		~ SendESC∧Null	Null
ESC	Control,d0,d1		
Data	d0–d7 = dataIn0–7 Control = ControlFlagIn		
DAT	d0, d1, DAT		
FCT	Control		
Null	Control		

Ausgabe-Schieberegister (**54**)

[0072] In der folgenden Zustandstabelle sind d0-7 und "Control" Boolean-Variablen, die abgefragt werden, wenn sie als Eingaben für die Zustandsmaschine spezifiziert sind, d. h. ihre Werte werden in den Zuständen "C" bis "7" nicht geändert.

Zustand	Ausgänge	Eingänge	Nächster Zustand
Any	Any	Load, d0–7,Control	C
C	Control	Shift	0
0	d0	Shift	1
1	d1	Shift	2
		Load, d0–7,Control	C
2	d2	Shift	3
3	d3	Shift	4
4	d4	Shift	5
5	d5	Shift	6
6	d6	Shift	7
7	d7	Shift	Hold
Hold	0		Hold

Token-Prioritätszuordnungseinrichtung (50)

Zustand	Ausgänge	Eingänge	Nächster Zustand
Any	Any	Enable∧DATRequest	DAT
		Enable∧FCTRequest-∧~DATRequest	FCT
		Enable∧DataRequest-∧~FCTRequest-∧~DATRequest	Data
		Enable∧NullRequest-∧~DataRequest-∧~FCTRequest-∧~DATRequest	Null
DAT	SendESC,SendDAT	~Enable	DAT
FCT	SendFCT	~Enable	FCT
Data	SendData	~Enable	Data
Null	SendESC,SendNull	~Enable	Null

[0073] Die Flankendetektoren, Verteileinrichtungen und Daten/Taktabfrageeinrichtungen sind asynchrone Zustandsmaschinen und daher ändert sich ein Zustand, sobald sich eine Eingabe ändert. Die anderen Zustandsmaschinen sind bei der halben Baud-Rate getaktet (d. h. 50 MHz für 100 MBaud) und sind synchron.

Flankendetektor (asynchron) (**95**-**98**)

[0074] In der folgenden Tabelle gilt. Reset = Main Reset v ResetEdgeDetector. Dieser Flankendetektor detektiert ansteigende Flanken. Bei Detektoren für abfallende Flanken wird die Eingabe ausgehend von den Verzögerungsleitungen invertiert.

Zustand	Ausgaben	Eingaben	Nächster Zustand
LowNoEdge		Reset ∧ InputHigh	HighNoEdge
		~Reset ∧ InputHigh	HighEdge
HighNoEdge		InputLow	LowNoEdge
HighEdge	EdgeDetected	Reset ∧ InputHigh	HighNoEdge
		Reset ∧ InputLow	LowNoEdge
		~Reset ∧ InputLow	LowEdge
LowEdge		Reset ∧ InputHigh	HighNoEdge
		Reset ∧ InputLow	LowNoEdge
		~Reset ∧ InputHigh	HighEdge

Verteileinrichtungen (asynchron) (**99**)

[0075] Es gibt 6 Verteileinrichtungen, eine zwischen jedem Eingangspaar

Verteileinrichtung	Eingang 1	Eingang 2	Ausgang 1	Ausgang 2
1	DataRising	Data Falling	DataRising1	Data Falling1
2	DataRising	StrobeRising	DataRising2	StrobeRising1
3	DataRising	StrobeFalling	DataRising3	StrobeFalling1
4	DataFalling	StrobeRising	DataFalling	StrobeRising2
5	DataFalling	StrobeFalling	DataFalling	StrobeFalling2
6	StrobeRising	StrobeFalling	StrobeRising3	StrobeFalling3

[0076] Die Zustandstabelle für eine der Verteileinrichtungen ist unten gezeigt:

Zustand	Ausgänge	Eingänge	Nächster Zustand
00		Input 1 High	10
		Input 2 High	01
01	Output2	Input 1 High	11 (2 first)
		Input 2 Low	00
10	Output1	Input 1 Low	00
		Input 2 High	11 (1 first)
11 (1 first)	Output1	Input 1 Low	01
		Input 2 Low	10
11 (2 first)	Output2	Input 1 Low	01
		Input 2 Low	10

[0077] Die Ausgaben von den einzelnen Verteileinrichtungen werden wie folgt kombiniert, um die vier Ausgaben ausgehend von dem Modul an Verteileinrichtungen in dem Diagramm zu erreichen.

DataRisingArbitrated = DataRising1∧DataRising2∧DataRising3

DataFallingArbitrated = DataFalling1∧DataFalling2∧DataFalling3

StrobeRisingArbitrated = StrobeRising1∧StrobeRising2∧StrobeRising3

StrobeFallingArbitrated = StrobeFalling1∧StrobeFalling2∧StrobeFalling3

Daten/Takt-Ausleseeinrichtung (asynchron) (101)

Zustand	Ausgänge	Eingänge	Nächster Zustand
Any	Any	Reset	Data0Strobe0
Data0Strobe0	Data = 0, Clock = 0	DataRisingArbitrated	Data1Strobe0
		StrobeRisingArbitrated	Data0Strobe1
Data0Strobe1	Data = 0, Clock = 1	DataRisingArbitrated	Data1Strobe1
		StrobeFallingArbitrated	Data0Strobe0
Data1Strobe1	Data = 1, Clock = 0	DataFallingArbitrated	Data0Strobe1
		StrobeFallingArbitrated	Data1Strobe0
Data1Strobe0	Data = 1, Clock = 1	DataFallingArbitrated	Data0Strobe0
		StrobeRisingArbited	Data1Strobe1

[0078] Die Taktgeberausgabe wird relativ zu der Datenausgabe verzögert, um die Vorbereitungszeit der Schieberegister, Paritätsüberprüfungseinrichtungen und Token-Synchronisationseinrichtungen zuzulassen.

[0079] Die folgenden Zustandsmaschinen sind synchron und können einen Zustand sowohl bei ansteigenden als auch bei abfallenden Taktflanken ändern.

Paritäts-Überprüfungseinrichtung (synchron) (**105**)

[0080] Das ResetParity-Signal wird während des letzten Bit jedes Token ausgegeben. Wenn dies auftritt, wird der Zustand (zusammen mit den Daten- und InvertParity-Eingaben) verwendet, um zu ermitteln, ob es einen Paritätsfehler gegeben hat.

Zustand	Ausgänge	Eingänge	Nächster Zustand
Any	Any	Reset	0
0		~ResetParity∧Data = 0	0
		~ResetParity∧Data = 1	1
		ResetParity ∧~InvertParity ∧Data = 0	ParityError
		ResetParity ∧~InvertParity ∧Data = 1	0
		ResetParity ∧InvertParity ∧Data = 0	0
		ResetParity ∧InvertParity ∧Data = 1	ParityError
1		~ResetParity∧Data = 0	1
		~ResetParity∧Data = 1	0
		ResetParity ∧~InvertParity ∧Data = 0	0
		ResetParity ∧~InvertParity ∧Data = 1	ParityError
		ResetParity ∧InvertParity ∧Data = 0	ParityError
		ResetParity ∧InvertParity ∧Data = 1	0
ParityError	ParityErrorOut	~Reset	ParityError

Token-Synchronisationseinrichtung (synchron) (106)

Zustand	Ausgänge	Eingänge	Nächster Zustand
Any	Any	Reset	ResetState
ResetState			FirstParity
FirstParity	ResetParity	Data = 0	DataFlag
		Data = 1	ControlFlag
NextParity	ResetParity, TokenValid	~Data ∧ ~ESCCode	DataFlag
		Data ∨ ESCCode	ControlFlag
DataFlag			Data0
Data0		~ParityError	Data1
		ParityError	Halt
Data1			Data2
Data2			Data3
Data3			Data4
Data4			Data5
Data5			Data6
Data6			Data7
Data7	LatchLongToken		NextParity
ControlFlag			Control0
Control0			Control1
Control1	LatchShortToken		NextParity
Halt		~Reset	Halt

Steuerkode-Detektor (synchron) (113)

Zustand	Ausgänge	Eingänge	Nächster Zustand
Any	Any	Reset	
NextToken		~TokenValid	NextToken
		TokenValid∧x111	ESC
		TokenValid∧x100	FCT
		TokenValid∧x0xxxxxxx	FIFOData
		TokenValid∧x101	FIFOData
		TokenValid∧x110	FIFOData
ESC	ESC	~TokenValid	ESC
		TokenValid∧x111	Null
		TokenValid∧x001	DAT Up
		TokenValid∧x011	DAT Down
FCT	FCT Received		NextToken
FIFOData	WriteToFIFO		NextToken
Null			NextToken
DAT Up	Up		NextToken
DAT Down	Down		NextToken

Steuer-Zustandstabelle (**33**)

[0081] Dies ist ein Beispiel des Typs einer Steuer-Zustandsmaschine, die verwendet werden könnte. Die Verbindung wird zuerst auf einen langsamen Takt geschaltet und es wird eine Anzahl (in diesem Fall **100**) von Datensynchronisations-Token gesendet. Die Verbindung wird dann auf eine höhere Geschwindigkeit geschaltet. Datensynchronisations-Token werden dann periodisch gesendet.

Zustand	Ausgänge	Eingänge	Nächster Zustand
Any	Any	Reset	Start
Start	SlowClock		0
0	SendDAT	DATSent	1
1	SendDAT	DATSent	2
...			
100	SwitchToHigherSpeed		Running
Running	SendNull		R1
R1	SendNull		R2
...			
R100	SendDAT	DATSent	Running

FIGURENLEGENDE

Fig. 1

MICROPROCESSOR	MIKROPROZESSOR
PERIPHERAL	PERIPHERIEGERÄT
DISK CONTROLLER	FESTPLATTENSTEUEREINRICHTUNG
DISK	FESTPLATTE
ROUTING DEVICE	VERWEITERLEITUNGSVORRICHTUNG
LINK UNIT	VERBINDUNGSEINHEIT

Fig. 2

DATA TO BE OUTPUT	AUSZUGEBENDE DATEN
DATA VALID ACKNOWLEDGE	DATEN-GÜLTIG-BESTÄTIGUNG
INPUT DATA	EINGANG DATEN
ACKNOWLEDGE	BESTÄTIGUNG
DATA VALID	DATEN GÜLTIG
PARITY ERROR OUT	PARITÄTSFEHLERAUSGABE
HOST CLOCK (TO SYNCHRONISER ONLY)	HAUPTRECHNERTAKTGEBER (NUR ZU SYNCHRONISATIONSEINRICHTUNG)
DATA OUT	DATEN AUS
DATA IN	DATEN EIN
STROBE OUT	STROBE AUS
STROBE IN	STROBE EIN
LINK OUTPUT	VERBINDUNGSAUSGANG
LINK INPUT	VERBINDUNGSEINGANG
FLOW CONTROL UNIT	FLUSS-STEUER-EINHEIT
CONTROL	STEUERUNG
OUTPUT CLOCK	AUSGANG TAKTGEBER

Fig. 3

DATA	DATEN
STROBE	STROBE-SIGNAL
HOST I/F	HAUPTRECHNER-SCHNITTSTELLE
DATA TO SEND	ZU SENDENDE DATEN
DATA VALID	DATEN GÜLTIG
DATA ACKNOWLEDGE	DATENBESTÄTIGUNG
CLOCK	TAKT
INHIBIT DATA OUTPUT	UNTERBINDE DATENAUSGABE
TOKEN SENT	TOKEN GESENDET
DATA REGISTER	DATENREGISTER
TOKEN REQUEST LATCHES	TOKEN-ANFRAGE-ZWISCHENSPEICHER
FCT SENT	FCT GESENDET
TOKEN SEQUENCER	TOKEN-SORTIEREINRICHTUNG
TOKEN PRIORITISER	TOKEN-PRIORITÄTSZUORDNUNGSEINRICHTUNG
CONTROL COLDE ROM & DATA MULTIPLEXOR	STEUERCODE-ROM & DATEN-MULTIPLEXER
OUTPUT SHIFT REGISTER	AUSGANGSSCHIEBEREGISTER
PARITY GENERATOR	PARITÄTS-GENERATOR
DATA/STROBE ENCODER	DATEN/STROBE-SIGNALKODIERER
SEQUENCER	SORTIEREINRICHTUNG
DATA ALIGNMENT TOKEN GENERATOR	DATENSYNCHRONISATIONS-TOKEN-GENERATOR
OUTPUT DRIVERS	AUSGANGSTREIBER
SEND DAT	SENDE DAT
SEND FCT	SENDE FCT
SEND DATA TOKEN	SENDE DATEN-TOKEN
SEND NULL	SENDE NULL

Fig. 4

DATA	DATEN
STROBE	STROBE-SIGNAL
DELAY	VERZÖGERUNG
DELAY ADJUST	VERZÖGERUNGSEINSTELLUNG
FLOW CONTROL UNIT	FLUSS-STEUEREINHEIT
TOKEN SENT	TOKEN GESENDET
ACK	BESTÄTIGUNG
OUTPUT TOKEN COUNTER	TOKEN-AUSGANGSZÄHLER
INPUT TOKEN COUNTER	TOKEN-EINGANGSZÄHLER
INPUT SYNC	EINGANG SYNCHRONISATION
INHIBIT DATA OUTPUT	UNTERBINDE DATENAUSGABE
OUTPUT CLOCK	AUSGANGSTAKT
HOST CLOCK	HAUPTRECHNERTAKTGEBER
PARITY ERROR OUT	PARITÄTSFEHLER AUS

Fig. 5

PARITY BIT	PARITÄTSBIT
"DATA" FLAG	"DATEN"-FLAG
DATA	DATEN
"CONTROL" FLAG	"STEUER"-FLAG
STROBE	STROBE-SIGNAL

Fig. 6

PARITY BIT	PARITÄTSBIT
DATA	DATEN
STROBE	STROBE-SIGNAL
INHIBIT DATA OUTPUT	UNTERBINDE DATENAUSGABE
FCT SENT	FCT GESENDET
SEND FCT	SENDE FCT
DIVIDE BY 8	DIVIDIERE DURCH 8
OUTPUT TOKEN COUNTER	TOKEN-AUSGABEZÄHLER
SYNC	SYNCHRONISATION
DELAY	VERZÖGERUNG
DATA RISING EDGE DETECTOR	DATEN-ANSTEIGENDE FLANKE-DETEKTOR
DATA FALLING EDGE DETECTOR	DATEN-ABFALLENDE FLANKE-DETEKTOR
STROBE RISING EDGE DETECTOR	STROBE-ANSTEIGENDE FLANKE-DETEKTOR
STROBE FALLING EDGE DETECTOR	STROBE-ABFALLENDE FLANKE-DETEKTOR
BIT DELAY	BIT-VERZÖGERUNG
ARBITERS	VERTEILEINRICHTUNGEN
DATA/CLOCK EXTRACTOR	DATEN/TAKT-AUSLESEEINRICHTUNG
PARITY CHECKER	PARITÄTSÜBERPRÜFUNGSEINRICHTUNG
TOKEN SYNCHRONISER	TOKEN-SYNCHRONISATIONSEINRICHTUNG
INPUT SHIFT REGISTER	EINGANGSSCHIEBEREGISTER
TOKEN INPUT REGISTER	TOKEN-EINGANGSREGISTER
PARITY ERROR OUT	PARITÄTSFEHLER AUS
CONTROL CODE DETECTOR	STEUERKODEDETEKTOR
INPUT FIFO	EINGABE FIFO
DATA OUT	DATEN AUS
DATA VALID	DATEN GÜLTIG

Fig. 7

PARITY BIT	PARITÄTSBIT
DATA	DATEN
STROBE	STROBE-SIGNAL

Fig. 8

PARITY BIT	PARITÄTSBIT
DATA	DATEN
STROBE	STROBE-SIGNAL

Patentansprüche

1. Kommunikationsschnittstelle (**21**) für die Verwendung in einem Kommunikationssystem, das einen Computer mit wenigstens einer anderen Vorrichtung verbindet, wobei die Schnittstelle eine Ausgabeschaltkreisanordnung (**30**) zur Ausgabe von Nachrichten und eine Eingabeschaltkreisanordnung (**31**) zur Eingabe von Nachrichten umfasst, wobei die Ausgabeschaltkreisanordnung eine Steuerschaltkreisanordnung (**33**) und eine Kodierschaltkreisanordnung (**62**) aufweist, um zwei parallele Ausgaben bereitzustellen, eine in Form eines Datensignals (**25**) mit einem seriellen Bitmuster, das wenigstens einen Teil der Ausgabenachricht bildet, und die andere in Form eines Strobe-Signals (**26**), das, wenn Daten in das Datensignal ausgegeben werden, Signalübergänge nur an Bitgrenzen aufweist, wo es keinen Übergang auf das parallele Datensignal gibt, und die Eingangsschaltkreisanordnung eine Dekodierschaltkreisanordnung mit zwei Eingängen aufweist, einen ersten Eingang zum Empfang eines Datensignals und einen zweiten Eingang zum Empfang eines Strobe-Signals, wobei die Dekodierschaltkreisanordnung ausgelegt ist, sowohl auf beide der Daten- und Strobe-Signale anzusprechen, um Daten, die in dem Datensignal kodiert sind, zu dekodieren, **dadurch gekennzeichnet**, dass die Ausgabeschaltkreisanordnung ferner eine Synchronisations-Token erzeugende Schaltkreisanordnung (**65**) umfasst, eine Ausgabe eines Synchronisations-Token zu bewirken, das in das Datensignal auszugeben ist, und wobei die Ausgabeschaltkreisanordnung ausgelegt ist, zu bewirken, dass die beiden parallelen Ausgaben, simultane Übergänge haben, wenn ein Synchronisations-Token übertragen wird, und die Eingangsschaltkreisanordnung ferner eine Synchronisations-Schaltkreisanordnung aufweist, die auf das Synchronisati-

ons-Token anspricht, um eine Synchronisation von Signalen an den beiden Eingängen zu bewirken.

2. Kommunikationsschnittstelle nach Anspruch 1, bei der jedes Token ein Flag aufweist, das die Bit-Länge des Token angibt.

3. Kommunikationsschnittstelle nach Anspruch 1, bei der die Steuerschaltkreisanordnung einen Paritätsbit-Generator (**59**) aufweist, um ein Paritätsbit zur Implikation in jedes Token zu erzeugen.

4. Kommunikationsschnittstelle nach einem der vorhergehenden Ansprüche, bei der die Steuerschaltkreisanordnung einen Flag-Bit-Generator aufweist, um ein Flag-Bit zur Implikation in jedes Token zu erzeugen, um jedes Token als Daten-Token oder Steuer-Token zu identifizieren.

5. Kommunikationsschnittstelle nach einem der vorhergehenden Ansprüche, bei der die Eingangsschaltkreisanordnung eine Verzögerungsschaltkreisanordnung (**90**, **91**), die mit jedem der beiden Eingänge und dem Dekodierer verbunden ist, und eine Einrichtung aufweist, um die Verzögerung (**92**) an einem der oder an beiden Eingängen vor dem Dekodieren zu variieren.

6. Kommunikationsschnittstelle nach einem der vorhergehenden Ansprüche, bei die Ausgabeschaltkreisanordnung eine Fluss-Steuereinrichtung zum Erzeugen eines Fluss-Steuer-Token zur Ausgabe an eine angeschlossene Kommunikationsschnittstelle aufweist, und die Eingangsschaltkreisanordnung eine Einrichtung aufweist, die auf eine Eingabe eines Fluss-Steuer-Token anspricht, um den Betrieb der Ausgabeschaltkreisanordnung beim Ausgeben weiterer Datensignale zu steuern.

7. Verfahren, um eine Kommunikation zwischen wenigstens zwei miteinander verbundenen Vorrichtungen zu bewirken, wobei mindestens eine der Vorrichtungen einen Computer umfasst, wobei das Verfahren umfasst, parallele Datensignal- und Strobe-Signal-Kommunikationswege zwischen zwei Verbindungsschnittstellen aufzubauen, die jeweils mit einer entsprechenden der beiden Vorrichtungen verbunden sind, ein serielles Bitmuster zu kodieren, um ein Datensignal als wenigstens ein Teil einer Ausgabenachricht zu bilden, und das Datensignal auf den Datensignalweg ausgehend von einer der Verbindungsschnittstellen und auf den Strobe-Signalweg ausgehend von der einen Verbindungsschnittstelle ein Strobe-Signal auszugeben, das, wenn Daten in das Datensignal ausgegeben werden, Signalübergänge nur an Bitgrenzen aufweist, wo es keinen Übergang auf das parallele Datensignal gibt, das Datensignal und das Strobe-Signal an der anderen der Verbindungsschnittstellen parallel einzugeben, und auf beide der Daten- und Strobe-Signale anzusprechen, um Daten, die in dem Datensignal kodiert sind, zu dekodieren, und, gekennzeichnet durch, ferner ein Synchronisations-Token bereit zu stellen, das in das Datensignal auszugeben ist, und Übergänge gleichzeitig zu bewirken, die in die Daten- und Strobe-Wege auszugeben sind, wenn das Synchronisations-Token auf dem Datenweg übertragen wird, und die gleichzeitigen Übergänge zu verwenden, um eine Synchronisation der Signale auf den Daten- und Strobe-Weganordnung zu bewirken, wenn sie eine Verbindungsschnittstelle eingeben werden.

8. Verfahren nach Anspruch 7, ferner umfassend ein Daten-Token bereitzustellen, in dem Daten, die auf den Datensignalweg ausgegeben werden, nur dann Signalübergänge aufweisen, wenn sich die Daten ändern.

9. Verfahren nach Anspruch 7 oder 8, ferner umfassend, vier unidirektionale Kommunikationswege zwischen jedem Paar von Verbindungsschnittstellen aufzubauen, wobei die vier Wege ein erstes paralleles Paar an Daten- und Signalwegen in einer Richtung und eine zweites paralleles Paar an Daten- und Signalwegen in der entgegengesetzten Richtung aufweisen.

10. Verfahren nach einem der Ansprüche 7 bis 9, ferner aufweisend, ein Flag-Bit zur Implikation in jedes Token zu erzeugen, um das Token als Daten-Token oder Steuer-Token zu identifizieren.

11. Verfahren nach Anspruch 10, ferner umfassend zwei sukzessive Steuer-Token zu bilden, um ein zusammengesetztes Token zu bilden, wobei das erste der Steuer-Token ein Bitmuster aufweist, das angibt, dass ein weiteres Token erforderlich ist, um die durch das zusammengesetzte Token angegebene Steuerung festzulegen.

12. Verfahren nach einem der Ansprüche 7 bis 11, umfassend ein Fluss-Steuer-Token zur Ausgabe durch eine Verbindungsschnittstelle zu bilden, ein Fluss-Steuer-Token ausgehend von der zweiten Schnittstelle an die erste Schnittstelle auszugeben, um der ersten Schnittstelle anzuzeigen, dass weitere Daten-Token an die zweite Schnittstelle ausgegeben werden können.

13. Verfahren nach einem der Ansprüche 7 bis 12, ferner umfassend, eine Zählung aufrechtzuerhalten, die durch Ausgabe von Token durch eine ausgebende Verbindungsschnittstelle angepasst wird, eine weitere Ausgabe von Daten-Token ausgehend von der genannten Verbindungsschnittstelle zu sperren, wenn die Zählung eine vorbestimmte Zahl erreicht, und die Zählung in Antwort auf eine Eingabe eines Fluss-Steuer-Token ausgehend von einer Verbindungsschnittstelle anzupassen, um die Ausgabe von weiteren Daten-Token zu ermöglichen.

14. Verfahren nach einem der Ansprüche 7 bis 13, bei dem jede Verbindungsschnittstelle ausgelegt ist, Daten-Token einer vorbestimmten Bit-Länge auszugeben, die länger als eine zweite vorbestimmte Bit-Länge für Steuer-Token ist.

15. Verfahren nach einem der Ansprüche 7 bis 14, bei dem jede Verbindungsschnittstelle ausgelegt ist, Steuer-Token von zwei Typen auszugeben, wobei ein erster Typ zum Steuern des Betriebs einer angeschlossenen Verbindungsschnittstelle und ein zweiter Typ zur Verwendung durch die Vorrichtung, die mit der zweiten Verbindungsschnittstelle verbunden ist.

Es folgen 7 Blatt Zeichnungen

Anhängende Zeichnungen

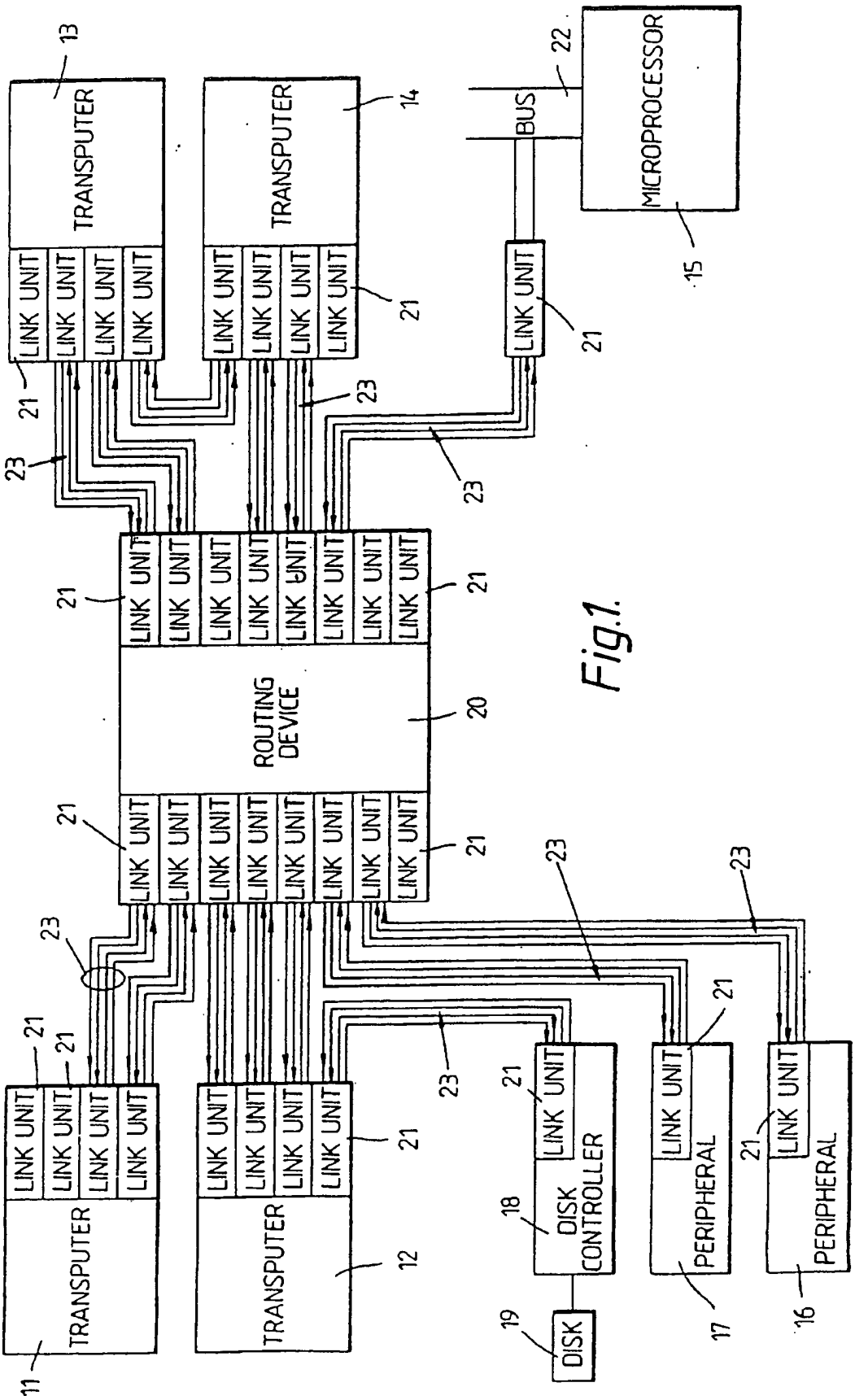


Fig.1.

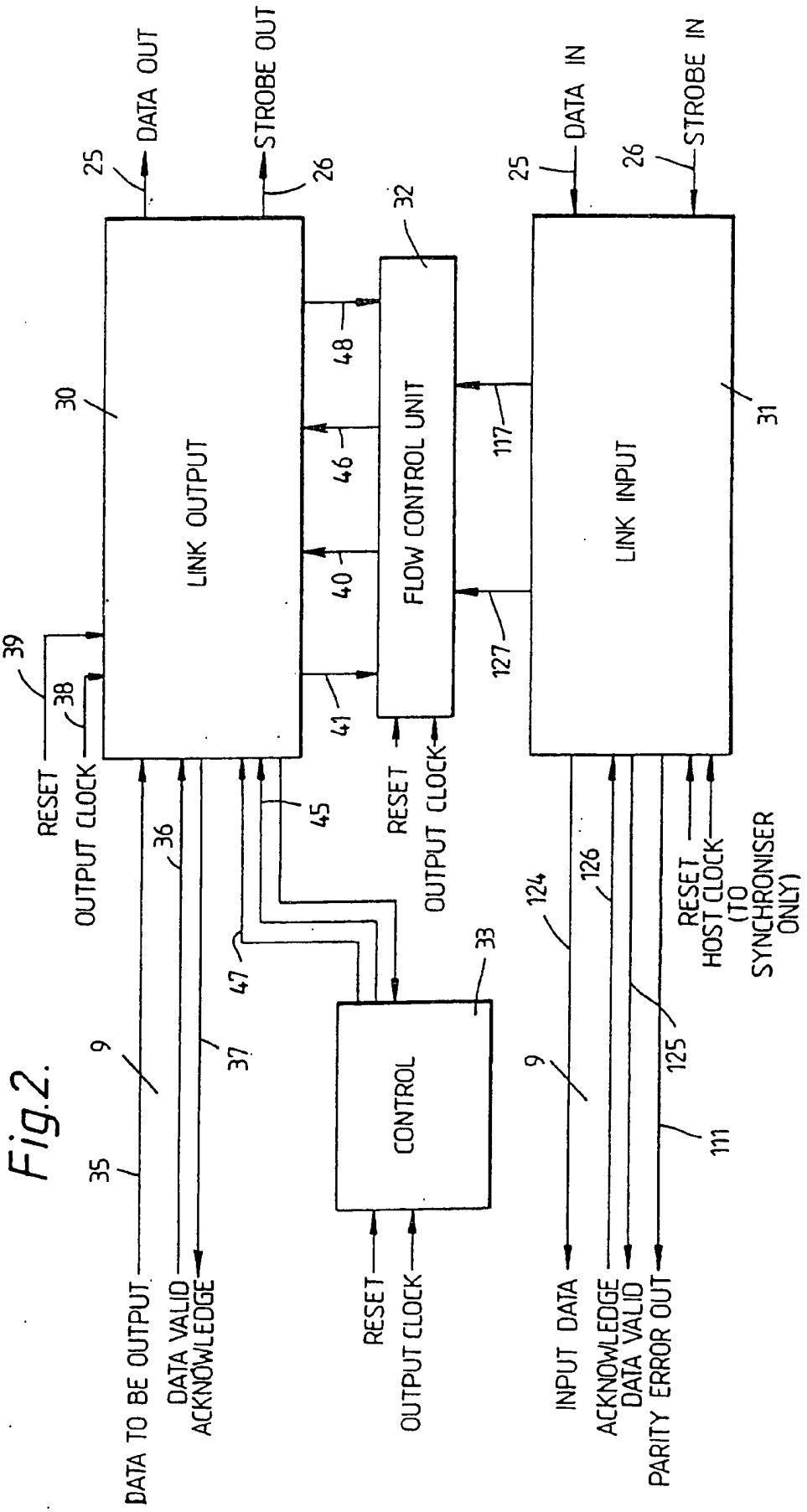

DATA OUT
STROBE OUT
DATA IN
STROBE IN
LINK OUTPUT
FLOW CONTROL UNIT
LINK INPUT
CONTROL
RESET
OUTPUT CLOCK
DATA TO BE OUTPUT
DATA VALID
ACKNOWLEDGE
INPUT DATA
ACKNOWLEDGE
DATA VALID
PARITY ERROR OUT
RESET
HOST CLOCK
(TO SYNCHRONISER ONLY)
25
26
30
31
32
33
35
36
37
38
39
40
41
45
46
47
48
111
117
124
125
126
127

Fig.2.

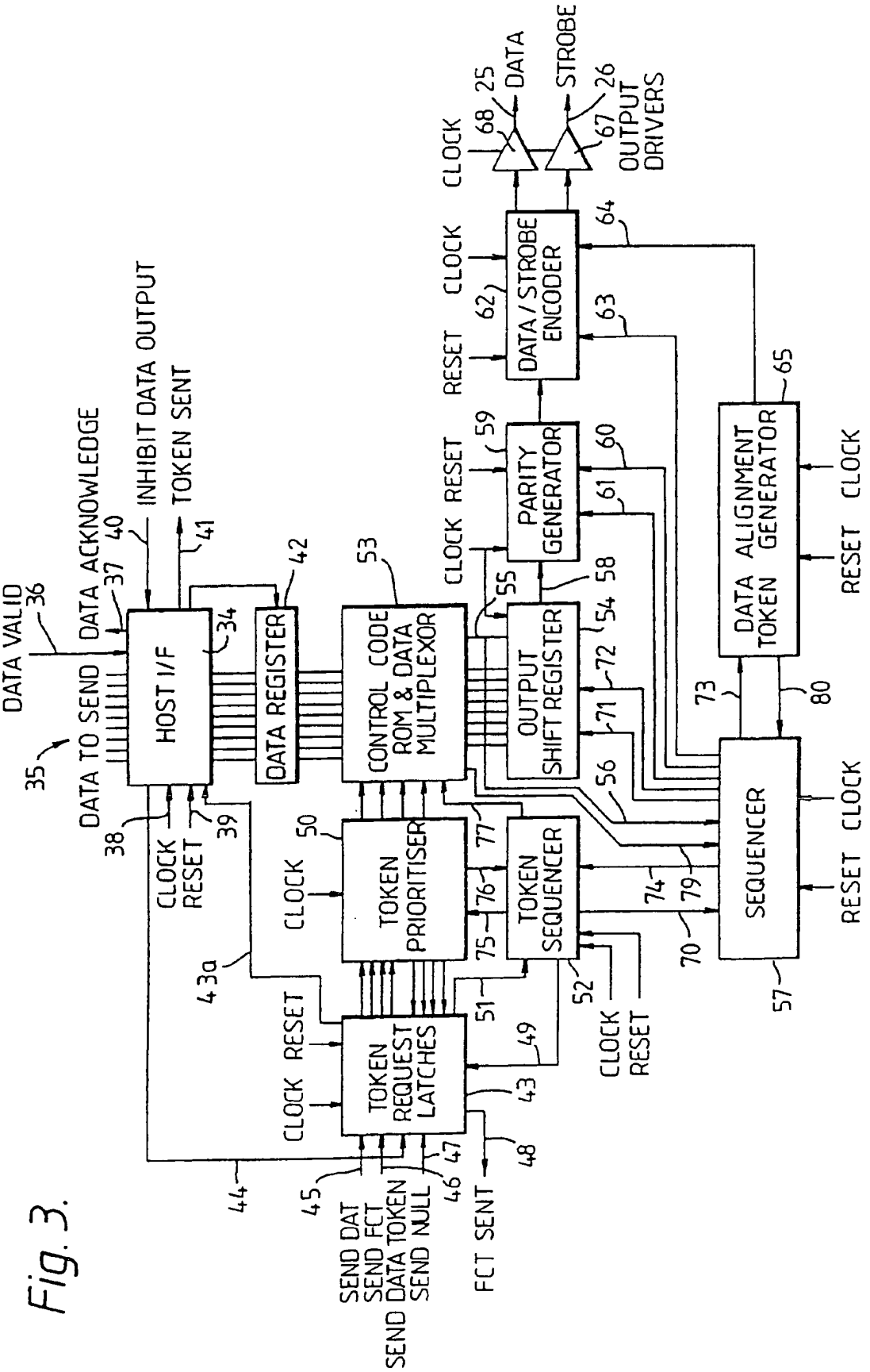

DATA VALID
DATA ACKNOWLEDGE
INHIBIT DATA OUTPUT
TOKEN SENT
DATA TO SEND
HOST I/F
DATA REGISTER
CONTROL CODE ROM & DATA MULTIPLEXOR
OUTPUT SHIFT REGISTER
PARITY GENERATOR
DATA/STROBE ENCODER
DATA
STROBE
OUTPUT DRIVERS
DATA ALIGNMENT TOKEN GENERATOR
SEQUENCER
TOKEN PRIORITISER
TOKEN SEQUENCER
TOKEN REQUEST LATCHES
CLOCK
RESET
SEND DAT
SEND FCT
SEND DATA TOKEN
SEND NULL
FCT SENT

Fig. 3.

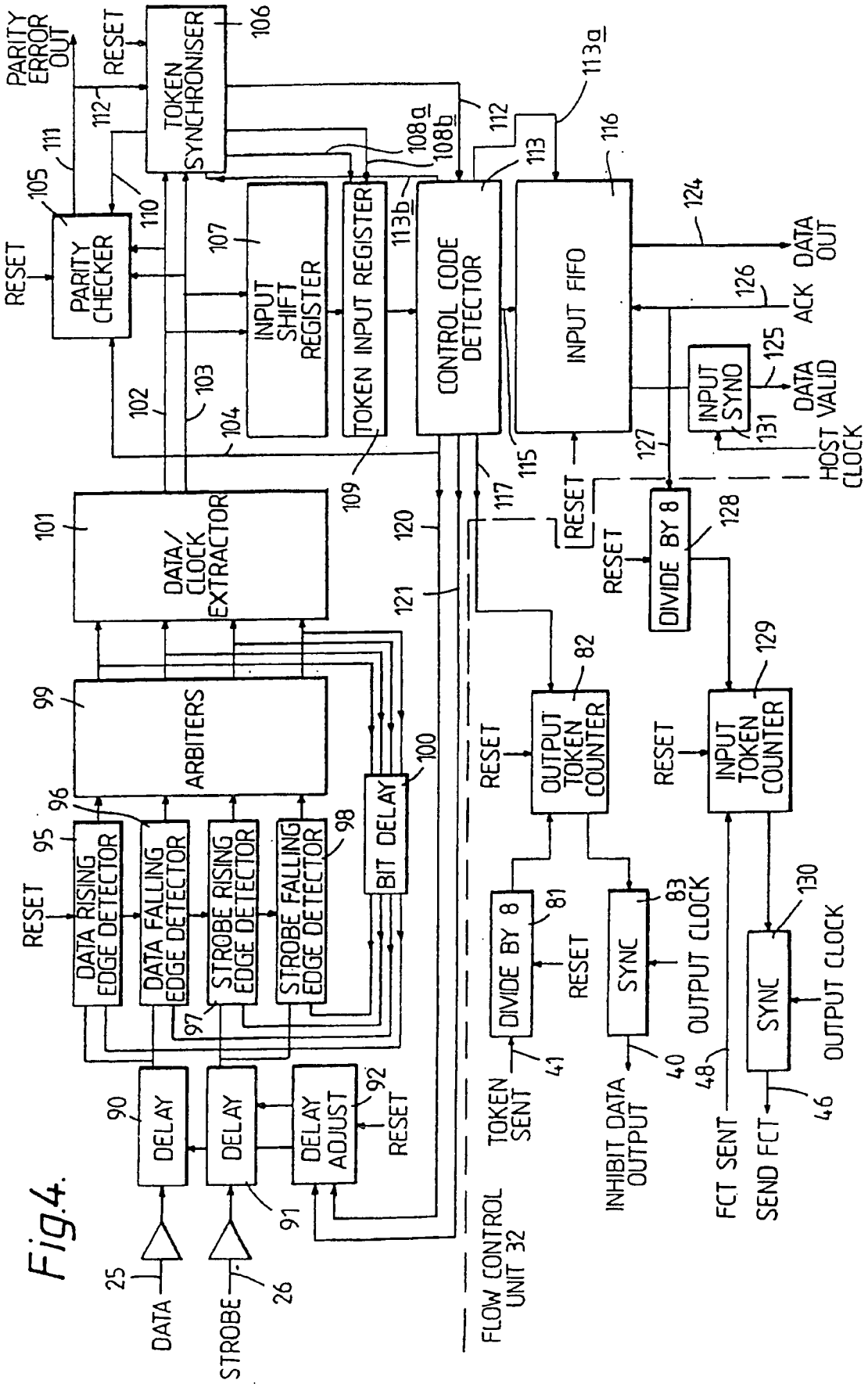
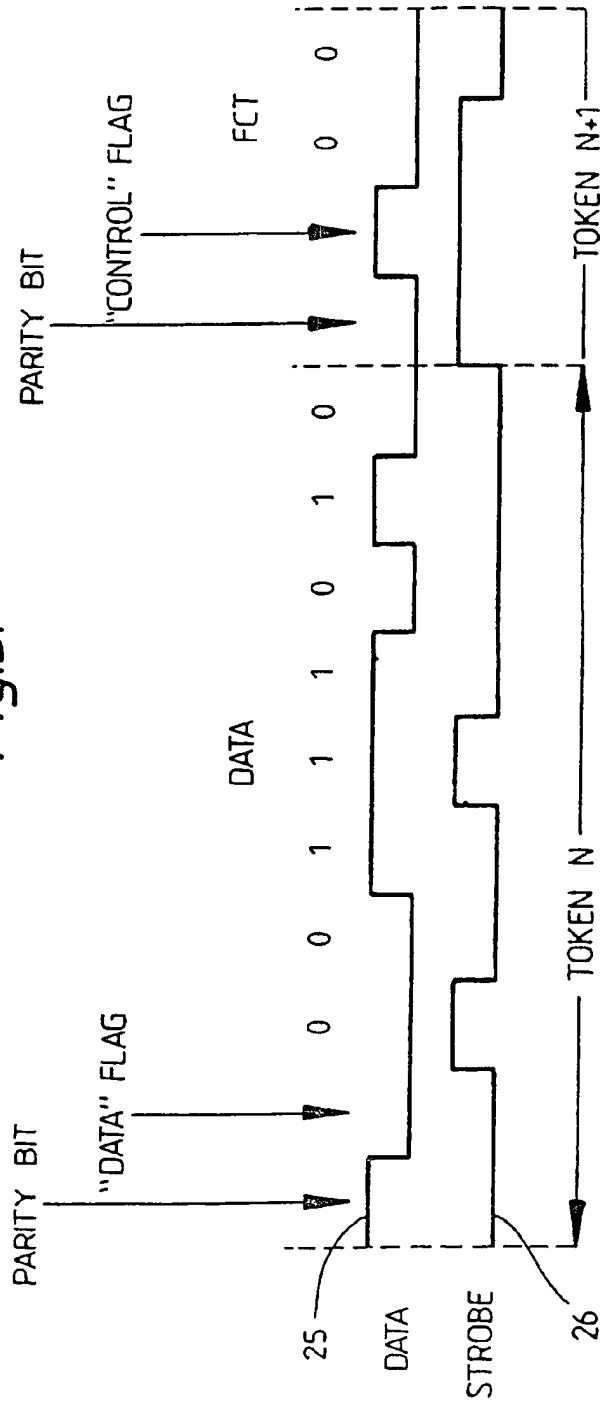

PARITY ERROR OUT
RESET
TOKEN SYNCHRONISER
PARITY CHECKER
INPUT SHIFT REGISTER
TOKEN INPUT REGISTER
CONTROL CODE DETECTOR
INPUT FIFO
DATA OUT
ACK
INPUT SYNO
DATA VALID
HOST CLOCK
DATA/ CLOCK EXTRACTOR
ARBITERS
DATA RISING EDGE DETECTOR
DATA FALLING EDGE DETECTOR
STROBE RISING EDGE DETECTOR
STROBE FALLING EDGE DETECTOR
BIT DELAY
DELAY
DELAY ADJUST
DIVIDE BY 8
OUTPUT TOKEN COUNTER
INPUT TOKEN COUNTER
SYNC
OUTPUT CLOCK
TOKEN SENT
INHIBIT DATA OUTPUT
FCT SENT
SEND FCT
DATA
STROBE
FLOW CONTROL UNIT 32

Fig.4.

Fig.5.


PARITY BIT
"DATA" FLAG
PARITY BIT
"CONTROL" FLAG
DATA
0 0 0 1 1 1 0 1 0 0
FCT
0 0
TOKEN N
TOKEN N+1
25
26
DATA
STROBE

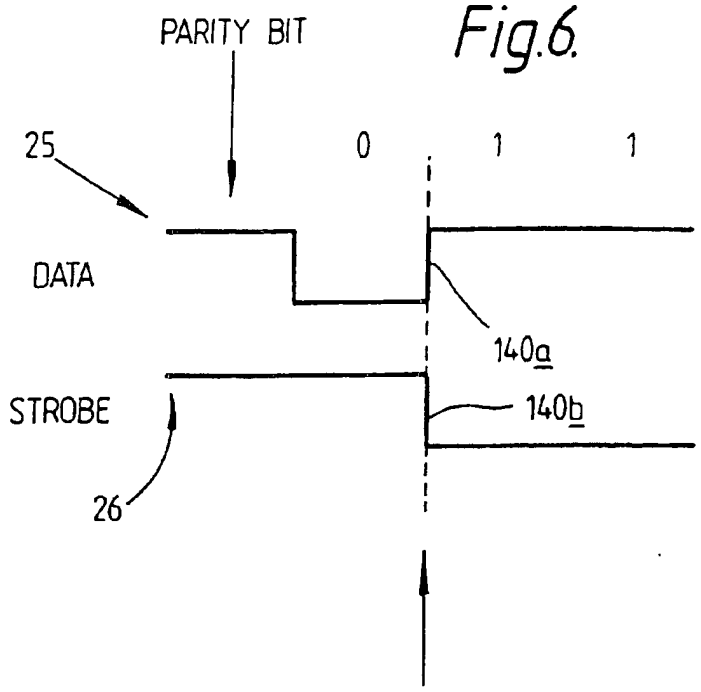

PARITY BIT
25
0
1
1
DATA
140a
STROBE
140b
26

Fig.6.

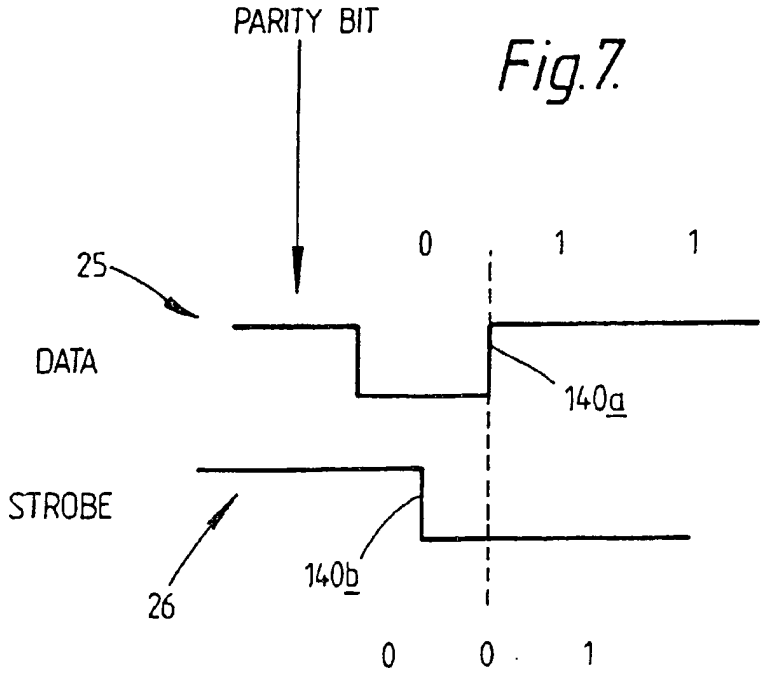

PARITY BIT
0
1
1
25
DATA
140a
STROBE
26
140b
0
0
1

Fig.7.

Fig.8.