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

(10) **DE 693 05 203 T3** 2006.08.24

(12) Übersetzung der geänderten europäischen Patentschrift

(97) **EP 0 560 226 B2**
(21) Deutsches Aktenzeichen: **693 05 203.1**
(96) Europäisches Aktenzeichen: **93 103 563.8**
(96) Europäischer Anmeldetag: **05.03.1993**
(97) Erstveröffentlichung durch das EPA: **15.09.1993**
(97) Veröffentlichungstag der Patenterteilung beim EPA: **09.10.1996**
(97) Veröffentlichungstag des geänderten Patents beim EPA: **11.01.2006**
(47) Veröffentlichungstag im Patentblatt: **24.08.2006**

(51) Int Cl.[8]: ***G05B 19/414*** (2006.01)
G05B 19/418 (2006.01)

(30) Unionspriorität:
847542 **06.03.1992** **US**

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

(73) Patentinhaber:
Pitney Bowes, Inc., Stamford, Conn., US

(74) Vertreter:
HOFFMANN & EITLE, 81925 München

(72) Erfinder:
Di Giulio, Peter C., Fairfield, Connect. 06430, US;
Lee, David K., Monroe, Connecticut 06468, US;
Riley, David W., Easton, Connecticut 06612, US;
Ryan, Frederick W., Jr., New Haven, Connecticut 06515, US

(54) Bezeichnung: **Flexibele Kommunikationsarchitektur für ein Bewegungssteuerungssystem**

Beschreibung

[0001] Diese Erfindung betrifft Bewegungssteuersysteme mit Serienkommunikation, einschließlich derjenigen, die auf Robotersysteme anwendbar sind, sowie Autosysteme oder Partikelbearbeitungssysteme, insbesondere Papierhandhabungssyteme. In Papierhandhabungssystemen werden Papierbögen in allgemein aufeinanderfolgender Weise über eine Reihe von motorgetriebenen Komponenten geführt, beispielsweise Bänder und Walzen. Oft bewegen sich die Bögen mit hoher Geschwindigkeit, und während der Überleitung von einem Band oder einer Walze zu der nächsten erfahren sie besonders hohe Beschleunigungen. Der Betrieb der Bewegungssteuerkomponenten ist besonders sorgfältig in Echtzeit zu koordinieren, damit gewährleistet ist, daß die Komponenten-zu-Komponenten-Übergänge reibungslos und ohne Stau verlaufen. Es ist oft auch günstig, die Positionen der Bögen sowie die Positionen der Motorwellen und deren Geschwindigkeiten mit Sensoren zu überwachen. Üblicherweise erfolgt die Steuerung für die Motorsteuerung, die Sensorsteuerung und der allgemeinen Systemfunktionen mit zumindest einem Prozessor. Steuersignale müssen zu mechanischen Antriebsmaschinen abgegeben werden (Motor, Solenoide, Wandler, usw.), und Sensordaten werden durch die Sensoren erfaßt. Ist zudem mehr als ein Sensor in dem System vorgesehen, so ist die Prozessor-zu-Prozessorkommunikation zu unterstützen. Zusätzlich ist es in modularen Papierhandhabungssytemen erforderlich, eine Vorrichtung für die Rekonfiguration des Systems und zum Hinzufügen neuer Module vorzusehen.

[0002] In US-A-4 835 699 ist ein Kommunikationssystem mit Merkmalen gemäß denjenigen des Oberbegriffs des Patentanspruchs 1 beschrieben. In Elektronik Bd. 40, 29. Oktober 1991, Seiten 59-62, 65, 65, 68, 69 ist eine Netzwertkommunikation zwischen Knoten beschrieben, unter Einsatz von "Leitwegelementen", "Brücken", und "Gateway"-Elementen.

[0003] Relevanter Stand der Technik ist auch in dem technischen Artikel "Time Critical Communication Networks: Field Buses", IEEE Network, 2 (1988), Nr. 3, New York, NY, US, Seiten 55-63., von P. Pleinevaux et. al. offenbart.

[0004] In EP-A-0 200 365 ist ein Steuersystem mit verteilten programmierbaren Kontrollern beschrieben. Ein "aktiver Monitor" ruft zum Bereitstellen einer Liste weitere Knoten in dem Netzwerk ab und ordnet jedem Knoten zum Senden für dessen Daten einen Zeitschlitz zu.

[0005] Demnach wäre es wünschenswert, eine Kommunikationsarchitektur für ein Bewegungssteuersystem zu schaffen, das ein Konfiguration des Systems derart ermöglicht, daß (a) ein zentralisierter Steuerknoten und periphere Randknoten vorgesehen sind, sowie (b) ein zentralisierter Steuerknoten und verteilte Steuerknoten oder (c) ein zentralisierter Steuerknoten, Randsteuerknoten und verteilte Steuerknoten.

[0006] Es wäre auch wünschenswert, eine Kommunikationsarchitektur zu ermöglichen, die das einfache Miteinbeziehen zusätzlicher Module in das System ermöglicht.

[0007] Es wäre auch wünschenswert, den Umfang der Verdrahtung zum Verbinden der zahlreichen Elemente in den Bewegungssteuersystemen zu reduzieren.

[0008] Es wäre auch wünschenswert, eine Kommunikationsarchitektur für ein Bewegungssteuersystem zu ermöglichen, die eine sorgfältige Isolation der Verdrahtung gegenüber einer elektromagnetischen und einer Hochfrequenz-Interferenz ermöglicht.

[0009] Es wäre auch wünschenswert, eine Kommunikationsarchitektur für ein Bewegungssteuersystem zu ermöglichen, das eine Zunahme der Zahl der benützten gemeinsamen Teile ermöglicht, sowohl innerhalb einer Produktlinie als auch von Produktlinie zu Produktlinie, um hierdurch die Systemkosten zu reduzieren.

[0010] Es wäre auch wünschenswert, eine Kommunikationsarchitektur und ein Bewegungssteuersystem zu schaffen, das eine einfache Rekonfiguration für das System ermöglicht, für eine schnellere Produktentwicklung.

[0011] Es wäre auch wünschenswert, eine Kommunikationsarchitektur für ein Bewegungssteuersystem zu schaffen, bei dem Servicedienste auf Anwendungsebene während des Arbeitsbetriebs angeboten werden, einschließlich Herstellungs- und Servicediagnosefunktionen, und bei dem nach einer Systemrekonfiguration ein Netzwerk automatisch konfiguriert wird.

[0012] Es wäre auch wünschenswert, eine Architektur zu schaffen, die Bewegungssteuerungs-Systemknoten

ermöglicht, bei denen eine Boundary-Scan-Technik einsetzbar ist und bei der Boundary-Scan-Information von oder zu einem einzigen Netzwerkverbindungspunkt übertragbar ist, damit ein Herstellungstest und eine Servicediagnose möglich ist.

[0013] Es wäre auch wünschenswert, eine Kommunikationsarchitektur für ein Bewegungssteuersystem zu schaffen, bei dem die Angleichung der Kommunikations-Baudrate zum Erzielen einer optimalen Abwägung zwischen Kosten/Leisten ermöglicht wird.

[0014] Eine Aufgabe der vorliegenden Erfindung besteht in der Schaffung einer Kommunikationsarchitektur für ein Bewegungssteuersystem, das die Konfiguration des Systems mit (a) einem zentralisierten Knoten und peripheren Steuerknoten, (b) einen zentralisierten Steuerknoten und verteilten Steuerknoten oder (c) einem zentralisierten Steuerknoten und peripheren Steuerknoten und verteilten Steuerknoten ermöglicht.

[0015] Gemäß der Erfindung wird ein Steuersystem mit serieller Kommunikation gemäß dem Patentanspruch 1 geschaffen.

[0016] Die obigen und weitere Aufgaben und Vorteile dieser Erfindung ergeben sich bei Betrachtung der folgenden detaillierten Beschreibung im Zusammenhang mit der beiliegenden Zeichnung, in der gleiche Bezugszeichen gleiche Teile kennzeichnen; es zeigen:

[0017] Fig. 1 ein Blockschaltbild der Elemente eines Systems, die gemäß einer bevorzugten Ausführungsform der Architektur der vorliegenden Erfindung verbunden sind;

[0018] Fig. 2 ein schematisches Diagramm zum Darstellen der Unterstützungsgebiete der Architektur gemäß der Erfindung;

[0019] Fig. 3 ein Blockschaltbild einer zentral gesteuerten Konfiguration gemäß der Erfindung;

[0020] **Fig.** 4 ein Blockschaltbild der verteilten Steuerkonfigurationen gemäß der Erfindung;

[0021] **Fig.** 5 ein Blockschaltbild der kombinierten Steuerkonfiguration gemäß der Erfindung;

[0022] Fig. 6 ein Blockschaltbild einer erweiterten Konfiguration gemäß der Erfindung;

[0023] Fig. 7 ein Diagramm eines Beispiels einer bevorzugten Ausführungsform für ein bei der vorliegenden Erfindung benütztes Codierschema;

[0024] Fig. 8 ein schematisches Blockschaltbild zum Darstellen dritter Leitungsverbindungen gemäß der Erfindung;

[0025] Fig. 9 ein Diagramm eines gewählten Synchron-Datenübertragungssteuerungs-(SDLC, Synchronous Data Link Control)-Rahmens gemäß der Erfindung;

[0026] Fig. 10 ein Diagramm gemäß einer Knotenidentifikation (ID, Node Identification) und einer Knotenadresse gemäß der Erfindung;

[0027] Fig. 11 zum Darstellen der Zugriffsteuerungen gemäß der Erfindung;

[0028] Fig. 12 ein Diagramm zum Darstellen der Zugriffsteuerung für einen verteilten Steuerknoten (DCN, distributed control node) bei Betrieb gemäß der Erfindung;

[0029] Fig. 13 ein Diagramm zum Darstellen der DCN-Zugriffsssteuerung während des Normalbetriebs gemäß der Erfindung;

[0030] Fig. 14 ein Diagramm zum Darstellen, wie gemäß der Erfindung ein DCN-Knoten einen Schnittzeitpunkt berechnet, um eine Störung zu vermeiden;

[0031] Fig. 15 ein Diagramm zum weiteren Darstellen, wie gemäß der Erfindung ein DCN-Knoten einen Konkurrenzbetrieb vermeidet;

[0032] Fig. 16 ein Diagramm zum Darstellen einer zyklischen Blockprüfung (cyclical redundancy check; CRC) gemäß der Erfindung für 16 Bit;

[0033] Fig. 17 ein Diagramm zum Darstellen eines Steuerrahmens eines zentralisierten Steuerknotens (centralized control note) CCN gemäß der Erfindung;

[0034] Fig. 18 ein Diagramm eines Datenrahmens für die Übertragung zwischen dem CCN-Knoten zu einem PCN-Knoten (peripheral control node) oder umgekehrt gemäß der Erfindung;

[0035] Fig. 19 ein Diagramm eines Datenrahmens für die Übertragung von einem DCN-(CCN)-Knoten zu einem DCN-Knoten gemäß der Erfindung;

[0036] Fig. 20 ein Diagramm eines Netzwerkaustauschrahmens zum Festlegen der Baudrate gemäß der Erfindung;

[0037] Fig. 21A und Fig. 21B Diagramme der CNN-Austauschrahmen gemäß der Erfindung;

[0038] Fig. 22 ein Diagramm eines Netzwerkkonfigurations-Austauschrahmens gemäß der Erfindung;

[0039] Fig. 23A und Fig. 23B Diagramme für eine Statusabfrage durch den CNN-Knoten gemäß der Erfindung;

[0040] Fig. 24A und Fig. 24B Diagramme für Schreibvorgänge zu einem Steuer/Konfigurationsregister gemäß der Erfindung;

[0041] Fig. 25 ein Diagramm eines Austauschrahmens für die Zugriffsprioritätszuordnung/Löschung in einem DCN-Knoten gemäß der Erfindung;

[0042] Fig. 26 einen Synchronisationssenderahmen (SYNC) gemäß der Erfindung;

[0043] Fig. 27 ein Diagramm eines Boundary-Scan-Austauschrahmens gemäß der Erfindung;

[0044] Fig. 28 ein Diagramm eines Eintaktrahmens zum Verbreiten von Boundary-Scan-Daten gemäß der Erfindung;

[0045] Fig. 29A und Fig. 29B Diagramme zum Abfragen von DCN-Knoten im Hinblick auf ein Boundary-Logikergebnis gemäß der Erfindung;

[0046] Fig. 30 ein Diagramm für einen Datenaustauschrahmen zwischen dem DCN-Knoten und einem PCN-Knoten gemäß der Erfindung;

[0047] Fig. 31 ein Diagramm mit einem Datenaustauschrahmen für die Übertragung von dem DCN-Knoten zu einem DCN-Knoten gemäß der Erfindung;

[0048] Fig. 32 ein Diagramm eines Datenrahmens für die Übertragung von einem DCN-Knoten zu einem anderen DCN-Knoten gemäß der Erfindung;

[0049] Fig. 33 ein Diagramm zum Darstellen von Netzwerk-Zustandsübergängen gemäß der Erfindung;

[0050] Fig. 34 ein Diagramm der DCN-zu-DCN-Übergangs-Zustandsmaschine gemäß der Erfindung;

[0051] Fig. 35 ein Diagramm eines Funktionsstatusbytes gemäß der Erfindung;

[0052] Fig. 36 ein Diagramm eines Datenstatusbytes gemäß der Erfindung;

[0053] Fig. 37 ein Diagramm gemäß einer Mitteilungslänge gemäß der Erfindung; und

[0054] Fig. 38 ein Diagramm einer Zugriffszustandsmaschine gemäß der Erfindung.

Detailierte Beschreibung der Erfindung

[0055] In dem Bewegungssteuerungsumfeld ist es nicht nur wichtig, eine Echtzeit-Bewegungssteuerung zu unterstützen, sondern auch eine Kommunikation zwischen anderen Elementen des Systems zu unterstützen. Diese Elemente können Eigenschaften aufweisen, die zu einem unterschiedlichen Satz von Kommunikationsanforderungen führen, die sich von denjenigen der im Zusammenhang mit der Bewegung vorgesehenen Elementen unterscheidet. Beispielsweise können Systemkonfigurationsparameter periodisch durch einen Betreiber eingegeben werden. Diese Information wird den Komponenten des Systems asynchron zugeführt. Ferner kann für die intelligenten Systemkomponenten auch zusätzlich die Durchführung asynchroner Kommunikationsvorgänge von einem gleichrangigen Element zu einem anderen gleichrangigen Element erforderlich sein. Im Gegensatz zu diesem ereignisgetriebenen Modus der Kommunikation sind hochleistungsfähige festgekoppelte Bewegungssteuerelemente üblicherweise in Echtzeit über einen Kanal mit hoher Bandbreite zu treiben.

[0056] Die vorliegende Erfindung schafft eine Kommunikationsarchitektur, die hinreichend flexibel ist, um sowohl eine synchrone als auch eine asynchrone Kommunikation in Bewegungssteuerungssystemen zu ermöglichen. Die Architektur setzt weiterhin einen seriellen Bus ein, der eine Reduzierung des Verdrahtungsaufwands für die Verbindung des Systems ermöglicht, insbesondere wenn das physikalische. Layout eine lineare Anordnung der Komponenten aufweist. Der Aufbau im seriellen Bus ist auch leichter abzuschirmen als andere Aufbauten und ermöglicht eine Zunahme der Zahl der gemeinsamen Teile in dem System, was zu einer Reduktion der Kosten pro Einheit und demnach der gesamten Systemkosten führt. Durch die Architektur der vorliegenden Erfindung wird eine Vielzahl von Motorsteuersystem-Aufbauten unterstützt, was bei den diese Architektur einsetzenden Systemen eine einfache Modifikation und Erweiterung und eine einfache Entwicklung neuer Systeme ermöglicht.

[0057] In Fig. 1 ist ein schematisches Schaltbild einer Systemimplementierung der Motorsteuer-Kommunikationsarchitektur **10** der vorliegenden Erfindung gezeigt. Das speziell gezeigte System ist ein Postversandsystem, das einen System-Controller **11** und eine Tastatur und eine Anzeige **12** für die Schnittstelle zu dem Betreiber aufweist. Das System enthält auch einen Bewegungs-Controller **15** zum Treiben von Solenoiden **20**, zum Treiben von smarten Motoren **21** und zum Empfangen von Daten von der Sensorgesamtheit **22**, jeweils über den seriellen Bus **230** und die smarte I/O-Anschlußkarte **19**. Gelegentlich ist es im Hinblick auf Plazierungsaspekte wirtschaftlich, anstelle des Aufbaus mit der Anschlußkarte **19** eine Sensorgesamtheit **23** mit eingebauter Kommunikationsschaltung und einen smarten Motor **24** mit eingebauter Kommunikationsschaltung einzusetzen. Der System-Controller **11** und der Bewegungs-Controller **15** kommunizieren asynchron über den Bus **230**. Weitere Module wie ein Meßgerät **16**, ein Datiergerät **17**, eine In-line-Waage **18**, ein Druckeranschluß **13** und ein zugeordnetes I/O-Element **14** kommunizieren über den Bus **230**. In dem System können auch Optionalmodule **26** vorgesehen sein, damit beispielsweise ein Modem für die externe Kommunikation bereitgestellt wird, oder ein Floppy-Laufwerk. Die Komponenten des Versandsystems werden durch die Stromversorgung **25** versorgt, die wie gezeigt, mit allen Knoten verbunden ist.

[0058] Die Bewegungssteuerungsarchitektur **10** unterstützt drei Arten von Komponenten oder Knoten. Wie in den Fig. 1 – Fig. 6 gezeigt ist, bestehen die unterstützten Knotentypen aus einem zentralisierten Steuerknoten (CNN-Knoten) **210**, einem verteilten Steuerknoten **310** und peripheren Steuerknoten **220**, die wie gezeigt, konfiguriert werden können.

[0059] Die unterschiedlichen Knotentypen sind sowohl durch den Umfang ihrer elektronischen Leistungsfähigkeit als auch durch ihre Rolle in der gesamten Systemarchitektur gekennzeichnet. Ein zentralisierter Steuerknoten **210** weist ein signifikantes lokales Verarbeitungsvermögen auf, und zwar in der Form einer Zentralverarbeitungseinheit (CPU). Dieser Knoten vermittelt Kommunikationsvorgänge zwischen den weniger leistungsstarken verteilten Steuerknoten **310**, die typischerweise ebenfalls CPU-basiert sind. Der zentralisierte Steuerknoten **210** ist auch der Knoten, der direkt periphere Steuerknoten **220** treiben kann, die die am wenigsten leistungsfähigen Arten von Steuerknoten darstellen.

[0060] Die verteilten Steuerknoten **310** sind Knoten, die typischerweise einen lokalen Prozessor enthalten. Jedoch können die verteilten Steuerknoten **310** nicht Knoten-zu-Knoten-Kommunikationsvorgänge wie der zentralisierte Steuerknoten **210** vermitteln.

[0061] Der am wenigsten entwickelte Steuerknoten ist der periphere Steuerknoten **220**. Periphere Steuerknoten **220** enthalten elektronische Schaltungen vom mittleren Niveau zum Bilden einer Schnittstelle mit Hardware-Komponenten, beispielsweise Motoren, jedoch sind sie nicht CPU-basiert. Periphere Steuerknoten können durch den zentralisierten Steuerknoten **210** getrieben werden, können jedoch nicht das Netzwerk steuern

oder Kommunikationsvorgänge initiieren

[0062] Die zahlreichen in Fig. 1 gezeigten Systemkomponenten können durch ihre Art des Knotentyps gekennzeichnet werden. Der Bewegungs-Controller **15** nimmt typischerweise die Rolle eines zentralisierten Steuerknotens **200** ein, während der Aufbau der Anschlußkarte **19** mit dem Solenoid **20**, dem smarten Motor **21** und der Sensorgesamtheit **22** einen der peripheren Steuerknoten **220** bildet. Die Sensorgesamtheit **23** mit eingebauter Kommunikationsschaltung und der smarte Motor **24** mit eingebauter Kommunikationsschaltung stellten ebenfalls periphere Randknoten **220** dar, genauso wie die Tastatur und die Anzeige **27**. Die verbleibenden in Fig. 1 gezeigten Module, beispielsweise das Frankiergerät **16** und das Datiergerät **17**, sind typischerweise verteilte Steuerknoten **310**, obgleich alternative Anordnungen möglich sind. Beispielsweise könnte der System-Controller **11** der CCN-Knoten **210** sein, aber der Druckeranschluß **13** könnte als einer der PCN-Knoten **220** anstelle des DCN-Knotens **310** konfiguriert sein. Es ist möglich, einen zugeordneten I/O-Anschluß **14** einzusetzen, der einer der PCN-Knoten **220** ist, und zwar als kostengünstige Vorrichtung zum Bereitstellen einer Schnittstelle zwischen dem zentalisierten Steuerknoten **210** und einer Zentralverarbeitungseinheit (CPU) über den seriellen Bus **230**.

[0063] Wie in Fig. 2 gezeigt ist, die den Unterstützungsbereich der Architektur **10** auf einem allgemeineren Niveau als in Fig. 1 gezeigt darstellt, kann die Funktion der ereignisgetriebenen Steuerung **2001** durch Einsatz entweder eins PCN-Knotens **220** oder eines DCN-Knotens **310** erreicht werden, während die Funktion der festgekoppelten Steuerung **2002** typischerweise durch Einsatz eines der PCN-Knoten **220** erzielt wird. Die Funktion der Sensoreingabe **2004** wird üblicherweise durch Einsatz von PCN-Knoten **220** erreicht, obgleich bei einigen Funktionen DCN-Knoten **310** eingesetzt werden können. Sonstige Eingabe/Ausgabefunktionen **2005** werden allgemein durch Einsatz von DCN-Knoten **310** oder von PCN-Knoten **220** erzielt. Die Systemmodularität **2007** läßt sich durch Einsatz sowohl von PCN-Knoten **220** als auch von DCN-Knoten **310** erzielen. Die allgemeine Funktion der hochperformanten koordinierten Bewegungssteuerung **2003** wird typischerweise durch Einsatz eines der PCN-Knoten **220** erreicht. Die durch die geringere Performanz gekennzeichneten Funktionen oder die unabhängige Bewegungssteuerung **2004** kann durch die PCN-Knoten **220** oder die DCN-Knoten **310** erzielt werden, während CPU-zu-CPU-Datenübertragungen mit hoher Rate **2009** vorzugsweise durch Einsatz von DCN-Knoten **310** erzielt werden.

[0064] In allen Betriebsmoden führt der CCN-Knoten **210** die Netzwerksteuerfunktion durch. In einem Betriebsmodus, der zentralisierten Steuerung, steuert ein CPU-basierter Netzwerkmaster-DCN-Knoten **210** mehrere PCN-Knoten **220**, die nicht CPU-basiert sind, sondern im Gegensatz eine smarte Eingabe/Ausgabe-(I/O)-Logik-Verarbeitungshardware und eine serielle Kommunikationslogik enthalten. Typische I/O-Verarbeitungsfunktionen sind Pulsbreitenmodulation, Quadraturdekodierung, Sensorabtastung, usw.. Die PCN-Knoten **220** bilden eine Schnittstelle auf mittlerem Niveau zu den Bewegungssteuerelementen, beispielsweise den Motoren, den Solenoiden und den Sensoren in dem System.

[0065] Bei der verteilten Steuerung können die CPU-basierten DCN-Knoten **310** Zuordnungsaufgaben unabhängig von dem CCN-Knoten **210** durchführen, während der CCN-Knoten **210** eine Kommunikation zwischen DCN-Knoten **310** ermöglicht.

[0066] Ein hybrides System mit sowohl DCN-Knoten **310** und PCN-Knoten **220** wird ebenfalls unterstützt, beispielsweise in einer Anordnung, die eine Systemerweiterung über eine Verzweigung ermöglicht.

[0067] Wie schematisch in Fig. 3 gezeigt ist, wird ein hochperformantes Bewegungssteuerungs-Umfeld mit vielen PCN-Knoten **220** (smarter I/O-Servomotor und smarte Sensorknoten) unter zentralisierter Steuerung unterstützt. Der DCN-Knoten **210** sendet Bewegungssteuerdaten an Servomotoren und Solenoide und empfängt jeweils einen Sensorstatus und Motorcodier-Wellenpositionsdaten von den Motoren und den Sensoren. Die Kommunikation dieser Daten enthält eine Fehlerdetektorlogik. Da die Steuer/Statustdaten zu/von diesen Knoten mit hohen Raten regeneriert wird, ist eine erneute Datenübertragung bei Erfassung von Fehlern nicht erforderlich. Bis zu 31 der PCN-Knoten **220** können an dem seriellen Mehrfachabzweig-Kommunikationsbus **230** angeschlossen sein. Jeder der PCN-Knoten **220** ist vollständig mit Hardwarelogik ohne eine CPU implementiert. Die PCN-Knoten **220** empfangen Meldungen lediglich von dem CCN-Knoten **210**. Der PCN-Knoten **220** dekodiert die Meldung, führt die Zuordnungsfunktionen der Meldung durch, gewinnt eine Rückmeldung und überträgt sie zu dem CCN-Knoten **210**. Allgemein dienen die Befehle des CCN-Knotens **210** der Steuerung der Ausgangsleitungen der PCN-Knoten **220**, wohingehend die Antworten der PCN-Knoten **220** zum Anzeigen des Status der Eingabeleitungen der PCN-Knoten **220** dienen. Die Anwendungshardwarelogik der PCN-Knoten **220** ermöglicht eine intelligente Verarbeitung der Ausgabebefehle des DCN-Knotens **210** und der ungefähren Eingabedaten der PCN-Knoten **220**, damit der CCN-Knoten **210** nicht die I/O-Verarbeitung auf

niedrigem Niveau durchführen muß. Zum Vereinfachen der Kommunikation mit den PCN-Knoten **220** ist es erforderlich, daß unmittelbar nach dem Senden einer Meldung durch den CCN-Knoten **210** an die PCN-Knoten **220** dieser eine unmittelbare Antwortmeldung anfordert. Diese Antwortmeldung wird durch den CCN-Knoten **210** auch als implizite Bestätigung kleiner Meldung benützt. Eine Kommunikation von einem PCN-Knoten **220** zu einem PCN-Knoten **220** wird nicht zugelassen.

[0068] Der Modus mit verteilter Steuerung unterstützt die Kommunikation zwischen intelligenten Knoten in dem System. In diesem Modus erfolgt die Steuerung und die Übertragung von Daten zum Unterstützen der folgenden Vorgänge: Prozessor-zu-Prozessor-Kommunikation, Motorbefehle, Motor/Sensor-Status, initialisierte Befehle, Diagnosemitteilungen und Fehlerstatus. Diese steuerungsorientierte Datenübertragung basiert vollständig auf einer Fehlerdetektion, einer erneuten Übertragung und einem duplizierten Meldungsschutz sowohl auf dem Niveau der Datenpakete als auch der Meldungen mit mehreren Paketen.

[0069] Der Aufbau des Netzwerks unter Einsatz lediglich der DCN-Knoten **310** ist in **Fig.** 4 gezeigt. Bis zu 31 DCN-Knoten **310** können an dem Bus angeschlossen sein. Jeder DCN-Knoten **310** kann mit jedem anderen der DCN-Knoten **310** kommunizieren, einschließlich dem DCN-Knoten **210**, auf einem Niveau einer Kommunikation zwischen gleichrangigen Elementen. Demnach kann ein DCN-Knoten **310** eine logische Datenverbindung zu einem anderen DCN-Knoten **310** herstellen. Die Rolle des Netzwerkmaster-DCN-Knoten **210** in diesem Aufbau besteht in der Bereitstellung von Netzwerkdiensten, beispielsweise dem Sammeln und Verteilen von Information für die Zugriffssteuerung zu dem Bus **320** bei Stromversorgung, Überwachung des Netzwerks und Durchführen der Netzwerkdiagnose. Der DCN-Knoten **210** kann auch die Funktion eines der DCN-Knoten **310** durchführen.

[0070] Es ist auch möglich, sowohl PCN-Knoten **220** als auch DCN-Knoten **310** in einem einzigen Netzwerk zu kombinieren, wie in **Fig.** 5 gezeigt ist. Bis zu 31 gemischte Knoten lassen sich an dem Bus **330** anschließen. Der DCN-Knoten **210** steuert das Zugriffsrecht eines Knoten auf den Bus **230** in fester Weise. Der DCN-Knoten **210** ordnet einem der PCN-Knoten **220** Priorität gegenüber den DCN-Knoten **310** zu, da die PCN-Knoten **220** allgemein eine synchronisierte Kommunikation mit hohen Regenerierdaten erfordern, wohingehend der kritische Punkt für die Kommunikation des DCN-Knotens **310** in der Meldungswartezeit und dem publizierten Meldungsschutz besteht. Nachdem alle Kommunikationsanforderungen für die PCN-Knoten **220** erfüllt sind, berechnet der DCN-Knoten **210** den Umfang der für einen DCN-Knoten **310** zu einem anderen DCN-Knoten **310** zur Verfügung stehenden Zeit, und er kann den Befehl abgeben, der es einem der DCN-Knoten **310** ermöglicht, die Übertragung zu initiieren. Es wird davon ausgegangen, daß die Bandbreite des Kommunikationskanals groß genug ist, um PCN-Knoten **220** und DCN-Knoten **310** handzuhaben. Bei diesem Aufbau wird nicht zugelassen, daß DCN-Knoten **310** direkt PCN-Knoten **220** ansprechen, und auch eine Kommunikation von einem PCN-Knoten **220** zu einem anderen PCN-Knoten **220** wird nicht zugelassen. Ist das Ansprechen von PCN-Knoten **220** durch einen der DCN-Knoten **310** erforderlich, so überträgt der DCN-Knoten **210** die Meldungen.

[0071] Zusätzlich kann das System, wie in Fig. 6 gezeigt ist, erweitert werden. Ein Erweiterungsknoten **510** ist so aufgebaut, daß er sowohl die Funktionen des DCN-Knotens **310** als auch des DCN-Knotens **210** durchführt. Der Erweiterungsknoten **510** kann an einem Netzwerk in derselben Weise angeschlossen sein, wie es für einen der DCN-Knoten **310** der Fall wäre, der PCN-Knoten **220** in einem Unternetzwerk steuert, genauso wie jetzt für den CCN-Knoten **210** der Fall wäre.

[0072] Bevorzugte Spezifikationen für Systeme, die die Bewegungssteuerungs-Kommunikationsarchitektur **10** einsetzen, sind in der Tabelle 1 zusammengestellt.

TABELLE 1

PHYSIKALISCHE SCHICHT:	verdrilltes Drahtpaar RS485 (kann ein Abschirmen erfordern)
DISTANZ:	maximal 75 Fuß
MAXIMALZAHL DER KNOTEN:	32 (1 CCN-Knoten mit irgendeiner Kommunikation der DCN-Knoten und der PCN-Knoten)
DATENRATE:	bei der Netzwerkinitialisierung wählbar (10 Mbps, 5 Mbps, 2,5 Mbps, 1,26 Mpbs, 675 Kbps)
RAHMENTECHNIK:	SDLC (International Business Machines, Inc., Synchronous Data Link Control, Allgemeine Information, IBM Formblatt, GA27-3093)
UNTERSTÜTZTE KOMMUNIKATION	
VERTEILT:	CPU-zu-CPU
ZENTRALISIERT:	Master/Slave von Zentrale CPU zu smarten I/O-Knoten
ZUGRIFFSSTEUERUNG:	
VERTEILT:	Zeitschlitzzugriff zwischen CPU-Knoten
ZENTRALISIERT:	zeitsynchron auf Grundlage eines Ansprechens
FEHLERERFASSUNG:	SCLC-16-bit-CRC mit unmittelbaren Bestätigen
FEHLERBEHEBUNG:	
VERTEILT:	Bestätigung mit erneutem Versuch und Duplizierungsschutz
ZENTRALISIERT:	hohe Regenerierrate für smarte I/O-Knoten
DATENLÄNGE	
VERTEILT:	bis zu 33 Bytes Daten, die einem Anwender zugeordnet sind
ZENTRALISIERT:	bis zu 8 Bytes Daten
LEISTUNG BEI 10 Mbps:	
VERTEILT:	maximale Wartezeit für Meldung < 5 ms unter der Annahme, daß zumindest 30% der Bandbreite den DCN-Knoten zugeordnet ist
ZENTRALISIERT:	Wiederholungen mit durchschnittlich 1 ms bei bis zu 31 PCN-Knoten
UNTERSTÜTZUNGSANFORDERUNGEN AUF ANWENDUNGSNIVEAU	
AUTOMATISCHE SYSTEMKONFIGURATION	
MODULARITÄT:	Option zum Hinzufügen identischer Module mit geringer/keinen Anpassungen
HERSTELLUNGS/SERVICEDIAGNOSE	
PROGRAMMLADEFÄHIGKEIT	

[0073] Bewegungssteuersysteme, die die Kommunikationsarchitektur **10** einsetzen, können viele Module enthalten, die als Hauptuntereinheiten identifiziert werden. Im Hinblick auf die Kommunikation kann ein Modul

jedoch als Kommunikationsknoten betrachtet werden. Sind Module durch Kabel verbunden, so kann es erforderlich sein, daß der Netzwerk-Controller (CCN-Knoten **210**) die physikalische Reihenfolge der Modulverbindungen bestimmt. Identische Module können in einem System in Serie verbunden sein. Demnach kann – wie im folgenden gezeigt – ein zusätzlicher Draht in dem Kommunikationskabel zum Erfüllen dieser Anforderung enthalten sein, zusätzlich zu den Kommunikationssignalleitungen.

[0074] In allen Systemkonfigurationen teilen die Knoten vorzugsweise einen bidirektionalen RS485-Mehrfachanschlußbus **230**, obgleich andere Hardwareimplementierungen auch einsetzbar sind, beispielsweise optische Grasfaserkabel, IEEE-802.3 10-Basis-T-Busse oder Busse mit offenem Kollektor. Der Bus **230** kann bis zu 35 Fuß von einem Ende bis zum anderen Ende lang sein und mit bis zu 10 Mbps betrieben werden. Größere Längen können bei entweder geringeren Geschwindigkeiten (< 10 Mbps) oder bei einer Ringtopologie mit anderen Kommunikationsmedien erzielt werden. Bis zu 32 Knoten - ein zwingend erforderlicher CCN-Knoten **210** und eine Kombination von bis zu 31 PCN-Knoten **220** und DCN-Knoten **210** – können mit einem Bus **230** verbunden sein.

[0075] Bei einer Kommunikation lediglich zwischen PCN-Knoten **220** ist die Bandbreitenkapazität des Systems groß genug, um die Abtastrate von 1000 Abtastungen pro Sekunde pro PCN-Knoten **220** für bis zu 31 PCN-Knoten **220** handzuhaben, um hochleistungsfähige Motoren zu unterstützen. Eine Abtastung entspricht einem Rahmen von dem CCN-Knoten **210** zu dem PCN-Knoten **220** und dem Antwortrahmen von dem PCN-Knoten **220** zu dem CCN-Knoten **210**. Jeder Rahmen weist 8 Byte Informationsdaten auf. In diesem Kontext enthält ein Rahmen sämtliche Rahmen-Overheads zum Bereitstellen von Informationsdaten. Das System ermöglicht einem Anwender die Abtastkapazität zwischen den PCN-Knoten **220** zuzuordnen, da einige PCN-Knoten **220** mehr als 1000 Abtastungen pro Sekunde erfordern können und andere PCN-Knoten **220** mit einer erheblich geringeren Abtastung betrieben werden können. Innerhalb einer festgelegten Kapazität sind zahlreiche Zuordnungen möglich, beispielsweise 2000 Abtastungen/sek für alle 16 Knoten oder 2000 Abtastungen/sek für 10 Knoten und 500 Abtastungen/sek für weitere 20 Knoten.

[0076] Es ist möglich, die Abtastrate für die zahlreichen PCN-Knoten **220** dynamisch anzugleichen, um den Hochleistungsanforderungen einiger PCN-Knoten **220** während gewisser Abschnitte der Maschinenzyklen gerecht zu werden, und anderen PCN-Knoten das Erzielen einer höheren Performanz während anderer Perioden des Maschinenzyklus zu ermöglichen, ohne daß die gesamte Kommunikationsbandbreite der PCN-Knoten **220**, die durch das Netzwerk ermöglicht wird, überschritten wird.

[0077] Für eine Kommunikation mit lediglich den DCN-Knoten **310** sollte das System in der Lage sein, einen Datendurchsatz von 30000 Meldungen (mit einer Durchschnittslänge von 16 Byte) pro Sekunde zu unterstützen, damit die Aktualisierungsanforderungen für Graphikdaten unterstützt werden. Eine Meldung ohne Rahmen-Overheads kann zwischen 1 und 33 Bytes lang sein und sie kann von einem DCN-Knoten **310** zu irgendeinem anderen der DCN-Knoten **310** mit einer positiven Bestätigung abgegeben werden. Die Wartezeit eines Rahmens übersteigt nicht 5 ms. Meldungen, die länger als 33 Byte sind, werden durch einen stabilen Software-Kommunikationstreiber gehandhabt. (Bei der genauen Definition der synchronen Datenverbindungssteuerung (SDLC) ist das Quellenadressenfeld bei der Kommunikationsarchitektur **10** ein Teil des SDLC-Datenfelds. Ein Anwender kann eine Meldung von bis zu 33 Bytes ausschließlich der Quelladresse senden, die automatisch durch die Kommunikationshardwarearchitektur **10** gehandhabt wird).

[0078] Für das kombinierte Netzwerk ist in dem System eine Vorrichtung vorgesehen, mit der ein Anwender die Kapazität zwischen den PCN-Knoten **220** und den DCN-Knoten **310** zuordnen kann. Die Zuordnungspriorität wird vorzugsweise den PCN-Knoten **220** eingeräumt. Ein Anwender ordnet vorzugsweise den Kommunikationsverkehr in optimierter Weise, um die Wartezeitanforderungen der DCN-Knoten innerhalb der Grenzen der Kanalbandbreite zu erfüllen.

[0079] Parameter der physikalischen Schicht der Bewegungssteuerungs-Kommunikationsarchitektur **10** sind wechselseitig abhängig und beeinflussen auch den Kommunikations-Hardwareentwurf und das Protokoll. Die gewählte Kanaltopologie besteht aus einem Mehrfachanschluß-, Bidirektional-, Halbduplexdifferential-Signalbus mit einer vorgegebenen Betriebsgeschwindigkeit von bis zu 10 Mbps (Megabit pro Sekunde). Demnach müssen die die Kommunikation regelnden Logikschaltungen ebenfalls an diese Geschwindigkeit angepaßt sein. Beim Anschalten wird das System mit der niedrigsten Vorgabegeschwindigkeit, 675 Kbps (Kilobit pro Sekunde) betrieben. Ein Anwender kann das Netzwerk auf eine höhere Geschwindigkeit einstellen: 1,25 Mbps, 3,5 Mbps, 5 Mbps oder bis zu 10 Mbps. Da die Geschwindigkeit wählbar ist, wird der Begriff "Bitzeit" benützt, da er geschwindigkeitsunabhängig ist.

[0080] Der Kanal erzeugt kein übermäßiges elektromagnetisches Rauschen, und gleichzeitig widersteht er dem Rauschen in dem Umfeld. In einem System kann Rauschen auf dem Kanal aufgrund der Hochgeschwindigkeitslogik, der Treiberschaltungen, der Motoren und Erddifferentialen einwirken. Es wird ein Differentialtreiber eingesetzt, der den RS422A und den RS485-Standard erfüllt. Ein derartiger Transceiver steht über den Betreiber zur Verfügung und kann mit bis zu 35 Mbps betrieben werden.

[0081] Im Zusammenhang mit der Kanallänge basieren aufgrund der Tatsache, daß die Gesamtlänge zwischen den beiden Enden des Netzwerks vorzugsweise niedriger als 35 Fuß ist, die Berechnungen der Übertragungsverzögerung auf der Maximallänge von 35 Fuß. Die Verzögerungszeit wird auch durch die Modulationsgeschwindigkeit und den Kabeltyp beeinflußt. Die vom Ausgang des Transceivers gemessene Verbindungslänge zu dem Kanalkabel wird vorzugsweise minimal gehalten. Der Kommunikationskanal ist vorzugsweise auch auf die Impedanz des ausgewählten Kabels abgestimmt. Die Zahl der Leiter für das Serienkommunikationssignal wird vorzugsweise zu 2 gewählt, was sowohl für das Kabel als auch den Verbinder gilt. Ein optionaler Leitungsdraht zum Bestimmen der physikalischen Reihenfolge der Knotenverbindungen kann erforderlich sein. In einem rauhen Umfeld muß das Kabel zum Schützen des Signals gegenüber einem Rauschen abgeschirmt sein, wohingehend bei niedriger Geschwindigkeit und/oder einem günstigen Umfeld das Abschirmen nicht erforderlich sein muß.

[0082] Bis zu 32 Knoten können an dem Bus **230** angeschlossen sein. Beispielsweise können bei einem Netzwerk mit ausschließlich PCN-Knoten **220** bis zu 31 der PCN-Knoten **220** und ein CCN-Knoten **210** vorliegen. Für ein Netzwerk mit lediglich DCN-Knoten **310** können bis zu 31 DCN-Knoten **310** und ein CCN-Knoten **210** an dem Kanal angeschlossen sein.

[0083] Ein sich langsam veränderndes Signal eignet sich nicht für eine Übertragung über eine lange Distanz. Andererseits kann ein besonders schnelles Schalten zur Emission von elektrischem Rauschen führen, und elektronische Komponenten können teuer werden. Bei dieser Implementierung wird das NRZI-(das invertierte NRZ-Schreibverfahren)-Codierschema bei der SDLC-Steuerung benützt. Bei der NRZI-Codierung werden Daten dadurch übertragen, daß der Zustand des Ausgangs immer dann geändert wird, wenn eine logische 0 übertragen wird. Immer dann, wenn eine logische 1 übertragen wird, bleibt der Ausgang wie bei dem vorhergehenden Bit unverändert und über die gesamte Bitzeit gültig. Ein Beispiel ist in Fig. 7 gezeigt. Bei der SDLC-Steuerung wird ein Bit-Einfügen zusammen mit der NRZI-Codierung eingesetzt. Treten fünf aufeinanderfolgende logische Einsen in dem Datenstrom auf, so wird das 0-Bit eingefügt.

[0084] Bei der Architektur **10** wird vorzugsweise ein Mehrfachanschlußbus **230** eingesetzt, der in und von sich selbst aus keine Vorrichtung zum Bestimmen der physikalischen Reihenfolge der Verbindung ist. Zum Erzielen der Modularität wird eine optionale Verbindung zusätzlich zu dem Mehrfachanschlußbus **230** eingesetzt. (Unter "Modulatität" wird hier die Fähigkeit verstanden, identische Module zum Durchführen derselben Funktion an unterschiedlichen physikalischen Stellen in einem Bewegungssteuersystem anzuschließen.) Der CCN-Knoten **210** bestimmt dann die Identität und die physikalischen Stelle sämtlicher Knoten und lädt – soweit erforderlich – neue Adressen zum Unterscheiden identischer Module, wie vollständiger nachfolgend beschrieben wird. Da das RS485-Differenzsignal zwei Drähte benützt, wird diese optinale Verbindung oft als "dritter Draht" bezeichnet. Wie in Fig. 8 gezeigt ist, wird der dritte Draht zum Bilden eines kostengünstigen Rings eingesetzt. Das im Zusammenhang mit der Kommunikation übertragene Steuersignal kann die physikalische Reihenfolge festlegen. Für den kostengünstigen Ring ist jeder Knoten mit einer Eingangsmeßleitung **720** versehen, sowie einer Logik **740** und einer Ausgangsleitung **730**, die mit dem Treiber **750** verbunden ist, der die Eingangsmeßleitung **720** des benachbarten Knotens treibt.

[0085] Bei der Architektur **10** erfolgt vorzugsweise die Bestimmung der physikalischen Reihenfolge der Verbindung wie folgt. Zum Rücksetzen des Netzwerks zieht ein Master **760** (typischerweise der CCN-Knoten **210**) seinen offenen Kollektorausgang (Leitung **730**) auf L-Pegel. Jeder Knoten in dem Netzwerk spricht auf ein niederwertiges Eingangssignal bei der Leitung **720** an, indem ein Rücksetzen erfolgt, sowie ein Ziehen an dessen offenem Kollektorausgang auf den L-Pegel; hierdurch breitet sich das Rücksetzsignal durch das Netzwerk aus. (Hier wird die gesamte Systemelektronik in einem Rücksetzzustand gehalten, mit Ausnahme des CCN-Knotens **210**). Zum erneuten Starten des Kommunikationsnetzwerks zieht der Master **760** seinen offenen Kollektorausgang auf den H-Pegel und sendet eine Mitteilung, die eine Antwort von einem Knoten verlangt, der gerade einen Rücksetzzustand verlassen hat. Wie in Fig. 10 gezeigt ist, antwortet der benachbarte Knoten **770** mit seiner Knotenidentifikation einschließlich der Vorgabeadresse. Dann fordert der Master **760** den Knoten **770** auf, seinen offenen Kollektorausgang anzuheben. Hierdurch wird dieselbe Situation bei dem zweiten Knoten **780** erzeugt, wie sie zuerst bei dem ersten Knoten **770** vorlag. Der Master **760** wiederholt die Sendemeldung, und der Prozeß wird wiederholt, bis der Master **760** auf seiner Eingabeleitung einen H-Pegel erfaßt.

Durch diesen Prozeß bestimmt der Master **760** die physikalische Reihenfolge der Knotenverbindungen, die Zahl der Knoten und den Typ eines Knotens (PCN-Knoten **220** oder DCN-Knoten **310**).

[0086] Der Master **760** setzt dann das Netzwerk erneut zurück, indem er seinen offenen Kollektorausgang auf L-Pegel zieht. Erfaßt der Master **760** einen L-Pegel an seinem Eingang **720**, so hebt er seinen offenen Kollektorausgang erneut an und sendet dann, falls erforderlich, eine neue Adresse, und anschließend sendet er einen Befehl zum Anheben des offenen Kollektorausgangs. Jeder Knoten spricht auf die neue gesendete Adresse lediglich dann an, wenn sein Eingang **720** auf H-Pegel liegt und wenn er noch nicht einem Befehl zum Anheben seines offenen Kollektoausgangs empfangen hat. Dieser Prozeß wird für jeden Knoten wiederholt, so daß das Netzwerk mit einer einzigen Adresse für jeden Knoten konfiguriert wird.

[0087] Dieser kostengünstige Ring läßt sich unter Einsatz eines offenen Kollektorausgangs mit einem Hochziehwiderstand am Eingang des benachbarten Knotens implementieren. In diesem Fall kann bei dem Empfänger ein einfaches digitales Filter zum Kompensieren des elektrischen Rauschens zwischen zwei benachbarten Knoten erforderlich sein, oder der Einsatz eines teureren RS485-Treibers kann erforderlich sein. Die Auswahl dieser Schnittstelle wird durch die Systemerdungstopologie bestimmt.

[0088] Die Architektur **10** basiert auf einem einfachen Protokoll auf der Datenverbindungsebene, da dies die Dokumentation, die Implementierung, das Debugging/Testen fördert und die Zuverlässigkeit erhöht.

[0089] Die hier angegebenenen Spezifikationen gelten für die Datenverbindungs-Steuerungsschicht (DLC), sowie unterhalb in der schichtweisen Kommunikationsarchitektur **10**. In diesem Zusammenhang besteht ein Kommunikationsknoten aus vielen Schichten. Eine oberhalb der Datenverbindungsschicht liegende Schicht, möglicherweise die Netzwerkschicht oder die Anwendungsschicht, generiert eine Meldung und fordert die Datenverbindungsschicht auf, diese zu senden. Ein Rahmen oder ein Paket wird durch Hinzufügen der erforderlichen Bits zu einer Meldung aufgebaut. Auf dem DLC-Niveau werden sämtliche Meldungen über den Rahmen **810** übertragen. Der gewählte SDLC-Rahmen **810** ist in Fig. 9 gezeigt. Der Rahmen **810** besteht aus dem Startflag, einer Adresse(en), einem Befehlsbyte, einem optinalen Datenfeld, Bits für die zyklische Blockprüfung (CRC), die den Bereich zwischen dem Adressenfeld und dem optionalen Datenfeld abdecken, und das Endflag. Eine Meldung ist in dem Datenfeld aufgenommen, und deren Länge beträgt 8 Byte für die Kommunikation für die PCN-Knoten **220** und 33 Byte bei DCN-Knoten **310** ausschließlich der Quelladresse. Bei der Kommunikation mit DCN-Knoten **310** kann es nötig sein, daß das optinale Datenfeld vorliegt, da die Architektur **10** das erste Byte des optionalen Datenfelds als Quelladresse benützt.

[0090] Durch die beiden Flags werden das Stimmungsadressenfeld, das Steuerfeld und das CRC-Feld identifiziert. Demnach kann die Länge einer Meldung in einem Rahmen ohne eine Bytezählung bestimmt werden. Die empfangenen Daten sind für einen Anwender über eine FIFO-Schnittstelle verfügbar. Entsprechend wird für übertragene Daten eine FIFO-Schnittstelle eingesetzt, um Anwenderdaten der Kommunikationshardware zur Verfügung zu stellen.

[0091] Die Architektur **10** ist vorzugsweise lediglich an dem SCLC-Rahmen **710** angepaßt, und sie entspricht nicht der SDLC-Steuerung im Hinblick auf die logische Datenverbindungssteuerung. Die Architektur **10** weist ihre eigene Definition für das Steuerfeld auf, und sie benützt sowohl Qeull- als auch Bestimmungsadressen für die Kommunikation mit DCN-Knoten **310**.

[0092] Ein Knoten muß vorzugsweise in der Lage sein, das Startflag zu detektieren, um den Empfangsprozeß auszulösen, sowie das Endflag, um den Empfang eines Rahmens **710** abzuschließen. Das Bitmuster der Flags muß eindeutig sein, und darf sich innerhalb des Rahmens **710** nicht wiederholen. Detektiert ein Knoten das Endflag mit oder ohne einem Rahmenfehler (d.h., ein Rahmen ist zu kurz oder zu lang), so sollte er in den Neustartzustand sobald wie möglich zurückkehren. Gemäß der SDLC ist 01111110 als Start/Endflag spezifiziert. Da bei der NRZI-Codierung die 0 als Übergang am Beginn einer Bitzeit definiert ist, muß der ruhige oder freie Zustand des Kanalsignals das Gegenteil des Signalpegels nach dem ersten O-Übergang des Rahmens **710** sein. Da dieser allererste Übergang von dem ruhigen Zustand im Vergleich zu anderen Bitübergängen bei hoher Geschwindigkeit und einer langen Übertragungsleitung sein kann, geht ein Zweibitmuster (O gefolgt durch 1) dem 01111110-Startflag voraus. Dieses Muster, das als eine Art von Präambel betrachtet werden kann, wird durch den Empfänger zur Rahmensynchronisierung benutzt. Zum Übertragen mehrerer Rahmen **710** wird vorzugsweise ein festgelegter Umfang einer Totzeit zwischen dem Endflag und dem Start eines neuen Rahmens **710** vorgesehen.

[0093] Wie in Fig. 10 gezeigt ist, weist jeder Knoten einschließlich des CNN-Knotens **210** vorzugsweise eine

Knotenidentifikation (ID) von drei Byte auf. Ein Teil der Knoten-ID wird als Knotenvorgabeadresse benützt. Jeder Knoten weist vorzugsweise ein Knotenadressenregister auf, das von der Adressenerkennungshardware benützt wird. Das Knotenadressenregister wird vorgabegemäß mit den niederwertigeren Bits der Knotenidentifikation geladen. Es läßt sich durch den CNN-Knoten **210** dann modifizieren, wenn das Netzwerk mit der Modularitätsoption konfiguriert ist. Die Fig. 10 zeigt das Laden eines Knotenadressenregisters. In dem Bewegungssteuersystem sind funktionelle Module Komponenten, die mit anderen Modulen kommunizieren. Die Identifikation eines Moduls in einem System oder in einer Reihe von zusammenhängenden Systemen ist zum Steuern der Systemvarianten und zum Konfigurieren des Systems wichtig. Die Architektur **10** ermöglicht die elektrische Identifikation und es erfolgt die Identifikation der Revisionsstufe der zugeordneten anwendungsspezifischen integrierten Schaltung (ASIC) für die Kommunikation. Für die elektrische Identifikation eines Moduls kann ein Mehrfachpositionsschalter benützt werden, und ein Teil hiervon diesem kann als physikalische Adresse eingesetzt werden (beispielsweise 5 Bit). Der CNN-Knoten **210** benützt vorzugsweise einen reservierten Befehl, um die Identifikationsinformation eines Knotens abzufragen. Empfängt einer der PCN-Knoten **220** oder der DCN-Knoten **310** diesen Befehl, so gibt er vorzugsweise den Registerwert mit 3 Byte an den CNN-Knoten **210** ab.

[0094] Der Wert eines Knotenadressenregisters erscheint vorzugsweise in dem Bestimmungsadressenfeld des SDLC-Rahmens. Die Architektur **10** nützt vorzugsweise auch eine Quelladresse, die ebenfalls von einem Knotenadressenregister kommen muß. Um bis zu 32 PCN-Knoten **210** und **32** DCN-Knoten **310** eindeutig zu identifizieren, sind 6 bit erforderlich. Ein Positionsschalter mit 5 Bit kann zusammen mit einem knotenabhängigen vorprogrammierten Bit in dem ASIC benützt werden. Lediglich der CNN-Knoten **210** kann die Sendeadresse erzeugen, während alle PCN-Knoten **220** und DCN-Knoten **310** in der Lage sind, die gesendeten Rahmen zu empfangen. Die Modularitätsoption der Architektur **10** ermöglicht das Laden der Adressen. Werden Adressen nicht geladen, so müssen die fest verdrahteten physikalischen Adressen eindeutig in dem in Fig. 2 gezeigten Bereich liegen.

Tabelle 2

	Knotenadressenwert		
CCN:	x0000000		
DCN:	x0000001	-x0011111	(01 - 1F)
PCN	x0100000	-x0111111	(20 - 3F)
Senden an alle	x1xxxxxx		
	x:	Bit nicht festgelegt	

[0095] Selbst bei der Modularitätsoption kann sich ein Betreiber so entscheiden, daß er die Adresse nicht lädt. Sind Adressen zu laden, so ordnet der CNN-Knoten **210** eine eindeutige Adresse jedem Knoten gemäß der physikalischen Reihenfolge der Verbindung zu. Der Ladeprozeß basiert vorzugsweise nicht auf einer Bestimmungsadresse, sondern benützt die Sendeadresse im Zusammenhang mit dem dritten Draht.

[0096] Bei einer Kommunikation von einem zu einem anderen gleichwertigen Element, beispielsweise von einem DCN-Knoten **310** zu einem DCN-Knoten **310**, können die empfangenen Knoten feststellen, welcher Knoten einen Rahmen gesendet hat, da das erste Byte in dem optionalen Datenfeld des Rahmens **710** mit einer Quelladresse für die Kommunikation der DCN-Knoten **310** gewählt ist: Kommunikationsvorgänge zwischen PCN-Knoten **220** erfordern nicht den Einsatz von Quelladressen, da der CCN-Knoten **210** sämtliche PCN-Knoten **220** direkt steuern.

[0097] Der CCN-Knoten **210** steuert genau den Zugriff auf den Bus **230** für alle Knoten. Für die Kommunikation mit den PCN-Knoten **220** sendet der CCN-Knoten **210** einen Rahmen **710** zu dem PCN-Knoten **220**. Hierin besteht die einzige Möglichkeit für den adressierten PCN-Knoten **220** zum Übertragen. Für die Kommunikation mit DCN-Knoten **310** gibt der CCN-Knoten **210** einen Befehl für eine Kommunikation von einem DCN-Knoten **310** zu einem anderen DCN-Knoten **310** (oder den CCN-Knoten **210**) ab. Wenn der CCN-Knoten **210** einen Rahmen zu dem DCN-Knoten **310** sendet, so kann dies ohne die Abgabe eines speziellen Zugriffssteuerbefehls erfolgen. Während die DCN-Knoten **310** Hardwarelogik für die Zugriffsteuerung enthalten, ist für die PCN-Knoten **220** keine spezielle Logik erforderlich. Die Adressenerkennungsfähigkeit der PCN-Knoten **220** ist ausreichend, um auf einen Befehl des CCN-Knotens **210** zu antworten.

[0098] Wie in Fig. 11 gezeigt ist, tastet der CCN-Knoten **210** zum Unterstützen der I/O-Funktionalitäten der PCN-Knoten **220** periodisch die PCN-Knoten **220** ab. Die Architektur **10** offeriert den Anwendern die Möglichkeit, die einzelne Abtastrate für jeden PCN-Knoten **220** zu steuern. Aus diesem Grund sendet der CCN-Knoten **210** einen speziellen Rahmen an alle Knoten, der als der Synchronisationsrahmen (SYNC-Rahmen) bezeich-

net-wird. Das Intervall für SYNC-Rahmen ist vorzugsweise anwenderkonfigurierbar, typischerweise zwischen 500 µs und einer Millisekunde. In dem Betriebsnormalmodus der Architektur **10** treten SYNC-Rahmen auf dem Bus **230** wie Taktimpulse auf. Das Zeitintervall zwischen zwei aufeinanderfolgenden SYNC-Rahmen wird als die (SYNC)-Tickperiode bezeichnet.

[0099] Der SYNC-Rahmen ist vor allem für die PCN-Knoten **220** vorgesehen. Das Fehlen der SYNC-Rahmen auf dem Bus **230** könnte auf einer Fehlerbedingung hinweisen. Ein SYNC-Rahmen-Überwachungsdatenzeitgeber kann zum Abschalten der I/O-Leitungen der PCN-Knoten **220** vorgesehen sein. Der PCN-Knoten **220** kann auch den SYNC-Rahmen benützen, um einen zeitlichen Versatz aufgrund des Einfügens eines Bits oder aus anderen Gründen benützen. Während einer Tickperiode lassen sich viele Rahmen zu unterschiedlichen PCN-Knoten **220** übertragen. Für einen perfekten Zeitablauf kann der PCN-Knoten **220** einen Betrieb im "synchronisierten" Modus durchführen und einen empfangenen Rahmen erhalten und dessen Inhalt synchron zu dem Ende eines SYNC-Rahmens ausführen. In dem synchronisierten Modus können die PCN-Knoten **220** so konfiguriert sein, daß sie auch die Eingabeleitungen unter simultanem Einsatz der SYNC-Rahmen abtasten. Hierdurch läßt sich eine besonders synchrone Aufnahme sämtlicher Sensoren in dem System erreichen.

[0100] Zum Eliminieren der zeitlichen Synchronisationsversatzes aufgrund des Einfügens von Bits bei dem SDLC-Protokoll startet der CCN-Knoten **210** die Übertragung am Beginn eines regulären Zeitintervalls. Dieses Intervall wird bstimmt, indem von einer Worst-Case-Biteinfügung für ein vorgegebenes System bei Einsatz der Architektur **10** ausgegangen wird. Für 8 Byte Daten mit ausschließlich logischen Einsen müssen 12 Bit logischer Nullen eingefügt werden. Die maximale Puffergröße für einen Übertragungspuffer und einen Empfangspuffer eines PCN-Knotens **220** werden durch den CCN-Knoten **210** nach dem Untersuchen der Tabelle der Knotenidentifikation bestimmt. Innerhalb eines Systems mit der Architektur **10** können PCN-Knoten **220** unterschiedliche Kommunikationspuffergrößen aufweisen. Die synchronisierte Übertragung der Meldungen der PCN-Knoten **220** basiert auf der Maximalgröße der Puffer der PCN-Knoten **220**. Demnach sollten Anwendungsprogramme vorzugsweise dieses Intervall für die Kommunikation mit PCN-Knoten **220** spezifizieren.

[0101] Die DCN-Knoten **310** erkennen auch den SYNC-Rahmen, da beispielsweise mit PCN-Knoten **220** einige Aktivitäten der DCN-Knoten **310** mit den Aktivitäten der PCN-Knoten **220** zu synchronisieren sind.

[0102] Die Kommunikation zwischen dem CCN-Knoten **210** und den PCN-Knoten **220** kann auch in einem "unmittelbaren" Modus durchgeführt werden, bei dem PCN-Knoten **220** auf Meldungen unmittelbar antworten, um empfangene Daten an ihre Ausgangsleitungen weiterzuleiten und/oder ihren Status unmittelbar rückzugeben. Vorzugsweise wird zum Aufrechterhalten der Synchronisation der CCN-Knoten **210** immer mit einem bestimmten PCN-Knoten **220** während desselben Zeitintervalls, bezogen auf den SYNC-Rahmen einer Tickperiode kommunizieren.

[0103] Als ein Beispiel sei angenommen, daß ein Netzwerk drei PCN-Knoten **220** enthält, A, B und C. Weiterhin sei angenommen, daß der Knoten A eine Aktualisierung bei jeder Tickperiode erfordert, während die Knoten B und C lediglich eine Aktualisierung bei jeder weiteren Tickperiode erfordern. Die Kommunikationsvorgänge können dann in einen Ablauf A, B; A, C; A, B; usw., eingeordnet werden. Die mit dem Stand der Technik Vertrauten erkennen, daß durch das Einordnen von Komunikationsvorgängen innerhalb einer Periode, deren Länge ein ganzzahliges Vielfaches der Tickperioden ist, sich die Servofunktionen für die Bewegungssteuerung und dergleichen erheblich vereinfachen lassen. Jedoch werden die mit dem Stand der Technik Vertrauten auch erkennen, daß bei Anwendungen, die einen gewissen Umfang von Synchronisationsstörungen bei den Zeitabläufen von Kommunikationsvorgängen tolerieren, die Kommunikationsvorgänge in dem unmittelbaren Modus nicht periodisch verlaufen müssen.

[0104] Bei einer bevorzugten Ausführungsform der vorliegenden Erfindung kann jeder der PCN-Knoten **220** und das Netzwerk wahlweise entweder in einem synchronisierten oder in einem unmittelbaren Modus betrieben werden, und das Netzwerk als ganzes kann in einem sychronisierten, einem unmittelbaren oder einem gemischten Modus betrieben werden.

[0105] Die Zugriffsteuerung für DCN-Knoten **310** ist so entworfen, daß eine Wartezeit des DCN-Knoten **310** garantiert ist, d.h. die maximale Wartezeit für die Übertragung. Da ein Anwender die gesamte Bandbreite für die PCN-Knoten **220** benützen kann, muß ein Anwender eine bestimmte Bandbreite den DCN-Knoten **310** zuordnen. Bei der Architektur **10** ist die Vorrichtung vorgesehen, mit der ein Anwender die maximale Wartezeit der DCN-Knoten **310** steuern kann. Ein Anwender. kann dynamisch eine kurze Wartezeit für einen wichtigen DCN-Knoten **310** auswählen.

[0106] Nachdem der CCN-Knoten **210** seine geplante Kommunikation mit den PCN-Knoten **220** innerhalb einer Tickperiode beendet hat, berechnet er die vor dem nächsten SYNC-Rahmen verbleibende Zeit. Ist die Zeit für eine Kommunikation mit einem DCN-Knoten **310** ausreichend, so initiiert der CCN-Knoten **210** eine Kommunikation von einem DCN-Knoten **310** zu einem anderen DCN-Knoten **310**, wie in Fig. 11 gezeigt ist. Bis zu einer Meldung kann dann zwischen den beiden DCN-Knoten **310** ausgetauscht werden. Anschließend berechnet der CCN-Knoten **210** erneut die verbleibende Zeit vor dem nachfolgenden SYNC-Rahmen und der Prozeß kann wiederholt werden.

[0107] Die Kommunikationsvorgänge von einem DCN-Knoten **310** zu einem anderen DCN-Knoten **310** oder von einem DCN-Knoten **310** zu einem CCN-Knoten **210** bestehen aus den durch den CCN-Knoten **210** gesendeten DAC-(DCN-Zugriffssteuerungs)-Rahmen, der Störvermeidungsperiode, einem durch den DCN-Knoten **310** übertragenen Informationsrahmen und den durch den anderen DCN-Knoten **310** initiierten Bestätigungsrahmen. Die DCN-Knoten **310** wirken zusammen, um eine Störung während der Störvermeidungsperiode zu vermeiden. Aus diesem Grund nützen die DCN-Knoten **310** die Daten in dem DAC-Rahmen sowie die Daten in ihrem internen Speicher, die durch den CCN-Knoten **210** im Zeitpunkt der Netzwerkkonfiguration geladen wurden.

[0108] Die Störvermeidungsperiode besteht aus einem Prioritäts- und 32 Zeitschlitzen, insgesamt also aus 33 Schlitzen. Ein Zeitschlitz ist als eine Zeitperiode definiert, indem die DCN-Knoten **310** den Start einer Rahmenübertragung durch andere DCN-Knoten **310** detektieren können. Die Periode eines Zeitschlitz ist unter Beachtung der Zeitkonstanten der Transceiverschaltung der DCN-Knoten **310** zu bestimmen, sowie der Verzögerung aufgrund der logischen Entscheidungsfindung, der Kanallänge und der Ausbreitungsverzögerung. Ein Zeitschlitz bei einem Mehrfachanschlußbus kann mehrere Bitzeiten oder den Bruchteil einer Bitzeit einnehmen. Alternativ kann bei der Architektur **10** eine Ringkombination eingesetzt werden, obgleich die Zeitschlitzperiode länger wäre.

[0109] Die Architektur **10** ermöglicht den Prioritätsschlitz für ein Bewegungssteuer-Untersystem, bei dem der CCN-Knoten **210** auf das Auftreten eines Ereignis in einem bestimmten DCN-Knoten **310** warten kann. Derartige Ereignisse können in zahlreichen DCN-Knoten **310** auftreten. Wird ein wichtiges Ereignis vorweggenommen, so sollte den DCN-Knoten **310** eine hohe Priorität eingeräumt werden. Zum Erfüllen dieses Erfordernisses weist der CCN-Knoten **210** vorzugsweise die Fähigkeit auf, die Zuordnung eines DCN-Knotens **310** zum unmittelbaren Übertragen nach einem DAC-Rahmen unter Einsatz des Prioritätsschlitzes durchzuführen. Der CCN-Knoten **210** kann die Zuordnung des Prioritätsmodus löschen und diese zu einem anderen DCN-Knoten übergeben. Die DCN-Knoten **310** weisen Flags auf, die anzeigen, ob sie im Prioritätsmodus oder in dem Schlitzmodus betrieben werden sollten. Details der Zugriffsteuerung für die DCN-Knoten **310** sind in den Fig. 11 bis Fig. 14 gezeigt.

[0110] Wie in Fig. 12 gezeigt ist, lädt zum Koordinieren der DCN-Knoten **310** während der Störvermeidungsperiode der CCN-Knoten **210** eine eindeutige Zahl in jeden -der DCN-Knoten **310** im Zeitpunkt der Netzwerkkonfiguration. Der DCN-Knoten **310** speichert die Zahl in seinem Knotenschlitzregister (note slot register, NSR). Wie in Fig. 12 gezeigt ist; kennzeichnet F1 das Prioritätsmodusbit (der CCN-Knoten **210** schreibt in F1 über den Rahmen), und F2 ist ein durch den DCN-Knoten **310** gesetztes/rückgesetztes Bit zum Vermeiden aufeinanderfolgender Übertragungen, und der DCN-Knoten **310** enthält auch ein Knotenadreßregister (node address register, NAR). Für die in Fig. 13 gezeigte Kommunikation des DCN-Knotens **310** im Normalbetrieb sendet der CCN-Knoten **210** vorzugsweise auch eine Schlitzsteuernummer (slot control number, SCN) in dem DAC-Rahmen. Die SCN bestimmt den Knoten zum Übertragen unter Einsatz des ersten Zeitschlitzes unmittelbar nach dem Prioritätsschlitz. Empfängt beispielsweise ein Knoten die SCN in einem DAC-Rahmen, die mit dem NSR-Wert übereinstimmt, d.h. ist die Differenz 0, so kann dieser inmittelbar nach dem Prioritätsschlitz übertragen. Wie in Fig. 14 gezeigt ist, berechnen die DCN-Knoten **310** die Differenz zwischen der SCN und dem NSR in Modulo-32-Arithmetik, was anzeigt, wieviele Schlitze einer der DCN-Knoten **310** abwarten muß, bevor er eine Übertragung initiiert. Die Differenz ist bei den DCN-Knoten **310** eindeutig, da das NSR der DCN-Knoten **310** bei sämtlichen DCN-Knoten **310** eindeutig ist. Zum Vermeiden einer Störung werden keine zwei NSR-Werte dupliziert. Der CCN-Knoten **210** verändert den SCN-Wert derart, daß alle DCN-Knoten **310** eine gleiche Chance auf den Zugriff zu dem Bus **230** haben.

[0111] Wie in Fig. 15 gezeigt ist, initiiert ein DCN-Knoten **310** dann, wenn er eine Meldung zu übertragen hat und wenn er einen DAC-Rahmen empfängt, eine Übertragung im Prioritätsschlitz, wenn der DCN-Knoten **310** sich in dem Prioritätsmodus befindet. Andernfalls wartet der DCN-Knoten **310** auf die Periode, die zu der Differenz zwischen dem NSR und der empfangenen SCN-Zahl identisch ist. Wird während dieser Periode keine Kommunikation erfaßt, so initiiert der DCN-Knoten **310** eine Übertragung. Andernfalls muß der DCN-Knoten

310 auf den nächsten DAC-Rahmen des Prozesses warten. Wird ein Rahmen durch einen DCN-Knoten **310** übertragen, so sendet der empfangene DCN-Knoten **310** unmittelbar einen Bestätigungsrahmen. Nachdem ein DCN-Knoten **310** einen Rahmen erfolgreich überträgt, wird er in der nächsten Störvermeidungsperiode keine Übertragung initiieren, so daß anderen DCN-Knoten **310** einen möglichen Zugriff haben.

[0112] Für das System mit 31 DCN-Knoten **310**, d.h. die Maximalkonfiguration, muß der NSR-Wert die Knotenadresse bei jedem DCN-Knoten **310** sein. Der NSR-Vorgabewert ist die Knotenadresse beim Rücksetzen. Der CCN-Knoten **210** kann die Werte zu den NSR-Registern zum Verändern der Vorgabewerte laden. Der NSR-Wert muß eindeutig sein, wie eine Adresse der DCN-Knoten **310**. Die Ladefähigkeit ermöglicht einem Anwender DCN-Knoten **310** frei zu (re-)konfigurieren, um Anforderungen im Hinblick auf Zugriffsteuervorgänge zu erfüllen.

[0113] Zum Verbessern des Wirkungsgrades bei Netzwerken mit im wesentlichen weniger als 31 DCN-Knoten **310** erzeugt die Hardware des CCN-Knotens **210** vorzugsweise die DAC-Rahmen in einer optimierten Weise. Die SCN-Zahlen in einem DAC-Rahmen wird von dem Zugriffsteuerregister (ACR) des CCN-Knotens **210** kopiert. Das ACR wird automatisch bei jedem DAC-Rahmen inkrementiert. Erreicht das ACR einen bestimmten Maximalwert (anwenderspezifizierbar), so wird für eine Wiederholung das Register rückgesetzt. Der Maximalwert kann die Zahl der DCN-Knoten **310** in dem Netzwerk sein. Unter Einsatz dieses Merkmals wird ein Kommunikationstreiber unter Optimierung der Zugriffsteuervorgänge aufgebaut.

[0114] Der CCN-Knoten **210** kann die DAC-Rahmen innerhalb einer Tickperiode solange wiederholen, solange die verbleibende Zeit vor dem nächsten SYNC-Rahmens für eine DCN-Knommunikation ausreicht. Liegen keine Aktivitäten vor, die DCN-Knoten **310** einschließen, so ist es möglich, viele DAC-Rahmen hintereinander ohne andere Rahmen zu beobachten.

[0115] In einer bevorzugten kostengünstigen Ausführungsform wartet der CCN-Knoten **210** während einer festgelegten Zeit, die der Summe der Störvermeidungsperiode und der schlechtmöglichsten Zeit für eine Kommunikation eines PCN-Knotens **310** ist, und inkrementiert dann das ARC-Register und überträgt eine DAC-Rahmen, wenn ausreichend Zeit verbleibt.

[0116] Soll die Bandbreite des Netzwerks vollständiger ausgenützt werden, so kann der CCN-Knoten **210** das Netzwerk überwachen und einen DAC-Rahmen unmittelbar nach der Störvermeidungsperiode übertragen, wenn kein DCN-Knoten **310** mit einer Übertragung begonnen hat, oder am Ende einer Meldung, wenn ausreichend Zeit verbleibt.

[0117] Wie in Tabelle 2 gezeigt ist, ist dem CCN-Knoten **210** vorzugsweise die Adresse 00(hex), die die Adresse eines DCN-Knoten **310** ist. Das heißt, das DCN/PCN-Bit ist 0. Demnach arbeitet bei einer Echtzeitkommunikation zwischen dem CCN-Knoten **210** und einem DCN-Knoten **310** (im Gegensatz zur Systemkonfiguration oder zur Kommunikationssteuerung) der CCN-Knoten **210** im wesentlichen in der gleichen Weise, und er enthält im wesentlichen die gleiche Schaltung, wie sie oben für den CCN-Knoten **210** beschrieben ist.

[0118] Die Architektur **10** ermöglicht die Detektion dreier Arten von Fehlern: Rahmenfehler, Zeitablauf und CRC-Fehler. Die Detektion einer Fehlerbedingung wird vollständig in Hardware implementiert. Obgleich ein Knoten zahlreiche Arten von Fehlern detektieren kann, zeigt er in seinem Statusregister vorzugsweise lediglich an, ob ein Fehler aufgetreten ist oder nicht. Ein Datenstrom, der nicht in einen festgelegten Rahmen paßt, wird als Rahmenfehler betrachtet. Beispiele enthalten: ein nicht definiertes Flagmuster, wenn ein bestimmtes erwartet wird, ein vorzeitiges Endflag, ein nicht erwartetes Startflag, eine nicht definierte Adresse, ein nicht definierter Steuercode, ein Überschreiten der festgelegten Länge eines Datenfelds oder ein Vorliegen eines Datenfelds, wenn kein Datenfeld erwartet wird.

[0119] Ein Zeitablauffehler wird festgestellt, wenn ein erwartetes Ereignis nicht auftritt. Überträgt der CCN-Knoten **210** einen Befehl zu einem PCN-Knoten **220**, so erwartet er unmittelbar eine Antwortmeldung oder eine explizite Bestätigung des adressierten PCN-Knotens **220**, da diese Funktion in Hardware implementiert ist. Wird keine Antwort innerhalb einiger Bitzeiten erhalten, so stellt der CCN-Knoten **210** den Zeitablauffehler für den PCN-Knoten **220** fest. Empfängt der PCN-Knoten **220** den SYNC-Rahmen nicht regulär, so erfaßt er den Zeitablauffehler. Überträgt ein DCN-Knoten **310** einen Daten-(Informations)-Rahmen an einen anderen DCN-Knoten **310**, so erwartet er eine positive oder negative Bestätigung von dem empfangenen Knoten innerhalb einiger Bitzeiten. In diesem Fall erfaßt der übertragende DCN-Knoten **310** den Zeitablauffehler.

[0120] Die Durchführung und Überprüfung. mit Hilfe der zyklischen Blockprüfung (CRC) stellt die wichtigste

Maßnahme zum Detektieren von Fehlern dar. Vorzugsweise wird der DRC-16 als Rahmenüberprüfungssequenz eingesetzt, und dieser wird durch die algebraische Gleichung $X^{16} + X^{15} ... + X^2 + 1$ dargestellt. Die CRC-Generierung ist in Fig. 16 gezeigt. Während des Betriebs wird ein Bit empfangen und wird mit dem Bit 15 des momentanen CRC gemäß EXOR verknüpft und in einem Zwischenspeicher abgelegt. Das Ergebnis der EXOR-Verknüpfung des Bits 15 mit dem empfangenen Bit wird dann mit dem Bit 4 EXOR-verknüpft, und das Bit 11 des CRC wird um eine Position nach rechts verschoben. Das Bit in dem Zwischenspeicher wird in die Position 0 verschoben.

[0121] Bei der Architektur **10** kommen vorzugsweise zwei Arten von Rahmen zum Einsatz: Informationsrahmen und Steuerrahmen. Der Rahmentyp läßt sich in dem Steuerfeld spezifizieren. Einer der Steuerrahmen ist der Bestätigungs-(ACK)-Rahmen. Bei der Architektur **10** kann die Bestätigung explizit mit dem ACK-Rahmen erfolgen, oder implizit, wenn ein Antwortrahmen empfangen wird.

[0122] Für die Kommunikation mit den PCN-Knoten **220** sendet der CCN-Knoten **210** einen Rahmen, und anschließend sendet innerhalb mehrerer Bitzeiten der adressierte PCN-Knoten **220** einen Rahmen zurück, was zu einer Bestätigung führt. Aufgrund der hohen Abtastrate der PCN-Knoten **220** erfolgt keine Bestätigung für den PCN-Knoten **220** durch den CCN-Knoten **210**.

[0123] Für die Kommunikation mit den DCN-Knoten **310** werden sämtliche Rahmen vorzugsweise über den ACK-Rahmen innerhalb mehrerer Bitzeiten bestätigt. Jedoch muß der ACK-Rahmen selbst nicht bestätigt werden, und vorzugsweise wird er es auch nicht. Die Informationsrahmen der DCN-Knoten **310** benützen sowohl Quellals auch Bestimmungsadressen. Da ein ACK-Rahmen unmittelbar auf einen Informationsrahmen folgt, benützt der ACK-Rahmen nicht die Quelladresse. Diese unmittelbare Bestätigung ist zum Vermeiden von einer Zwischenspeicherung von Meldungen ausgebildet. Der DCN-Knoten **310** sendet eine negative Antwort unter Einsatz des Bestätigungs-Rahmenformats lediglich dann, wenn er aufgrund einer Nichtverfügbarkeit eines Empfangspuffers einen Rahmen nicht empfangen kann.

[0124] Ein Empfang der Knoten kann entweder einen Rahmenfehler oder einen CRC-Fehler feststellen. Er kann die Art sowie das Auftreten der Fehler insgesamt bestimmen, den fehlerhaften Rahmen übergehen, jedoch keine Fehlerkorrektur durchführen. Ein übertragener Knoten ist immer für die Fehlerkorrektur bei Erfassung eines Zeitablauffehlers oder eines Rahmenfehlers während einer Bestätigung zuständig. Diese Strategie vermeidet mögliche Störungen beim Zugriff auf den Bus während einer Fehlerbeseitigung.

[0125] Bei der Kommunikation mit PCN-Knoten **220** soll keine Fehlerkorrektur durchgeführt werden. Stattdessen werden Fehler ignoriert. Erfaßt jedoch der CCN-Knoten **210** eine übermäßige Zahl von Kommunikationsfehlern während einer festgelegten Periode, so kann er bestimmen, daß das Netzwerk stillgelegt wird.

[0126] Bei der Kommunikation mit DCN-Knoten **310** basiert die Fehlerkorrektur auf einem erneuten Versuch. Im Zusammenhang mit der Zeitablauffehler-Bedingung oder einem Fehler bei dem erwarteten ACK-Rahmen wird der übertragende Knoten viele Versuche durchführen, bevor er anzeigt, daß eine Kommunikationsverbindung zu einem adressierten Knoten unterbrochen ist. Das Zeitintervall für die erneute Übertragung muß lange genug sein, so daß eine Ümfeldbedingung, die den Fehler auslöst, nicht länger vorliegt. Demnach führt der DCN-Knoten **310** den erneuten Versuch nicht unmittelbar durch, sondern wartet auf seine nächste Gelegenheit, die durch einen Verzögerungszeitgeber für eine erneute Übertragung bestimmt ist. Zeigt ein DCN-Knoten **310** über das ACK-Rahmenformat an, daß er eine Meldung aufgrund eines nicht verfügbaren Puffers nicht empfangen kann, so sollte das Zeitintervall für die erneute Übertragung lange genug sein, so daß die CPU des DCN-Knoten **310** dessen Empfangspuffer löscht. Bei der Architektur **10** ist das Löschen des Empfangspuffers innerhalb einer vorgegebenen Maximalzeit durch den DCN-Knoten **310** erforderlich.

[0127] Die PCN-Knoten **220** und die DCN-Knoten **310** weisen einen Übertragungspuffer und einen Empfangspuffer unter Steuerung der Kommunikationshardware auf. Andererseits weist der CCN-Knoten **210** 31 Pufferpaare für die Kommunikation mit den PCN-Knoten **220** auf, sowie lediglich ein Pufferpaar für die Kommunikation mit den DCN-Knoten **310**. Jeder PCN-Knoten **220** ist so entworfen, daß er nach einer positiven CRC-Überprüfung seinen Empfangspuffer überschreibt. Der CCN-Knoten **210** gibt seinen Empfangspuffer frei, wenn er eine Übertragung zu einem PCN-Knoten **220** durchführt. Ein DCN-Knoten **310** ist für die Übertragung einer Meldung zu einem anderen DCN-Knoten **310** freigegeben, wenn davon ausgegangen wird, daß der adressierte Knoten einen verfügbaren Empfangspuffer aufweist. Wird eine negative Antwort empfangen, so führt der initiierende DCN-Knoten **310** eine erneute Übertragung durch.

[0128] Eine Meldungsreihung (lediglich für die Kommunikation für DCN-Knoten **210**) wird eingefügt, um dup-

lizierte Meldungen zu erfassen (d.h., eine identische Meldung wird mehr als einmal übertragen, und zwar aufgrund einer erneuten Übertragung bei Erfassung eines Fehlers in dem ACK-Rahmen, oder wenn kein ACK-Rahmen vorliegt). Eine duplizierte Meldung wird erkannt und nicht berücksichtigt. Da eine Warteschlangenbildung für eine Meldung nicht benützt wird und jede Meldung bestätigt wird, reicht 1 Bit pro Einfachkommunikation von Knoten zu Knoten. Liegen bis zu 31 DCN-Knoten **310** und der DCN-Knoten **310** vor, so sind 34 Bit Speicherplatz pro DCN-Knoten erforderlich, damit 32 Paare von Meldungsreihenfolgezahlen gespeichert werden. Dies unterscheidet sich von dem SDLC-Schema, bei dem ein Knoten bis zu sieben Meldungen ohne Bestätigung senden kann, da es bis zu sieben Empfangspuffer spezifiziert und 3 Bit für die Meldungsreihenfolgezahl benützt. Diese Reihenfolgezahl wird zum Detektieren duplizierter Meldungen benützt. Da bei der Hardware der Architektur **10** lediglich ein Empfangspuffer möglich ist, reicht ein Bitpaar zum Detektieren einer duplizierten Meldung.

[0129] Bei der Einreihung einer Meldung sind vorzugsweise lediglich zwei Knoten, beispielsweise als Knoten A und B, mit einbezogen. Es liegt vorzugsweise ein Paar von Laufnummern pro Einwegübertragung vor (d.h., die Übertragungslaufnummer des Knotens A steht lediglich im Zusammenhang mit der Empfangslaufnummer des Knotens B). Ein Knoten unterhält ein Paar von Empfangs/Übertragungs-Laufnummern, da er sowohl überträgt als auch empfängt. Laufnummern sind vorzugsweise synchronisiert (Rücksetzen zu 0), wenn das Netzwerk beim Anfahren der Stromversorgung initialisiert wird. überträgt der Knoten A eine Meldung, so sendet er vorzugsweise seine in dem Steuerfeld gespeicherte Übertragungslaufnummer. Empfängt der Knoten B die Meldung ohne einen Fehler, so kippt er die Empfangslaufnummer und führt eine Rückantwort mit dem ACK-Rahmen durch. Empfängt der Knoten A den ACK-Rahmen korrekt, so kippt er die Übertragungslaufnummer, wodurch ein Zyklus beendet wird. Am Ende jedes erfolgreichen Zyklus sollten die Übertragungslaufnummer des Knotens A und die Empfangslaufnummer des Knotens B übereinstimmen. Eine Laufnummer ist in einem Meldungsrahmen enthalten, sie ist jedoch nicht in dem ACK-Rahmen enthalten.

[0130] Es sei angenommen, daß der Knoten A einen Rahmen gesendet hat und bei dem Knoten B ein Fehler aufgetreten ist. Der Knoten B führt keine Bestätigung durch. Keine Ablaufbits werden beeinflußt, und der Knoten A sollte erneut übertragen. In einem anderen Fall sei angenommen, daß der Knoten A einen Rahmen gesendet hat, und der Knoten B diesen korrekt empfangen hat. Der Knoten B hat eine Bestätigung gesendet, jedoch ist in dem Knoten A ein Fehler aufgetreten. Demnach wird der Knoten A den Rahmen erneut übertragen, obgleich der Knoten B den Rahmen bereits korrekt empfangen hat. Das Ablaufbit verhindert, daß der Knoten B einen duplizierten Rahmen empfängt.

[0131] Ein Empfangsknoten kann demnach eine duplizierte Meldung erkennen, daß seine gespeicherte Laufnummer nicht mit der empfangenen Laufnummer abgestimmt ist. Der Knoten läßt die duplizierte Meldung unberücksichtigt, sendet jedoch den ACK-Rahmen erneut zurück. Wird dieser ACK-Rahmen korrekt übertragen, so ist das Paar der Laufnummern der Einwegübertragung erneut synchronisiert. Es kann erforderlich sein, daß die Datenverbindungsschicht die Herstellung der Verbindung an eine höhere Schicht berichtet. Werden Laufnummern initialisiert, so sollten die Empfangs- und Übertragungspuffer gelöscht sein.

[0132] Die im Zusammenhang mit der Rahmenerzeugung, dem Empfang, der Deutung und zugeordneten Aktionen erforderlichen Logikfunktionen sind in Hardware realisiert. Bei der Architektur **10** ist die Zahl der definierten Rahmen, wenn möglich, minimiert. Obgleich bei der Architektur **10** der SDLC-Rahmen eingesetzt wird, erfolgt bei der Architektur **10** die Definition des Steuerfelds eines SDLC-Rahmens in eigener Weise, um die Datenverbindungssteuervorgänge in Hardware zu implementieren. Bei der Architektur **10** sind ferner zusätzliche Merkmale vorgesehen, die für die Systementwicklung wesentlich sind, beispielsweise ein Hardware-Rücksetzsignal oder eine Knotenidentifikation.

[0133] Wie bei der SDLC-Steuerung liegen zwei Arten von Rahmen vor: Informations-(Daten)-Rahmen und Steuerrahmen. Ein Informationsrahmen trägt eine Meldung, während die Steuerrahmen die Kommunikationsvorgänge steuern, um das Netzwerk in einen Zustand zu versetzen, in dem Informationsrahmen ausgetauscht werden können. Der Typ eines Rahmens wird anhand des Codes in dem Steuerfeld unterschieden. In Fig. 17 ist das Rahmenformat für einen Steuerrahmen des CCN-Knotens **210** gezeigt. Die Fig. 18 zeigt das Rahmenformat für einen Datenrahmen von den CCN-Knoten **210** zu einem PCN-Knoten **220** oder umgekehrt, während die Fig. 19 das Format für einen Datenrahmen von einem DCN-Knoten **310** (dem CCN-Knoten **210**) zu einem DCN-Knoten **310** zeigt.

[0134] Die Rahmenaustauschvorgänge lassen sich in die folgenden Kategorien gruppieren: Netzwerkkonfiguration (Boud-Rate, Vorgänge im Zusammenhang mit dem dritten Draht, beispielsweise einem Adressenladevorgang), Netzwerksteuerung (SYNC-Rahmen, Zugriffsprioritätszuordnung/Löschung), Status/Registerab-

frage, Schreiben in Steuerregister, Vorgänge im Zusammenhang mit der Boudary-Scan-Logik oder echtzeitbezogene Vorgänge. Einige Austauschvorgänge beeinflussen alle Knoten, einige lediglich die PCN-Knoten und andere lediglich die DCN-Knoten.

[0135] Die Fig. 20 bis Fig. 22 zeigen Netzwerkkonfigurations-Rahmenaustauschvorgänge. In Fig. 20 ist das Senden durch den CCN-Knoten **210** zum Einstellen der Baud-Rate gezeigt. Hier spezifizieren BCA=(x1xxxxx), DCFI=TBD, BRS=Daten die Baud-Rate (ein Byte). Es ist zu erkennen, daß bei jedem Knoten die Baud-Rate eingestellt sein sollte. In **Fig.** 21 ist das Senden durch den CCN-Knoten **210** für ein Netzwerk mit der Option des dritten Drahtes (Modularität) gezeigt. Die Fig. 21A zeigt den Empfang einer geladenen Adresse, wenn der Eingang bei dem dritten Draht auf L-Pegel liegt und der Ausgang bei dem dritten Draht auf H-Pegel liegt. Hierbei zeigen BCA=(x1xxxxx), DCF2, DCF22= den Empfang der Adresse an. Die Fig. 21B zeigt das Setzen des Ausgangs bei dem dritten Draht auf H-Pegel, wenn der Eingang bei dem dritten Draht auf H-Pegel liegt und der Ausgang bei dem dritten Draht auf L-Pegel liegt. Hierbei gilt BCA=(x1xxxxx), DCF3=Absenken des dritten Drahts, CCF33=abgesenkt. Die Fig. 22 zeigt den. Netzwerkkonfigurationsrahmen (in Software implementiert), und der CCN-Knoten **210** teilt den DCN-Knoten **310** mit, welche DCN-Knoten **310** und welche PCN-Knoten **220** an dem Netzwerk angeschlossen sind.

[0136] Die Fig. 23A, Fig. 23B, Fig. 24A und Fig. 24B beschreiben die Lese/Schreibvorgänge bei den Knotenregistern des CCN-Knoten **210**. Bei den Fig. 23A und Fig. 23B ist die Statusabfrage (oder das Lesen von Registern) bei einem Knoten durch den CCN-Knoten **210** gezeigt. Hierbei gilt DA=Knotenadresse, CCNA=0, SADD=Quelladresse, CCF4, DCF44, DCF5, DCF55=TBD DATA8=bis zu 8 Byte, DATA33=bis zu 33 Byte. Diese Befehle dienen zum Lesen der Knotenregister. und des Status. Die Fig. 24A und Fig. 24B zeigen die Schreibvorgänge bei dem (den) Steuer/Konfigurationsregister(n) des CCN-Knotens **210**. Hierbei gilt DCF6, DCF66=TBD. In Fig. 24A ist die Antwort des PCN-Knotens **220** ein Leerrahmen zum Anzeigen einer Bestätigung.

[0137] Die Fig. 25 und Fig. 26 zeigen Austauschvorgänge für den Netzwerksteuerrahmen. Die Fig. 25 zeigt den Rahmenaustausch für das Löschen der Zugriffsprioritätszuordnung bei einem DCN-Knoten **310**. Hier gilt DCF9=zeige Zuordnen/Löschen der hohen Priorität an, DCF99=ACK oder (NAK). Die Fig. 26 zeigt das Senden des SYNC-Rahmens. Hierbei gilt, DCF7=Code zum Anzeigen des SYNC-Rahmens.

[0138] Der CCN-Knoten **210** wird zum Initiieren des Scan-Logiktests aktiv. Dieser kann auf Information basieren, die durch ein unabhängiges Testgerät zur Verfügung gestellt wird, das mit dem Netzwerk verbunden wird, oder auf der Grundlage einer Systemselbsttestinformation basieren, die in den CCN-Knoten **210** einbezogen ist. Hierbei generiert der CCN-Knoten **220** Testvektoren, die über eine Netzwerkverbindung zu der Scan-Logik derjenigen Knoten von dem CCN-Knoten **210**, den PCN-Knoten **310** und den PCN-Knoten **220** verschoben werden, die integrierte Schaltungen enthalten, die den Scan-Test unterstützen. Diese Knoten werden in den geeigneten Scan-Testmodus versetzt. Es ist zu erwähnen, daß das (wenn vorhandene) Boundary-Scan-Logikergebnis durch Einsatz eines CCN-Knotens **210** mit Hilfe des Abfragebefehls unter Einsatz eines CCN-Knotens **210** gelesen werden kann.

[0139] Die Fig. 27, Fig. 28, Fig. 29A und Fig. 29B zeigen Boundary-Scan-bezogene Rahmenaustauschvorgänge. In Fig. 27 ist der Rahmenaustausch für Boundary-Scan-Logikdaten zu den DCN-Knoten **310** und/oder den PCN-Knoten **220** gezeigt. Hierbei gilt, DCFA=Anzeigen der Boundary-Scan-Logikmeldung, DCFA=Code für ACK oder NAK. Die Fig. 28 zeigt den Rahmen, der durch den CCN-Knoten **210** an die DCN-Knoten **310** und die PCN-Knoten **220** abgegeben wird, zum Anweisen der Boundary-Scan-Logik zum Festlegen des Scan-Modus (beispielsweise zum Bewirken eines Anwendungslogik-zu-Scanlogik-Datentransfers oder umgekehrt, DCFX=Einstellen des durchzuführenden Scanmodus). Der CCN-Knoten **210**, die DCN-Knoten **310** und die PCN-Knoten **220** können vorzugsweise integrierte Schaltungen enthalten, die Scan-Logiktests-Hilfspins aufweisen; diese Schaltungen werden in Art einer Daisy-Chain zum Bilden eines seriellen Scantestrings oder einer Scan-Logik verbunden, die bei der Netzwerkschaltung des CCN-Knoten **210**, der DCN-Knoten **310** und der PCN-Knoten **220** beginnt und endet. Die Fig. 29A zeigt das Abfragen bei einem DCN-Knoten **310** und/oder einem PCN-Knoten **220** gemäß dem Boundary-Scan-Logikergebnis. Ist die Boundary-Scan-Logik des DCN-Knotens **310** nicht fertig, so gilt Fig. 29B. Hierbei gilt, DCFB=Abfragebefehl, DCFBB=Anzeigen eines Datenrahmens, DCFBC=Anzeigen, daß keine Meldung zu senden ist.

[0140] Die Fig. 30 und Fig. 31 zeigen Datenrahmenaustauschvorgänge. In Fig. 30 ist der Datenaustausch zwischen dem CCN-Knoten **210** und einem PCN-Knoten **220** gezeigt. Hierbei gilt, DCF8, DCF88=Code zum Anzeigen eines Datenaustausches zwischen dem CCN-Knoten **210** und einem PCN-Knoten **210**. Die Fig. 31 zeigt den Datenrahmen vom CCN-Knoten **210** zu dem DCN-Knoten **310** (Laufzahl wird benützt). Hierbei gilt,

DCFC=Code zum Anzeigen eines Datenrahmens, DCF99=Code für ACK oder NAK.

[0141] Die Fig. 32 zeigt den Datenrahmen vom DCN-Knoten zum DCN- Knoten **310** (oder zum CCN-Knoten **210**) (Laufzahl benützt). Hierbei. gilt, DCFD=Code zum Anzeigen eines DAC-Rahmens, SCN=Schlitzzahl, DCFE=Code zum Anzeigen eines Datenrahmens, DCF99=Code für ACK oder NAK.

[0142] In Tabelle 3 ist eine Zusammenfassung von ßefehlcodes gezeigt, die für besonders bevorzugte Systemimplementierungen der Architektur **10** zu bestimmen sind.

Tabelle 3

```
DCF1 = Baud-Rateneinstellsendevorgang - - Baud-Ratendaten
       sollten folgen
DCF2 = Laden einer Adresse zu einem Knoten DCF22
DCF3 = Anheben des Ausgangs bei dem dritten Draht DCF33
DCF3A = Netzwerkkonfigurationsrahmen (softwaregeneriert)
DCF4 = Status-(Register)-Abfrage bei einem PCN-Knoten DCF44
DCF5 = Status-(Register)-Abfrage bei einem DCN-Knoten DCF55
DCF6 = PCN-Knoten-Anwendungshardware-Konfigurierungsbefehl
       DCF66
DCF7 = SYNC-Rahmen
DCF8 = Datenaustausch zwischen CCN-Knoten 210 und einem
       PCN-Knoten DCF88
DCF9 = Zuordnen/Löschen der Zugriffspriorität DCF99
DCF99= Code für ACK oder NAK
DCFA = Laden der Boundary-Scan-Logikdaten
DCFB = Abfrage bei einem DCN-Knoten der Boundary-Scan-
       logikantwort DCNBB, DCNBC
DCNC = Datenrahmen zwischen DCN-Knoten und CCN-Knoten
DCND = DAC-Rahmen
DCNE = Meldung von einem DCN-Knoten zu einem DCN-Knoten
```

[0143] Die Datenverbindungsschicht-Hardware der Architektur **10** ist nicht auf einen Industriestandard DLC abgestimmt, beispielsweise dem SDLC oder dem IEEE 802.2 (Internmationaler Standard 1508802-2 IEEE Std. 802.2, "Informationsverarbeitungssysteme-Lokale Netzwerke-Teil 2: Logische Verbindungssteuerung" 1989 – 12-31. Demnach kann es wünschenswert sein, die Hardware durch Software zu ergänzen, die einfach bereitgestellt werden kann, damit eine große Puffergröße, ein großer Adreßraum und eine Schnittstelle zu der Netzwerkschicht ermöglicht wird (der Kommunikationstreiber könnte eine 16 Bit logische Adresse für die Schnittstelle zu der Netzwerkschicht benützen, einschließlich einer Knotenadresse in den niederwertigen 8 Bits).

[0144] Das Leistungsvermögen der Architektur **10** kann auf vielfache Weise gemessen werden, einschließlich der Kanalkapazität und Verzögerung, wenn hier die Annahme getroffen wird, daß das Netzwerk normal nach der Konfiguration und Diagnose während des Anlaufens betrieben wird und wenn die Kanalfehler aufgrund des Rauschens oder bei Übertragungen nicht betrachtet werden. Die Basiseinheit der Kommunikation bei der SDLC ist ein Rahmen, der 6 Byte (48 Bit) Overhead aufgrund der Rahmenbildung aufweist (zwei Flagbytes, ein Adressenbyte, ein Steuerbyte und zwei CRC-Bytes). Zusätzlich zu diesem Overhead ist das Biteinfügungs-Null-Bit zu berücksichtigen, das immer dann einzufügen ist, wenn ein Bitstrom von fünf aufeinanderfolgenden logischen Einsen erfaßt wird. Die SDLC-Steuerung benützt auch einen Rahmen für die Bestätigung. Bei der SLC-Steuerung besteht ein Weg zu Erhöhen des Wirkungsgrads in der Erhöhung der Länge einer Meldung. Unter der Annahme, daß eine Meldung durchschnittlich ohne Bestätigung und Biteinfügung 16 Byte lang ist, beträgt die wirksame Kanalnutzung bestenfalls 73%.

[0145] Ein Netzwerkanwender ordnet vorzugsweise einen Teil der gesamten Kanalbandbreite den PCN-Knoten **220** und den Rest den DCN-Knoten **310** zu, um die Anforderungen im Hinblick auf das Aktualisieren der PCN-Knoten **220** zu erfüllen, und/oder die Abtastrate sowie die Meldungsverzögerungszeit der DCN-Knoten **310**. Beim Planen der Zuordnung haben die PCN-Knoten **220** eine höhere Priorität als die DCN-Knoten **310**. Bei der Architektur **10** ist für den Anwender ein Mittel zum Konfigurieren der Zuordnung vorgesehen. Im wesentlichen nützt der CCN-Knoten **210** das Zeitscheibenschema für eine genaue Abtastrate und zum Aufteilen des Kanals.

[0146] Der CCN-Knoten **210** enthält vorzugsweise einen Zeitablauf-Tick-Generator, dessen Periode sich ein-

stellen läßt. Lediglich der CCN-Knoten startet eine Rahmenübertragung zu einem PCN-Knoten **220** bei dem Zeitablauftick. Demnach sollte die Periode eines Ticks so gewählt sein, daß sie die Anforderungen derjenigen PCN-Knoten **220** erfüllt, die die schnellste Abtastrate erfordern. Die Abtastraten der verbleibenden PCN-Knoten **220** sind vorzugsweise äquivalent zu ganzzahligen Vielfachen der ausgewählten Basistickperiode. Es ist möglich, daß mehrere Meldungen der PCN-Knoten **220** innerhalb einer Tickperiode gesendet werden.

[0147] Der CCN-Knoten **210** überwacht vorzugsweise, wieviel Zeit für die Kommunikation mit den DCN-Knoten **310** bis zum nächsten Zeitablauftick verbleibt. In dem Maß, in dem die in der Tickperiode verbleibende Zeit zum Abschließen der Prozeduren zum Abgeben eines Informationsrahmens des DCN-Knotens **310** vor dem nächsten Tick ausreicht, kann der CCN-Knoten **210** mit der Abgabe des Befehls an die DCN-Knoten **310** fortfahren.

[0148] Die Architektur **10** ist vorzugsweise so entworfen, daß ein Anwender das Leistungsvermögen des Netzwerks optimieren kann. Bei einem optimalen Leistungsvermögen solle keine Wartezeit für die geplanten Mitteilungen der PCN-Knoten **220** erforderlich sein. Da die mittlere Zugriffsteuerung für die Kommunikationsvorgänge der DCN-Knoten **310** deterministisch ist, kann der Kanal bis zu seiner Kapazität ausgenützt werden. Die maximale Wartezeit für eine Meldung der DCN-Knoten **310** bleibt konstant, obgleich der Netzwerkbetrieb an seine Kapazitätsgrenze stößt. Bei einem Kommunikationsvorgang des PCN-Knotens **220** dauert es 12,4 µs, um 8 Byte Daten an einen PCN-Knoten **220** abzugeben und um 8 Byte Daten von einem PCN-Knoten **220** zu empfangen, was zu ungefähr 80.000 Aktualisierungsvorgänge pro Sekunde äquivalent ist. Bei einem Kommunikationsvorgang mit einem DCN-Knoten **310** dauert es 32,2 µs, um eine 16 Byte Meldung von einem DCN-Knoten **310** zu einem anderen DCN-Knoten **310** zu übermitteln, was zu ungefähr 30.000 Meldungen pro Sekunde äquivalent ist.

[0149] Die wirksame Kanalkapazität ist geringer als 10 Mbps, und zwar aufgrund der unterschiedlichen Arten der Kommunikations-Overheads. Ein Kommunikations-Overhead bei einem PCN-Knoten **220** ergibt sich lediglich aufgrund der Rahmenbildung. Ein Rahmen besteht aus 48 Bit Overhead und aus bis zu 8 Byte Daten. Die SDLC-Steuerung erfordert auch die Biteinfügung. Unter der Annahme, daß der Datenwert von jeweils 8 Byte **255** ist, müssen 12 Bit eingefügt werden (auch die Adresse und die CRC-Prüfung können eine Einfügung erforderlich machen -- dies ist in der unten gezeigten Implementierung nicht gezeigt). Der Wirkungsgrad für einen PCN-Rahmen beträgt demnach:

$$\frac{8 \times 8}{12 + (6 + 8) \times 8} 100 = 51{,}6\%$$

[0150] Ein vollständiges Übermitteln einer Meldung eines DCN-Knotens **310** erfordert den Zugriffssteuerrahmen, die Schlitzwarteperiode, eine auf Rahmen abgebildete Meldung und den ACK-Rahmen. Da eine Meldung bis zu 33 Byte lang sein kann, wird die Durchschnittszahl von 16 Byte mit 10 Bit Overhead für das Biteinfügen zum Berechnen des Wirkungsgrads benützt. Es können bis zu 31 Knoten vorliegen, so daß von einem durchschnittlichen Prioritätszugriff Overhead von 16 Bit ausgegangen wird.

[0151] Der Zugriffssteuerrahmen enthält 56 Bit: (2Flag-Bits+2-CRC-Bits+1Add-Bit+1-CMD-Bit+1(Daten-Bit) × 8 = 56 Bits. Die durchschnittliche Warteperiode bei einer Störung beträgt 24 Bit: 16 Schlitze × 1,5 Bitzeiten = 24 Bits; die Bytemeldung enthält 194 Bit: (7 + 16) × 8 + 10 = 194 Bits; und der ACK-Rahmen enthält 48 Bit: 6 × 8 = 48 Bits. Demnach beträgt der Wirkungsgrad bei einer Meldung eines DCN-Knotens **310**:

$$\frac{16 \times 8}{56 + 24 + 194 + 48} \times 100 = 39{,}8\%$$

[0152] Der Wirkungsgrad für eine 1 Byte Meldung oder eine 33 Byte Meldung beträgt jeweils 4,2% oder 56%.

[0153] Ein Abtastvorgang, d.h. ein Datenrahmen zu einem PCN-Knoten **220** (124 Bit pro Rahmen) und ein Antwortdatenrahmen von dem PCN-Knoten **220** erfordert **248** Bitzeiten (bei 10 Mbps ungefähr 25 Mikrosekunden). Demnach lassen sich bei Annahme der Tickperiode zu 500 µs ungefähr 5% (25/500) der Kanalkapazität aufgrund der Abtastung des PCN-Knotens **220** nicht nützen. Bei der Kommunikation mit DCN-Knoten **310** erfordert die Übermittlung einer 33 Byte Meldung eines DCN-Knotens **310** 468 Bitzeiten (47 µs bei 10 Mbps). Da die Hardware eine Rahmenübertragung blockiert, wenn eine Restzeit bis zu einem Tick geringer als eine Datenrahmen-Übermittlungsdauer eines DCN-Knotens **310** ist, lassen sich ungefähr 9,4% (47/500) der Kanalkapazität nicht nützen. Demnach wird bei einem Netzwerk mit ausschließlich PCN-Knoten **220** geschätzt, daß

sich ungefähr 50 % der Kanalkapazität für die tatsächliche Übermittlung der Daten nützen lassen. Der Rest stellt Kommunikations-Overhead dar und erzwungene Frei-(Warte)-Zeit. Bei einem kombinierten Netzwerk mit einer Hälfte DCN-Knoten **310** und einer Hälfte PCN-Knoten **220** wird geschätzt, daß sich ungefähr **405** der Kanalkapazität für die Datenübermittlung nutzen lassen.

[0154] Die Berechnung der Meldungswartezeit für die DCN-Knoten **310** erfordert die Kenntnis der Abtastraten sämtlicher PCN-Knoten **220** und des Verkehrsvolumens der Kommunikationsvorgänge der DCN-Knoten **310**. Da ein Anwender den optimalen Betrieb des Netzwerks steuert, wird ein Beispiel zum Schätzen der Kanalnutzung gegeben. Die Tabelle 4 zeigt das Verkehrsvolumen.

Tabelle 4

Zeitperiode, während der ein PCN-Knoten abzutasten ist	PCN-Knoten (Motorknoten-name, M1,M2,...M10)	PCN-Knoten (Hub-Knoten, H1,H2,...H8)	DCN-Knoten (Namen D1,... D2,...,D8)
0,5 ms	(M1, M2)	(H2, H2)	--
1,0 ms	(M3, M4, M5)	(H3, H4)	--
1,5 ms	(M6, M7 ... M10)	(H5, H6)	--
20 ms	--	(H7, H8)	(D1, ..., D8)
Meldungslänge	8 Bytes	8 Bytes	16 Bytes im Durchschnitt
	insgesamt 10 Knoten	insgesamt 8 Knoten	insgesamt 8 Knoten

Insgesamt **26** Knoten im Netzwerk
eine PCN-Abtastung einschließlich Overhead: 124 × 2 = 248 bit
eine DCN-Meldungsübermittlung einschließlich Overhead: 322 bit

[0155] Da die höchste Abtastrate 0,5 ms beträgt, kann der Anwender den Tickwert von 0,5 ms zuordnen. Der Anwender kann ein Scheduling durchführen, wie es in Tabelle 5 gezeigt ist.

Tabelle 5

```
im Zeitpunkt 0.0:   (M1, M2, H1, H2), (M3, M4, M5), (M6, M7), D1
im Zeitpunkt 0.5:   (M1, M2, H1, H2), (H3, H4), (M8, M9), (H7, H8), D2
im Zeitpunkt 1.0:   (M1, M2, H1, H2), (M3, M4, M5), (M10, H5, H6), D3

im Zeitpunkt 1.5:   (M1, M2, H1, H2), (M3, M4, M5), (M6, M7), D4
im Zeitpunkt 2.0:   (M1, M2, H1, H2), (H3,H4), (M8, M9), D5
im Zeitpunkt 2.5:   (M1, M2, H1, H2), (M3, M4, M5), (M10, H5, H6), D6

im Zeitpunkt 3.0:   (M1, M2, H1, H2), (M3, M4, M5), (M6, M7), D7
im Zeitpunkt 3.5:   (M1, M2, H1, H2), (H3,H4), (M8, M9), D8
im Zeitpunkt 4.0:   (M1, M2, H1, H2), (M3, M4, M5), (M10, H5, H6)

im Zeitpunkt 3.0:   (M1, M2, H1, H2), (M3, M4, M5), (M6, M7)
im Zeitpunkt 3.5:   (M1, M2, H1, H2), (H3,H4), (M8, M9)
im Zeitpunkt 4.0:   (M1, M2, H1, H2), (M3, M4, M5), (M10, H5, H6)


im Zeitpunkt 20.0:   (M1, M2, H1, H2), (M3, M4, M5), (M6, M7), D1
im Zeitpunkt 20.5:   (M1, M2, H1, H2), (H3,H4), (M8, M9), (H7, H8), D2
im Zeitpunkt 21.0:   (M1, M2, H1, H2), (M3, M4, M5), (M10, H5, H6), D3
```

[0156] Bei dem obigen Scheduling können bis zu 10 Meldungen der PCN-Knoten **220** (2480 Bit) und eine Meldung eines DCN-Knotens **310** (322 Bit) pro Tickperiode ausgetauscht werden. Unter der Annahme von 10 Mbps weist der Kanal eine Aktivität von ungefähr 56% auf ((2480+322/5000). In diesem Beispiel werden bei der aktivsten Tickperiode von 500 µs immer noch 220 µs nicht eingesetzt. Solange die DCN-Knoten **310** mehr als 65 Meldungen (durchschnittliche Meldungslänge von 16 Byte) pro 0,5 ms erzeugen, tritt bei den DCN-Knoten **310** nahezu keine Wartezeit auf.

[0157] Die Architektur **10** ermöglicht einen flexiblen Einsatz innerhalb eines Systems, was wiederum eine Reihe von Prozeduren zum Einstellen des Netzwerks erfordert, damit Anwenderanforderungen erfüllt werden. Ein Anwender konfiguriert das Netzwerksystem normalerweise während des Systemanlaufs.

[0158] Der CCN-Knoten **210** führt als Netzwerk-Controller vorzugsweise die vollständige Steuerung des Netzwerks aus. Jedoch wird und kann der CCN-Knoten **210** nicht fortlaufend die Netzwerk-Konfiguration überprüfen. Die Architektur **10** ermöglicht einem Anwender nicht, einen Kommunikationsknoten (einen PCN-Knoten **220** oder einen DCN-Knoten **310**) dem Netzwerk nach der Konfiguration des Systems hinzuzufügen. Die Stromversorgung sollte abgeschaltet sein, wenn ein Knoten hinzugefügt oder entfernt wird. Jedoch kann ein Netzwerk mit der Modularitätsoption das Netzwerk während der Stromversorgung rücksetzen.

[0159] Das Netzwerk kann sich in einem von mehreren Zuständen befinden: Netzwerkkonfiguration, PCN-Konfiguration, Diagnosemodus und On-Line-Modus. Die Fig. 33 zeigt Zustandsübergänge. Befindet sich das Netzwerk nicht in dem On-Line-Modus, so befindet es sich in dem Off-Line-Modus. Sämtliche erforderlichen Konfigurationen und Diagnoseschritte müssen dem On-Line-Modus-Betrieb vorausgehen. Echtzeitmeldungen können lediglich während dem On-Line(Normal)-Modus ausgetauscht werden.

[0160] Die Netzwerkkonfigurationen, die vorzugsweise im Off-Line-Modus eingestellt werden, enthalten: Identifizierung des Vorliegens/Nichtvorliegens der Modularitätsoption, Einstellen der Kommunikationsgeschwindigkeit, Identifizierung sämtlicher Knoten an dem Kanal, Laden der Adressen für das Netzwerk mit der Modularitätsoption, Laden der Parameter im Zusammenhang mit den Zugriffsteuerungen für die DCN-Knoten **310** und grundlegende Kommunikationsdiagnose. Sobald das Kommunikationsnetzwerk eingestellt ist, läßt sich die Anwendungshardware der PCN-Knoten **220** konfigurieren, sowie produktbezogene Funktionen, beispielsweise die Boundary-Scan-Logik oder das Softwareladen, soweit es erforderlich ist.

[0161] Da die einzelne bei den Knoten vorgesehene Hardware nicht auf den momentanen Zustand des Netzwerksystems abgestimmt sein kann, kann ein Kommunikationstreiber bei den CCN-Knoten **210** in einfacher Weise vorgesehen sein, der eine Menge von anwenderkonfigurierbaren Parametern ermöglicht, damit ein gleichmäßiger Betrieb des Netzwerks gewährleistet ist.

[0162] Die DCN-Knoten **310** und die PCN-Knoten **220** in dem Netzwerk ohne Modularitätsoption basieren auf der Rücksetzung der Hardware während des Anlaufens. Andererseits kann ein Netzwerk mit der Option vor-

zugsweise dadurch rückgesetzt werden, daß der dritte Draht eingesetzt wird, sowie Software, die in einfacher Weise von dem CCN-Knoten **210** bereitgestellt wird. Die Kommunikations-Hardware der PCN-Knoten **220** wird durch den Netzwerkanlauf rückgesetzt. Die Anwendungshardware der PCN-Knoten **220** wird ebenfalls immer bei Rücksetzung der Kommunikationshardware rückgesetzt. Die Anwendungshardware verbleibt solange rückgesetzt, bis der CCN-Knoten **210** einen Befehl sendet. Die DCN-Knoten **310** weisen keine durch Kommunikation rückzusetzende Anwendungshardware auf.

[0163] Ein Modus beim Anlaufen wird mit seiner geringsten Vorgabegeschwindigkeit betrieben (Vorgabegeschwindigkeit = 10 Mbps/16=0,625 Mpbs, höhere Geschwindigkeiten sind: 1,25 Mbps, 2,5 Mbps, 5 Mbps und 10 Mbps). Der CCN-Knoten **210** kann den Befehl für die Baudrate in jedem Zeitpunkt abgeben. Zum Vermeiden von Störungen wird empfohlen, daß die höhere Geschwindigkeit festgelegt wird, nachdem das System konfiguriert ist, unmittelbar vor dem Eintritt in den Normalmodus. Allgemein sollte die Kommunikationsgeschwindigkeit nicht während dem On-Line-Modus erneut konfiguriert werden. Zum Anheben der höheren Geschwindigkeit sendet der CCN-Knoten **210** den Baudrate-Einstellbefehl. Der CCN-Knoten **210** muß die neue Geschwindigkeit jedem Knoten in dem Netzwerk bestätigen. Für diesen Zweck sollte in dem CCN-Knoten **210** bekannt sein, welche Knoten in dem Kommunikationsnetzwerk vorliegen.

[0164] Zum Konfigurieren des Netzwerks sammelt der CCN-Knoten **210** zunächst Information darüber, welche Knoten in dem Netzwerk vorliegen, d.h. die Knotenidentifikationen. Der CCN-Knoten **210** bestimmt vorzugsweise das Kommunikationsintervall der PCN-Knoten **220** durch Betrachtung der Maximalgrößen der Übertragungs- und Empfangspuffer der PCN-Knoten **220**, in denen Information im Zusammenhang mit der Knotenidentifikation vorliegen sollte. Die Kenntnis der Zahl der DCN-Knoten **310** in dem Netzwerk ist für die wirksame Zugriffsteuerung der DCN-Knoten **310** wesentlich. Konfigurationsprozeduren sind durch die optionale Modularitätsanforderung bestimmt.

[0165] Für ein Netzwerk ohne die Modularitätsoption ist ein Laden der Adressen nicht erforderlich, da die Adresse eines Knotens fest verdrahtet ist. Zum Konfigurieren des Netzwerks für die Zugriffsteuerung bei der DCN-Knoten **310** sammelt der CCN-Knoten **210** die Knotenadressen bei den Knoten **310**. Die Zahl der CCN-Knoten **220** wird benützt, um der Zugriffsteuerungshardware der DCN-Knoten **310** mitzuteilen, wann das selbstinkrementierende Zugriffsteuerregister (ACR) rückzusetzen ist. Bei Untersuchung der Knotenadressen lädt der Master anschließend einen eindeutigen Wert in das Knotenschlitzregister (NSR) jedes DCN-Knotens **310**. Aus Gründen der Wirksamkeit sollte eine Gruppe der in die NSR-Register zu ladenden Zahlen sequentiell geordnet sein, ausgehend von 1. Beispielsweise sei angenommen, daß in den DCN-Knoten **310** vier Adressen gesammelt sind, beispielsweise #5, #8, #26 und #28. Der Master sollte #1, #2, #3, #4, #5 in die NSR-Register laden, da ein Rücksetzen des ACR-Registers erfolgt, wenn es 5 erreicht. Ein Anwender kann eine Zahl aus der Menge jedem beliebigen Knoten zuordnen. Beispielsweise kann #1 in das NSR-Register des Knotens #28 geladen werden. Das Zuordnen oder Löschen eines bestimmten DCN-Knotens **310** in die Zugriffsteuerung mit höchster Priorität muß nicht in dem Off-Line-Modus erfolgen. Tatsächlich weist die Architektur **10** ein Merkmal zum Handhaben dynamischer Situationen auf, die in Echtzeitsystemen auftreten.

[0166] In einem Netzwerk mit der Modularitätsoption kann die Bestimmung der Konfiguration des Systems zum genauen Laden der Knotenadressen ausgewählt werden. Werden Adressen nicht geladen, so ist jede Knotenadresse vorzugsweise eindeutig, und die Konfiguration stimmt mit denjenigen der Netzwerke ohne dieser Option überein. Jedoch kann mit der Option ein Knoten keinen Adressenschalter aufweisen oder Adressen können nicht für identische Module dupliziert werden. Ein derartiges Netzwerk sieht die Bestimmung der Systemkonfiguration und das Laden der Adressen allen anderen Datenkommunikationsvorgängen voraus. Ein Netzwerk läßt sich nicht ohne eindeutig den Knoten zugeordnete Adressen betreiben.

[0167] Ein Knoten empfängt den Master-Senderahmen, der eine Ladeadresse enthält, auf der Grundlage des Eingabezustands bei dem dritten Draht. Der Master sollte die Gesamtprozedur dadurch verifizieren, daß eine Bestätigung Knoten für Knoten erfolgt. Die physikalische Reihenfolge der Verbindung wird zur gleichen Zeit festgelegt, zu der Adressen geladen werden. Die Information im Zusammenhang mit der physikalischen Reihenfolge ist insbesondere für eine Papierhandhabungsanwendung wichtig, bei der identische Module in Serie verbunden sind. Nachdem die Adressen geladen sind, werden die NSR-Register der DCN-Knoten **310** geladen. Der Wert der geladenen Adresse eines Knotens stimmt oft mit dem geladenen Wert des NSR-Registers überein.

[0168] Vorzugsweise in einer Situation, in der der CCN-Knoten **210** einen besstimmten Adressenwert auf der Grundlage der Knoten-Idee zuweist, sammelt der CCN-Knoten **210** sämtliche Knoten-ID-Kennzeichnungen nach dem Laden der Adresse. Anschließend hat die CPU zwei Wahlmöglichkeiten: (1) Rücksetzen des Netz-

werks und erneutes Laden geeigneter Adressen, da der CCN-Knoten **210** nun die Knoten-ID-Kennzeichnungen kennt, oder (2) anstelle einer Netzwerkrücksetzung ordnet der CCN-Knoten **210** erneut die Adressen durch erneute Zuordnung und Vertauschen von Adressen.

[0169] Beim Anlaufen weiß der CCN-Knoten **210**, welche DCN-Knoten **310** und CCN-Knoten **210** in dem Netzwerk vorliegen. Jeder DCN-Knoten **310** kennt vorzugsweise ebenso diese Information. Demnach kann ein Kommunikations-Softwaretreiber, der einfach für CCN-Knoten **210** vorgesehen sein kann, vorzugsweise die Netzwerkkonfiguration für jeden der DCN-Knoten **210** generieren. Bei Empfang des Konfigurationsrahmens behandelt eine Treiber des DCN-Knotens **310** den Rahmen als Verbindungsherstellung zu jedem in dem Rahmen enthaltenen Knoten. Der Wiederholzählwert, der durch einen DCN-Knoten **310** benützt wird, wird von dem CCN-Knoten **210** geladen.

[0170] Ohne den dritten Draht sollte die Hardware der PCN-Knoten **220** beim Anlaufen rückgesetzt werden. Bei dieser Option steuert der CCN-Knoten **210** das Rücksetzen der Anwendungshardware explizit unter Einsatz der beiden rücksetzbezogenen Befehle: "Bestimme das Anwendungshardware-Rücksetzen" und "Lösche das Anwendungshardware-Rücksetzen". Nach dem Rücksetzen der Anwendungshardware kann der CCN-Knoten **210** mit der Konfigurierung der I/O-Anschlüsse der Anwendungshardware fortfahren. Immer dann, wenn die Kommunikationshardware eines PCN-Knotens **220** sich in einem Rücksetzzustand befindet, sollte auch die zugeordnete Anwendungshardware in einen Rücksetzzustand überführt werden. Der dritte Draht setzt den Kommunikationsanschluß eines PCN-Knotens **220** beim Anlaufen zurück. Lediglich in dem Fall, in dem die Kommunikationshardware eines PCN-Knoten **220** den Rücksetzzustand verläßt, kann die Anwendungshardware des PCN-Knotens **220** aus dem Rücksetzzustand herausgeführt werden. Weiterhin kann dies lediglich auf der Grundlage eines "Sperre-PCN-Rücksetzung"-Befehls durch den CCN-Knoten **210** auftreten. Es ist zu erwähnen, daß die Anwendungshardware des PCN-Knotens **220** wieder zwangsweise in den Rücksetzzustand durch einen "Ermögliche-PCN-Rücksetzung"-Befehl von dem CCN-Knoten **210** überführt werden kann.

[0171] Ein PCN-Knoten **220** kann mit konfigurierbarer Hardware ausgestattet sein, um unterschiedliche Anwendungsanforderungen mit einem minimalen Variationsumfang der PCN-Knoten **220** zu erfüllen. Demnach können die PCN-Knoten **220** eine Gruppe intelligenter I/O-Verarbeitungs-Funktionsblöcke im Zusammenhang mit der Bewegungssteuerung aufweisen, sowie programmierbare I/O-Leitungen. Ein Anwender triggert vorzugsweise eine ausgewählte Menge der Funktionsblöcke und programmiert die I/O-Anschlüsse beim Anlaufen. Der PCN-Knoten **220** sollte nicht in den normalen Betriebsmodus übergehen, solange seine Anwendungshardware nicht konfiguriert ist.

[0172] Zum Durchführen der Diagnose bei den PCN-Knoten **220** und den DCN-Knoten **310** sollte das Kommunikationsnetzwerk funktionsfähig sein. Läßt sich das Netzwerk nicht geeignet betreiben, so sollte der CCN-Knoten **210** denjenigen Knoten identifizieren, auf den das Problem zurückzuführen ist. In einigen Fällen kann es erforderlich sein, daß ein Betreiber die Knoten nacheinander entfernen muß, obgleich der Einsatz des dritten Drahts in großem Umfang dieses Erfordernis hinfällig macht. Vorzugsweise sollte ein leicht vorsehbarer Kommunikationstreiber die Diagnosefähigkeit zum Einengen des Bereichs eines Netzwerkproblems oder zum genauen Hinweisen auf den Problembereich aufweisen.

[0173] Für alle PCN-Knoten **220** und DCN-Knoten **310** erfolgt eine Unterstützung zum Miteinbeziehen einer Boundary-Scan-Möglichkeit als Option, die direkt im Zusammenhang mit der Kommunikation steht. Die Boundary-Scan-Möglichkeit vereinfacht das automatische Testen elektrischer Platinen während der Herstellung oder der Produktwartung. Beim Boundary-Scan-Testen erfolgt die Eingabe eines Testbitmusters und die Erzeugung eines Ausgabebitmusters. Diese Muster werden bestimmt, wenn eine elektronische Platine entwickelt wird. Das Boundary-Scan-Testen muß durchgeführt werden, während sich die Anwendungshardware in einem statischen Zustand befindet. Dies erfordert, daß die Verarbeitungseinheit eines DCN-Knotens **310** oder eines PCN-Knotens **210** in einem Wartezustand gehalten wird, und es kann weiterhin erforderlich sein, daß diese oder weitere Anwendungshardware der DCN-Knoten **310** und der PCN-Knoten **210** in einem Rücksetzzustand gehalten werden.

[0174] Beim Eintreten in den On-Line-Modus sollten alle zuvor erwähnten Schritte abgeschlossen sein. Im On-Line-Modus beginnt der CCN-Knoten **210** mit der Generierung des SYNC-Rahmens und der DAC-Rahmen. Im Off-Line-Modus liegen diese Rahmen nicht vor. Der CCN-Knoten **210** erfaßt die.

[0175] Modusveränderung. Jedoch müssen PCN-Knoten **220** oder DCN-Knoten **310** die Modusveränderung nicht erfassen, und sie können einfach einen festgelegten Befehlsrahmen in jedem Zeitpunkt von dem

CCN-Knoten **210** empfangen, diesen verarbeiten, und einen festgelegten Antwortrahmen entweder im On-Line-Modus oder im Off-Line-Modus erzeugen. Für die Kommunikation mit den PCN-Knoten **220** werden die Melde-Folgebits nicht benützt. Sobald der CCN-Knoten **210** einen Rahmen mit einem PCN-Knoten **220** erfolgreich austauscht, ist eine Kommunikationsverbindung hergestellt. Da die Kommunikation mit den DCN-Knoten **310** Folgebits benützt, die zu Null während des Anlaufens des Netzwerks initialisiert sind, ist eine Kommunikationsverbindung zwischen irgendeinem Paar der DCN-Knoten **310** oder zwischen dem CCN-Knoten **210** und jedem beliebigen DCN-Knoten **310** hergestellt, wenn ein Rahmen erfolgreich ausgetauscht wird. Nach dem Senden des ersten Rahmens durch einen übertragenen Knoten sollte dieser den Beschädigungsrahmen erfolgreich empfangen. Hierdurch wird eine Einweg-Kommunikationsverbindung hergestellt. Ein DCN-Knoten **310** muß nicht die Verbindung zu jedem DCN-Knoten **310** herstellen. Der Kommunikationstreiber eines CCN-Knotens **210** ist für die Herstellung der logischen Verbindung verantwortlich. Schlägt die Kommunikation zwischen einem DCN-Knoten **310** und einem anderen DCN-Knoten **310** fehl, so sollte das gesamte Netzwerk erneut gestartet werden, um die Folgebits der Knoten erneut zu initialisieren. Alternativ sollte nach der Korrektur des Problems die betroffene(n) Verbindung(en) erneut durch Senden von zwei Leermeldungen in jede Richtung hergestellt werden.

[0176] Während einige Funktionsmerkmale der Netzwerke unter Einsatz der Architektur **10** mit Hilfe von Hardware realisierbar sind, werden andere am besten durch Software implementiert, die sich leicht bereitstellen läßt. Beispielsweise kann ein Kommunikations-Softwaretreiber für den CCN-Knoten **210** die Schnittstelle zur Hardware des CCN-Knotens **210** vereinfachen. Entsprechend können Softwaretreiber für die Kommunikation der DCN-Knoten **310** die begrenzten Puffergrößen der Hardware des CCN-Knotens **210** und der DCN-Knoten **310** erweitern. Zusätzlich kann ein Satz Softwaremodule, die beispielsweise in der Programmiersprache C sowohl für den CCN-Knoten **210** als auch die DCN-Knoten **310** beschrieben sind, Programmcodes laden, als Spezialfall eines Filetransfer. Der Ladevorgang kann drei Arten von Speichervorrichtungen unterstützen: Speicher mit wahlfreiem Zugriff (RAM-Speicher), elektrisch löschbare Nurlesespeicher (EEPROM-Speicher) und FLASH-Nurlesespeicher (FLASH-ROM-Speicher), und jeder Typ kann eigene Zeitablaufanforderungen mit sich bringen. Eine Gruppe von Routinen in dem CCN-Knoten **210** und dem DCN-Knoten **310** kann einen Filetransfer in ASCII und Binärformat ermöglichen.

[0177] Obgleich das Protokoll nicht den direkten Austausch zwischen DCN-Knoten **310** und PCN-Knoten **220** aufgrund der begrenzten Möglichkeiten der PCN-Knoten **220** ermöglicht, die keine CPU enthalten, könnte Sofrware auf Anwendungsebene, die für den CCN-Knoten **210** einfach zur Verfügung gestellt werden kann, Information zwischen PCN-Knoten **220** und DCN-Knoten **310** übertragen. Besonders vorteilhaft könnten während des Anlaufens bei dem CCN-Knoten **210** Anforderungen von DCN-Knoten **310** gemäß spezifischer Sensorinformationen geladen werden. Weiterhin können PCN-Knoten **220** in der Lage sein, dem CCN-Knoten **210** mitzuteilen, wann ein signifikantes Ereignis aufgetreten ist. Beim Erfassen der Tatsache, daß ein signifikantes Sensorereignis aufgetreten ist, könnte der CCN-Knoten **210** überprüfen, ob diese Information durch DCN-Knoten **310** angefordert wurde. Ist dies der Fall, so könnte er eine Meldung an die DCN-Knoten **310** mit der neuen Sensorinformation senden. Ferner könnten DCN-Knoten **310** eine Meldung an den CCN-Knoten **210** senden, mit der Anforderung, die Steuerung eines bestimmten PCN-Knotens **220** zu modifizieren. Hierbei würde die maximale Verzögerung, bis eine Sensorinformation für DCN-Knoten **310** einen DCN-Knoten **310** erreicht oder bis eine Anforderung eines DCN-Knotens **310** für die Ausgabe einer Veränderung eines PCN-Knotens **220** (Veränderung) erreicht, weniger als 5 ms betragen.

[0178] Bei der Architektur **10** kann die Systemhardware für PCN-Knoten **220** bis auf den Kommunikationsanschluß üblicherweise als Anwendungshardware bezeichnet werden. Typische Funktionen, die durch die Anwendungshardware unterstützt werden; sind wie folgt vorgesehen:
Abschalten der Stromversorgungstreiber: der Überwachungsdaten-Zeitgeber des Kommunikationskerns erzeugt ein Signal für die Anwendungshardware, wenn der PCN-Knoten **220** nicht innerhalb einer festgelegten Zeit aktualisiert wird. Anschließend sollte die Anwendungshardware alle PWM-Vorrichtungen abschalten, sowie das Sensorabtasten und einfache Ausgänge in den Tri-State-Zustand versetzen. Demnach verbleiben Motoren und Solenoide nicht in einem angelaufenen Zustand, wenn das System hängt.

[0179] Motor/Solenoid-Controller-PWM-Generator: ein Motorsteuerungs-PWM-Generator wird unter Einsatz einer 8 Bit Tastverhältniszyklussteuerung und eines Richtungssteuerungsbit erzeugt. Der PWM-Generator sollte eine Sperrfunktion aufweisen, die die PWM-Ausgang zum Begrenzen der Motorwicklungsströme abschaltet. Der PWM-Ausgang sollte vorzugsweise lediglich erneut während der nächsten Anfahrperiode angeschaltet werden. Zusätzlich ist ein Anhaltschutz in dieser Funktion mit einzubeziehen, um die Feldeffekttransistor-(FET)-Treiber zu schützen, da diese allmählich Fehler aufweisen, wenn sie länger als 10 bis 20 ms angeschaltet sind. Dies kann dadurch erreicht werden, daß gezählt wird, wie lange der PWM-Tastzyklus größer als

ein Schwellwert (TBD) ist. Übersteigt das PWM diesen Schwellwert, dann sollte der PWM-Ausgang gesperrt und ein durch den CCN-Knoten **210** lesbares Flag gesetzt werden. Der PWM-Generator sollte erneut freigegeben werden, wenn der PWM-Befehl des CCN-Knotens **210** auf einen Nullwert rückgeführt wird.

[0180] D/A-Funktion: Eine Digital/Analog- oder D/A-Funktion kann durch Einsatz eines einfachen 8 Bit PWM und eines Integrators bereitgestellt werden.

[0181] A/D-Funktion: Eine Analog/Digital- oder A/D-Funktion unterstützt die Sensorabtastfunktion. Die A/D-Auflösung muß 8 Bit nicht überschreiten und die Umsetzgeschwindigkeit ist nicht kritisch. Die A/D-Funktion läßt sich kostengünstig durch Einsatz der D/A-Funktion und einer Logik als programmierbarer Logikvorrichtung (PLD) zum Steuern aufeinanderfolgender Approximationsschritte unter Einsatz eines Binärzugverfahrens implementieren. Bei dieser Vorgehensweise ist ein einfacher Komparator außerhalb der PLD-Vorrichtung erforderlich. Die PLD-Vorrichtung bestimmt eine Referenzspannung unter Einsatz der D/A-Funktion für die Durchführung eines Vergleichs mit der analogen Eingangsspannung und führt eine Entscheidung auf der Grundlage der Tatsache durch, ob die Spannung niedriger als oder größer als die Analogspannung ist.

[0182] Sensorabtastung: Die Fähigkeit zum Abtasten zahlreichen Arten von Sensoren enthält mindestens das Abtasten von: optoreflektierenden Sensoren (mit oder ohne Fensterabschirmung und Umlicht), Hall-Schalter (digital und analog), optodurchlässige Sensoren (lediglich digital), Tastaturschalter und Lückendetektion (detektiere Lücke und setze Flag, der CCN-Knoten **210** setzt das Flag explizit zurück).

[0183] Bitprogrammierbare I/O-Vorrichtung: Es sollten zumindest 8 Bit konfigurierbarer/O-Elemente vorgesehen sein, ausgehend in einem Tri-State-Zustand beim Anlaufen.

[0184] Während die meisten Knotenkommunikationsfunktionen oben beschrieben wurden, existieren zusätzliche Merkmale, beispielsweise eine Fehlerbehebung und Anforderungen, die im Zusammenhang mit dem Knotentypus spezifisch sind. Allen Knoten gemeinsam ist die Fähigkeit, einen Rahmen zu empfangen, einen zu erzeugen, und CRC-Fehler und Rahmenfehler zu erfassen.

[0185] Merkmale der Architektur **10** im Zusammenhang mit PCN-Knoten **220** betreffen Kommunikationsanforderungen und Anwendungshardware. Im Hinblick auf die Kommunikationsanforderungen ist es für PCN-Knoten **220** nützlich, einen Überwachungsdatenzeitgeber vorzusehen. Der Überwachungsdatenzeitgeber muß ein Signal für die Anwendungshardware erzeugen, wenn der PCN-Knoten **220** nicht innerhalb einer festgelegten Zeitdauer aktualisiert wird. Der Überwachungsdatenzeitgeber beginnt zu laufen, wenn der erste SYNC-Rahmen empfangen wird. Es ist zu erwähnen, daß der CCN-Knoten **210** in der Lage sein sollte, das 3 Byte. Knotenidentifikationsregister zu lesen.

[0186] Ein PCN-Knoten **220** unterstützt auch den Loop-Back-Test, in dem der CCN-Knoten **210** einen Befehl an einen PCN-Knoten **220** zum Wiederholen des empfangenen Rahmens sendet. Im Loop-Back-Modus empfängt der PCN-Knoten **220** eine Meldung, überträgt die Meldung an seinen Übertragungspuffer und überträgt sie unmittelbar an den CCN-Knoten **210** zurück der CCN-Knoten **210** weist dann den PCN-Knoten **220** an, den Loop-Back-Modus zu verlassen.

[0187] Weiterhin sollte der PCN-Knoten **220** auch dann funktionsfähig bleiben, wenn bei ihm Rahmenfehler und CRC-Fehler auftreten. Er ignoriert Rahmen, die in dem Steuerfeld nicht vordefiniert sind.

[0188] Die Kommunikationshardware erzeugt ein Signal für die Anwendungshardware zum Initiieren des Empfangs SYNC-Rahmens. Die über die Kommunikation empfangenen Ausgangsdaten können an die Ausgangsleitungen unmittelbar weitergeleitet werden, oder sie können mit dem SYNC-Rahmen synchronisiert werden. Die Wahl ist konfigurierbar.

[0189] Das Abtasten der Eingabeleitung läßt sich mit dem SYNC-Rahmen synchronisieren, oder sie kann unmittelbar durchgeführt werden. Die Auswahl ist ebenfalls konfigurierbar. Die Kommunikationshardware überträgt lediglich fehlerfreie Meldungen an die Anwendungshardware. Der Kommunikationskern eines PCN-Knotens **220** muß typischerweise keinen Pufferspeicher enthalten, der anders als das serielle Schieberegister aufgebaut ist.

[0190] Merkmale im Zusammenhang mit DCN-Knoten **310** umfassen die CPU-Schnittstelle und kommunikationsbezogene Spezifikationen. Besonders bevorzugte Merkmale im Zusammenhang mit Schnittstellen der CPU der DCN-Knoten **310** und der Register sind in Tabelle 5 aufgelistet.

Tabelle 5

Knoten-ID-Register
3 Byte Knotenidentifikationsregister

Knotenadressenregister (NAR)
Dieses Register wird mit den unteren 5 Bit der Knoten-ID beim Anlaufen geladen. Der DCN-Knoten 310 kann in dieses Register schreiben, und der DCN-Knoten 310 kann es lesen. Die Kommunikationshardware benützt diesen Wert für die Adressenerkennung und als Quelladresse.

Knotenschlitzregister (NSR)
Dieses Register, das für die Zugriffsteuerung vorgesehen ist, wird mit den unteren 5 Bit der Knoten-ID beim Anlaufen geladen. Der CCN-Knoten 210 schreibt in dieses Register, und der DCN-Knoten 310 kann es lesen.

Prioritätsflag
Durch den CCN-Knoten 210 geladen und vorgabegemäß rückgesetzt. Ist es gesetzt, so nützt ein DCN-Knoten 310 den Prioritätsschlitz für eine Rahmenübertragung.

Interrupterzeugung
Ein Interrupt ist bei den folgenden Ereignissen zu erzeugen: (a) Empfangspuffer fertig, (b) Übertragungspuffer gelöscht, (c) Wiederholungsgrenze erreicht, (d) maximaler Fehlerzählwert und (c) SYNC-Rahmen empfangen.

Empfangsbezogen
Empfangsflag
Der Kommunikations-Controller setzt dieses Flag, wenn ein fehlerfreier Informationsrahmen empfangen wird. Der CCN-Knoten 210 setzt es zurück.
Empfangspuffer-Byte-Zählwert
Zahl der Datenbytes ausschließlich irgendeiner Adresse
Rahmenquellregister
Die Knotenadresse eines Rahmenursprungs
Rahmensteuerfeld-Leseregister
Empfangenes Rahmensteuerfeld
Empfangspuffer
33 Byte lang.

Übertragungsbezogen
Übertragungsflag
Der CCN-Knoten 210 setzt dieses Flag, löscht es anschließend, nachdem der Puffer erfolgreich

übertragen wurde. (Für den Boundary-Scan-Rahmen löscht der Kommunikations-Controller das Flag, nachdem er den Rahmen verarbeitet hat. Dies erfolgt schnell und vor der Herstellung der Verbindung)

Übertragungspuffer-Byte-Zählwert
: Zahl der Datenbytes ausschließlich irgendeiner Adresse

Rahmenbestimmungsadresse
: Die Rahmenbestimmungsadresse

Rahmensteuerfeld-Schreibregister
: Das zu übertragende Rahmensteuerfeld

Übertragungspuffer
: 33 Byte lang.

Übertragungswiederholzähler
: Dieser Zähler zählt bis zu 15 und muß durch PCN-Knoten 310 und den CCN-Knoten 210 lesbar sein. Erreicht er das Maximum, so verbleibt er bei diesem Wert. Der Controller löscht den Computer, wenn die Übertragung erfolgreich verläuft. Ein durch einen DCN-Knoten 310 erzeugte Rahmen wird automatisch erneut (im Fall eines Fehlers) durch die Kommunikations-Hardware bis zu 15 mal übertragen, mit zumindest einem Intervall und bei einem günstigen Zeitpunkt, der durch die Zugriffsteuerung spezifiziert ist. Der Zählwert 0 zeigt keinen Fehler an.

Fehlerflag
: Der CCN-Knoten 310 sowie DCN-Knoten 310 müssen in der Lage sein, zumindest das folgende zu lesen: CRC-Fehlerflag, Rahmenfehler und Zeitablauffehler bei dem erwarteten ACK-Rahmen

32 Paare der Laufbits
: Es gibt ein Übertragungsfolgebit und ein Empfangsfolgebit für jeden DCN-Knoten 310. Beim Anlaufen sind alle Bits 0 und ein DCN-Knoten 310 ist in der Lage, die Bits zu lesen (Nurlesevermögen). Der Controller muß auch in der Lage sein, Meldungen zu erfassen, die die Laufzahl anfordern, und die Zahl zu aktualisieren. Nachfolgend ist das benützte Indizierschema gezeigt. Es ist zu erkennen, daß ein Paar gemäß seinem zugeordneten Knoten für einen Netzwerk-Loop-Back-Test benützt werden kann.

Index	Übertragungs-Laufbit	Empfangs-Laufbit	Anmerkungen
0	--	--	für CCN
1	--	--	für DCN 1
2	--	--	für DCN 1
:	--	--	:
:	--	--	:
29	--	--	:
30	--	--	für DCN 30
31	--	--	für DCN 31

[0191] Im Hinblick auf die kommunikationsbezogenen Merkmale sollte der CCN-Knoten **210** in der Lage sein, das 3 Byte Knotenidentifikationsregister zu lesen sowie das Knotenschlitzregister, das Prioritätsflag und das SYNC-Flag. Erfaßt ein DCN-Knoten **310** einen SYNC-Rahmen, so wird ein Impuls erzeugt und ein Flag gesetzt, und der DCN-Knoten **310** kann das Flag löschen. Weiterhin unterstützt der DCN-Knoten **310** den Loop-Back-Test, entsprechend der Funktion, die im Zusammenhang mit den PCN-Knoten **220** beschrieben ist.

[0192] Der DCN-Knoten **310** sollte auch dann funktionieren, wenn Rahmenfehler oder CRC-Fehler auftreten. Er ignoriert nicht vordefinierte Rahmen in dem Steuerfeld. Zusätzlich wird der DCN-Knoten **310** die Kommunikationsfähigkeit der Architektur 1 über die oben definierten Register ausnützen. Ist das Empfangpufferflag gesetzt, so antwortet der DCN-Knoten **310** mit dem NAK-Rahmen auf einen einkommenden Informationsrahmen. Der DCN-Knoten **310** löscht das Empfangsflag, und die Kommunikationshardware löscht das Übertragungsflag. In ähnlicher Weise setzt der DCN-Knoten **310** das Übertragungsflag, und die Kommunikationshardware setzt das Empfangsflag. Ein Interruptsignal wird erzeugt, wenn (a) eine neue Meldung empfangen wird und (b) der Übertragungspuffer bereit ist.

[0193] Ein durch einen DCN-Knoten **310** initiierter Rahmen sollte erneut übertragen werden, wenn ein ACK-Rahmen nicht innerhalb einer festgelegten Zeit empfangen wird. Der Prozeß wird bis zu 15 mal wiederholt, und die Intervalldauer für die erneute Übertragung sollte 1 ms betragen. Schlägt eine erneute Übertragung fehl, so muß ein DCN-Knoten **310** das Fehlerflag setzen. Der CCN-Knoten **210** scannt auch die DCN-Knoten **310** in regulärer Weise, und fährt das Netzwerk nach unten, wenn in irgendeinem der DCN-Knoten **310** das Flag gesetzt ist. Wird bei einer Informationsübertragung bei einem DCN-Knoten **310** der NAK-Rahmen empfangen, dann überträgt der DCN-Knoten **310** erneut immer wieder jede Millisekunde (konfigurierbar), bis er den ACK-Rahmen empfängt.

[0194] Die Merkmale der Architektur 10 im Zusammenhang mit dem CCN-Knoten **210** schließen die CPU-Schnittstellenbildung ein, sowie die Register zum Steuern des Netzwerks, die Puffer zum Steuern der Rahmen, die Schnittstellenbildung mit den PCN-Knoten **220** und die Hardwaresteuerung.

[0195] Die Schnittstelle zwischen den Kommunikations-Controller des CCN-Knotens **210** und der CPU des CCN-Knotens **210** erfolgt vorzugsweise primär über einen gemeinsamen Speicher. Demnach werden Register und Speicher wechselweise in diesem Abschnitt eingesetzt. Obgleich Details festzulegen sind, sind die erwarteten Schnittstellensignale: RESET, ADO-AD15, ALE, WAIT, CS, WR, RD, INT, CLK, VCC und GND. Ferner kann der CCN-Knoten **210** Interrupts einzeln oder als Ganzes freigeben/sperren. Allgemein werden Interrupts bei (1) Empfang des bereiten Puffers erzeugt, sowie (2) bei verfügbarem Übertragungspuffer, (3) einem signifikanten Ereignis in einem PCN-Knoten **220**, (4) einem Kommunikationsfehler und (5) einem Zugriff der CPU des CCN-Knotens **210** auf den Puffer des PCN-Knotens **220**, der momentan von der Kommunikationshardware benutzt wird. Die CPU des CCN-Knotens **210** löscht das durch den Kommunikations-Controller des CCN-Knotens **210** gesetzte Interruptflag.

[0196] Details im Hinblick auf die Register zum Steuern des Netzwerks sind in der Tabelle 6 aufgelistet. Es ist zu erwähnen, daß alle Register in Tabelle 6 vom Typ zum Lesen/Schreiben sind, soweit es nicht anders angegeben ist.

Tabelle 6

Knoten-ID-Register
3 Byte Knotenidentifikationsregister.

Baud-Raten-Register
Die CPU des CCN-Knotens 210 kann einen Code für den Kommunikations-Controller für den Einsatz einer der fünf definierten Raten setzen: 0=0,625 Mbps, 1=1,25 Mbps, 2=2,5 Mbps, 3=5 Mbps, und 4=10 Mbps.

PCN-Übertragungsintervall
Ein Rahmenübertragungsintervall eines PCN-Knotens 220 im Normalmodus-Betrieb.

Tickperioden-Register
Die CPU des CCN-Knotens 210 kann die Tickperiode im Hinblick auf die Zahl der Kommunikations-Controller-Takte festlegen. Es handelt sich um einen der wichtigsten Parameter, die ein Anwender benützen kann, um ein Kommunikations-Scheduling bei Einsatz der PCN-Knoten 220 zu bestimmen.

Modularitätsoptions-Anzeige
Dieses Bit ist gesetzt, wenn der Kommunikations-Controller das Vorliegen der Fähigkeit gemäß der optionalen Modularität erfaßt; andernfalls ist es gelöscht. Sein Vorliegen wird dadurch angezeigt, daß der Eingang des dritten Drahts beim Anlaufen auf L-Pegel liegt.

Register für die Zahl der PCN-Knoten (PCNR)
Der CCN-Knoten 210 schreibt in dieses Register, nachdem er die Zahl der PCN-Knoten 220 in dem Netzwerk bestimmt hat. Der Kommunikations-Controller kann diese Information nutzen.

Register für die Zahl der DCN-Knoten (DCNR)
Der CCN-Knoten 210 schreibt in diesen Speicher, nachdem er die Zahl der DCN-Knoten 310 in dem Netzwerk bestimmt hat. Der Kommunikations-Controller kann diese Information benützen.

Knotenadreßregister (NAR)
Dieses Register wird mit den vier unteren Bits der Knoten-ID beim Anlaufen geladen. Der CCN-Knoten 210 kann in dieses Register schreiben, und ein DCN-Knoten 310 sollte in der Lage sein, es zu lesen. Die Kommunikationshardware nützt diesen Wert für die Adressenerkennung und als Quelladresse.

Knotenschlitzregister (NSR)
Dieses Register wird mit den unteren 5 Bit der Knoten-Idee beim Anlaufen geladen. Der CCN-Knoten 210 schreibt in dieses Register, und ein DCN-Knoten 310 sollte in der Lage sein, es zu lesen. Hierdurch wird eine Zugriffsteuerung ermöglicht.

Prioritätsflag
Dieses wird durch den CCN-Knoten 210 geladen und es ist vorgabegemäß gelöscht. Ist es gesetzt, so benützt ein DCN-Knoten 310 den Prioritätsschlitz für eine Rahmenübertragung.

Zugriffsteuerungsregister (ACR)
Dies ist ein Inkrementierzähler, und es erfolgt eine Rücksetzung zu 0, wenn er den Wert in dem DCN-Knoten erreicht (Zahl der DCN-Knoten 310). Der Wert des ACR-Registers wird beim Aufnehmen eines DAC-Rahmens benützt. Das ACR-Register wird nur dann inkrementiert, wenn der CCN-Knoten 210 einen Rahmen von einem DCN-Knoten 310 unmittelbar nach dem DAC-Rahmen erfaßt.

Fehlerzähler
Immer wenn ein Kommunikationsfehler erfaßt wird, wird er inkrementiert. Dieser Zähler kann durch die CPU gelöscht werden.

[0197] Details im Hinblick auf die Puffer für die Steuerrahmen und ein DCN-Knoten **310** sind in Tabelle 7 aufgelistet.

Tabelle 7

Empfangsbezogen
- Empfangsflag
  - Der Kommunikations-Controller setzt dieses Flag, wenn ein fehlerfreier Informationsrahmen empfangen wird. Die CPU löscht es.
- Empfangspuffer-Byte-Zähler
  - Die Zahl der Datenbytes ausschließlich irgendeiner Adresse.
- Rahmenquellregister
  - Die Knotenadresse eines Rahmenursprungs
- Rahmensteuerfeld-Leseregister
  - Dies ist das Steuerfeld des empfangenen Rahmens
- Empfangspuffer
  - Dieses ist 33 Byte lang.

Übertragungsbezogen
- Übertragungsflag
  - Die CPU setzt dieses Flag. Anschließend löscht der Controller es, nachdem er den Puffer erfolgreich übertragen hat.
- Übertragungspuffer-Bytezähler
  - Zahl der Datenbytes ausschließlich irgendeiner Adresse
- Rahmenbestimmungsadresse
  - Die Rahmenbestimmungsadresse
- Rahmensteuerfeld-Schreibregister
  - Zu übertragendes Rahmensteuerfeld
- Übertragungspuffer
  - Dieses ist 33 Byte lang.

Zähler für erneute Übertragung

Dieser Zähler zählt bis zu 15 und muß durch DCN-Knoten 310 und den CCN-Knoten 210 lesbar sein. Erreicht er das Maximum, so bleibt er auf diesem Wert stehen. Der Controller löscht ihn, wenn die Übertragung erfolgreich verlaufen ist. Ein DCN-gebundener Informationsrahmen wird automatisch erneut übertragen, (im Fall eines Fehlers) durch die Kommunikationshardware, und zwar bis zu 15 mal, mit zumindest einem Intervall von 1 ms bei einem günstigen Zeitpunkt, der für die Zugriffsteuerung

spezifiziert ist. Ein Zählerwert von 0 zeigt einen fehlerfreien Zustand an.

Fehlerflags
Der CCN-Knoten 210 sowie die DCN-Knoten 310 müssen in der Lage sein, die folgenden Größen zu lesen: CRC-Fehlerflag, Rahmenfehler und Zeitablauffehler für den erwarteten ACK-Rahmen.

32 Paare von Laufbits
Es wird ein Übertragungslaufbit und ein Empfangslaufbit für jeden DCN-Knoten 310 übertragen. Beim Anlaufen sind alle Bits 0. Ein DCN-Knoten 310 muß in der Lage sein, die Bits zu lesen (Nurlesevorgang). Der Controller muß in der Lage sein, Meldungen zu erfassen, die die Laufzahl anfordern, und diese Zahl zu aktualisieren. Nachfolgend ist das verwendete Initiierschema gezeigt.

Index	Übertragungs-Laufbit	Empfangs-Laufbit	Anmerkungen
0	--	--	nicht verwendet
1	--	--	für DCN 1
2	--	--	für DCN 1
:	--	--	:
:	--	--	:
29	--	--	:
30	--	--	für DCN 30
31	--	--	für DCN 31

[0198] Im Hinblick auf die Schnittstellenregister für PCN-Knoten **220** sollte der CCN-Knoten **210** vier "signifikante PCN-Ereignisse"-Register unterhalten; bei 8 PCN-Knoten **220** pro 8 Bit Register.

[0199] Weiterhin sollten vier "signifikante Ereignismaskierungs"-Register vorgesehen sein; 8 PCN-Knoten **220** pro 8 Bit Register. Diese Registerbits ermöglichen/sperren einen "signifikanten PCN-Interrupt" bei dem CCN-Knoten **210**.

[0200] Tritt bei einem der PCN-Knoten **220**, dessen Maske für ein signifikantes Ereignis freigegeben ist, ein signifikantes Ereignis auf, so sollte anschließend ein Interrupt an den CCN-Knoten **210** abgegeben werden. Weiterhin sollte ein Interruptvektor für ein signifikantes Ereignis erzeugt werden, um die Auslösung einer geeigneten Maßnahme durch den CCN-Knoten **210** zu beschleunigen. Tritt weiterhin mehr als ein freigegebenes signifikantes Ereignis bei einem der PCN-Knoten **220** auf, so wird der CCN-Knoten **210**, nachdem er den ersten Vektor zwischendurch gehandhabt hat, erneut mit dem neuen unterbrochen, bis alle signifikanten Ereignisse bearbeitet sind.

[0201] Der CCN-Knoten **210** setzt auch ein Scan-Flag zum Triggern des Scannens der Puffertabelle. Beim Rücksetzen des Controllers wird es gelöscht. Der Controller untersucht jede Einheit, und ist das Tick-Register bis zu 0 herabgezählt, so überträgt der Controller den Puffer und empfängt einen Rahmen. Empfängt der Controller keinen Rahmen, so setzt er das Fehlerflag. Der Prozeß wird für alle Tabelleneinträge fortgesetzt. Am Ende eines Scan-Vorgangs und vor dem Überleiten sollte der CCN-Knoten **210** in der Lage sein, die verbleibende Zeit innerhalb einer Tickperiode zu bestimmen. Verbleibt Zeit für eine Kommunikation eines DCN-Knotens **310**, so gibt er den DAC-Rahmen ab. Die DAC-Rahmen können bis zum nächsten Ticksignal wiederholt werden. Liegt beim CCN-Knoten **210** eine eigene Meldung für DCN-Knoten **310** vor, so überträgt er die Meldung, bevor er den Befehl abgibt.

[0202] Der Controller schreibt den Index der Puffertabelle in das Scan-Indexregister, um dem PCN-Knoten **220** anzuzeigen, daß er kommuniziert. Vorzugsweise könnte Software, die sich einfach für den CCN-Knoten **210** bereitstellen läßt, durch Lesen dieses Registers und durch die Übertragungs- und Empfangsflags den Puffer feststellen, den der Controller momentan benützt. Mit Ausnahme dieses Puffers sollte der CCN-Knoten **210**

in der Lage sein, sämtliche Puffer zu lesen/zu beschreiben. Jedoch sollte der CCN-Knoten **210** nicht auf einen in einen Kommunikationsvorgang einbezogenen Puffer zugreifen. Dies wird ausschließlich mit Software gehandhabt. Versucht der CCN-Knoten **210** auf den aktiven Puffer zuzugreifen, so wird der Zugriffsversuch fehlschlagen, und ein Interrupt wird erzeugt. Dieser ist als Fehlersuchhilfe gedacht, da in einem endgültigen System der DCN-Knoten **310** niemals auf einen aktiven Puffer zugreifen sollte. Es ist zu erwähnen, daß zwei Möglichkeiten bestehen: erstens versucht der CCN-Knoten **210** auf einen bereits aktiven Puffer zuzugreifen, und zweitens greift der CCN-Knoten auf einen Puffer zu, der aktiv wird. Die Kommunikationshardware wird die Puffer planmäßig senden, und wird diese Störung nicht unterbrechen.

[0203] Die Kommunikations-Controllerhardware des CCN-Knotens **210** bietet viele Möglichkeiten. In dem CCN-Knoten sollte auch ein Kommunikations-Software-Treiber vorliegen, der einfach vorgesehen werden kann, um die Hardwaremöglichkeiten vollständig zu nützen. Der Treiber könnte auch zusätzliche Logikfunktionen für die logischen Verbindungssteuerungen ermöglichen. Vorzugsweise besteht die vorrangige Aufgabe des Treibers in der Konfigurierung des Netzwerks. Soll die Modularitätsoption benützt werden, so geht die Bestimmung der Produktkonfiguration gemäß der physikalischen Reihenfolge mit dem Verbinden und dem Laden der Adressen sämtlichen anderen Befehlen voraus. Der Treiber könnte sämtliche Fehler beheben.

[0204] Der nächste Schritt besteht im Zusammenfassen aller Knoten-ID-Kennzeichnungen, Knotenadressen, der Zahl der PCN-Knoten **220** und der Zahl der DCN-Knoten **310**. Die PCNR-Register und die DCNR-Register sollten in zugeordneter Weise geladen werden. Anschließend sollten geeignete Werte in die Schlitzregister der DCN-Knoten **310** geladen werden, und falls erforderlich, sollte einem bestimmten DCN-Knoten **310** die oberste Priorität für die Zugriffsteuerungen der DCN-Knoten **310** zugeordnet werden. Hierdurch wird die Netzwerkkonfiguration abgeschlossen. Wenn es gewünscht wird, können die DCN-Knoten **310** einer nach dem anderen unter Einsatz der Boundary-Scan-Logikrahmen untersucht werden.

[0205] Nach dem Konfigurieren und Testen des Netzwerks kann die Anwendungshardware eines PCN-Knotens **220** aus dem Rücksetzzustand freigegeben werden. Auch die Anwendungshardware eines PCN-Knotens **220** sollte konfiguriert werden. In diesem Zeitpunkt ist das System für den On-Line-Betrieb bereit. Der Treiber erfaßt alle Fehler, die die zu einem Außerbetriebsetzen des Netzwerks führen können, wenn die Zahl der Fehlerereignisse überhand nimmt.

[0206] Zum Überwachen des Kommunikationsbusses **230** ist eine Testschaltung in einer feldprogrammierbaren Gate-Array-Schaltung (FPGA)-Schaltung oder in einer anderen Form einer Elektronik implementiert, und sie weist vorzugsweise die Fähigkeit auf, den Baud-Rate-Befehl zu erkennen und die Rate geeignet festzulegen. Weiterhin sollte die Testschaltung die Fähigkeit aufweisen, die Start- und Endflags zu erkennen und all die Bytes zwischen den beiden Flags aufzunehmen. Die Testschaltung sollte ferner eine Schnittstelle zu dem PC-Bus aufweisen und einen Zeitgeber zur Verfügung stellen, der durch den Anwender in dem Bereich zwischen einer Mikrosekunde bis zu einigen Stunden eingestellt werden kann. Die Hardware sollte auch in der Lage sein, sämtliche Rahmen aufzunehmen. Eine Möglichkeit, um dies zu erreichen, besteht darin, 64-Byte-Puffer pro Rahmen einzusetzen, und den Puffer-Zeiger durch die Hardware zu inkrementieren.

[0207] Zum Testen des Netzwerks und zum Erzeugen anormaler Fälle wird vorzugsweise eine anwendungsspezifisch integrierte Schaltung für den Test (ASIC) eingesetzt, damit die Fähigkeit besteht, ein durch einen Anwender festgelegtes Bitmuster von bis zu 54 Byte (432 Bit) zu erzeugen. Durch den ASIC wird vorzugsweise auch eine wählbare Baudrate (fünf Geschwindigkeiten) ermöglicht, sowie eine Schnittstelle zu einem PC-Bus.

[0208] Weiterhin ist es wünschenswert, diese Möglichkeiten auf einer Schaltungsplatine vorzusehen. Bei einer Schaltungsimplementierung der Architektur **10** weist die Platine zwei Hauptabschnitte auf: die Kommunikationsschaltung und die Anwendungsschaltung. Die Kommunikationsschaltung handhabt die gesamte Kommunikation über das Netzwerk sowie das Scheduling des Zugriffs der PCN-Knoten **220** und der DCN-Knoten **310** zu dem Netzwerk. Die Anwendungsschaltung enthält einen Prozessor, beispielsweise den Intel 80196 Mikroprozessor, der von Intel Corporation in Santa Clara, Kalifornien, angboten wird, sowie einen Speicher mit wahlfreiem Zugriff (RAM-Speicher), einen Nullesespeicher (ROM-Speicher), und eine RS232-Schnittstelle, und sie übernimmt die Konfigurierung des Netzwerks über die Kommunikationsschaltung und die Verarbeitung der Daten, die zwischen den CCN-Knoten **210** und den PCN-Knoten **220** und/oder dem CCN-Knoten **210** und dem DCN-Knoten **310** über das Kommunikationsnetzwerk ausgetauscht werden.

[0209] Die Kommunikationsschaltung kann unter Einsatz feldprogrammierbarer Gate-Arrays-Schaltungen (FPGA) der Xilinx-4000-Serie implementiert werden, wie sie von Xilinx Corporation in San Jose, Kalifornien, angeboten werden, mit Ausnahme von Fällen, in denen Frequenzanforderungen im Vordergrund stehen. Beim

Anlaufen oder einem generellen Rücksetzen werden die FPGA-Vorrichtungen entweder durch einen EPROM-Speicher konfiguriert, oder durch einen PC-Computer über eine PC-Schnittstelle, was mit Hilfe von Modusschaltern festgelegt wird. Die Haupt-(oder die Master)-Vorrichtung liest vorzugsweise die Konfigurationsdaten für alle FPGA-Einrichtungen von einem EPROM-Speicher. Alle anderen FPGA-Einrichtungen werden vorzugsweise mit einer Daisy-Chain-Master/Slave-Konfiguration geladen. Nachdem alle Vorrichtungen geeignet konfiguriert sind, wird vorzugsweise eine lichtemittierende Diode (LED-Diode) angeschalten, und vorzugsweise wird ein Interrupt zum Anzeigen der Tatsache erzeugt, daß die FPGA-Konfigurierung abgeschlossen ist.

[0210] Daten des Netzwerks erreichen die Platine vorzugsweise über einen Verbinder und eine serielle Busschnittstelle, vorzugsweise einen RS485-Interface-Chip. Nach dem Filtern der Daten wird der Übertragungstakt vorzugsweise durch Flankendetektion in einem besonders schnellen programmierbaren Feldlogikchip (PAL-Chip) abgeleitet. Aus dem abgeleiteten Übertragungstakt wird ein Empfangstakt in der Mitte für jede eintretende Bitperiode erzeugt. Der Empfangstakt wird vorzugsweise erneut mit dem eintretenden. Datentakt synchronisiert, jedesmal dann, wenn eine Flanke auftritt. Die Daten werden anhand dieses Takts einem Haltespeicher zugeführt und dem Rest der Schaltung zugänglich gemacht.

[0211] Die Platine weist vorzugsweise auch einen Kernkommunikationsblock auf. Die Kernkommunikationsschaltung handhabt vorzugsweise die Übertragung, den Empfang, die Fehlerdetektion und den duplizierten Meldungsschutz von Datenpaketen, wie es oben detailliert beschrieben ist. Durch diesen Block wird vorzugsweise die gesamte Biteinfügung und Herauslösung durchgeführt, sowie die Rahmendetektion und der Rahmenaufbau, die CRC-Erzeugung und -überprüfung, die Rahmenerkennung für spezielle Steuervorgänge, die Adressenerkennung und Übertragung, die Meldungseinreihung und die ACK/NACK-Beantwortung. Zusätzlich sind alle kommunikationsspezifischen Register (beispielsweise die Baudrate, die Ladeadresse) vorzugsweise in diesem Block enthalten, und sie werden durch diesen gesteuert.

[0212] Die Platine enthält einen Kommunikationsblock für DCN-Knoten, die die First-In-First-Out-Speicher (FIFO-Speicher) zum Senden und Empfangen bei der Kommunikation mit den CCN-Knoten **210** und die Zugriffsteuerungs-Zustandsmaschinen für die DCN-Knoten **310** enthält. Ein Zugriff auf die Sende- und Empfangs-FIFO-Speicher ist sowohl für den Kernkommunikationsblock als auch die Schnittstelle zu dem DCN-Knoten **310** möglich. Die Zustandsmaschine des DCN-Knotens **310** zeigt dem Kernkommunikationsblock an, wenn eine Meldung zu übertragen ist, und bestimmt, ob ein Zeitablauffehler aufgetreten ist oder nicht. Der Kernkommunikationsblock zeigt dem Kommunikationsblock für die DCN-Knoten **310** an, wenn das Netzwerk belegt ist, wenn ein ACK-Rahmen empfangen wurde und wenn eine NACK-Rahmen empfangen wurde. Dieser Block signalisiert auch der Schnittstelle zu dem DCN-Knoten **310**, wenn eine Meldung empfangen wurde, ein Fehler aufgetreten ist und/oder eine Meldung korrekt abgegeben wurde. Die Zustandsmaschine für die Übertragung zwischen dem CCN-Knoten zu dem DCN-Knoten ist in **Fig. 34** gezeigt.

[0213] Die Schnittstelle zwischen dem Mikroprozessor und dem PCN-Knoten **220**, die in dem Netzwerk vorliegen, besteht vorzugsweise aus einem PCN-RAM-Speicher. Die gesamte Scheduling-Information für jeden PCN-Knoten **220** liegt vorzugsweise in dem PCN-RAM-Speicher vor, der diesem PCN-Knoten **220** zugeordnet ist. Die bevorzugte PCN-RAM-Speicherorganisation ist in Tabelle 8 gezeigt.

Tabelle 8

	Netzwerkadresse	Tick-Status	Datenstatus	Meldungslänge	12 nicht benutzte Bytes	8 Bytes Ausgangspuffer	8 Bytes Eingabepuffer
0	--	--	--	--	--	--	--
1	--	--	--	--	--	--	--
2	--	--	--	--	--	--	--
3	--	--	--	--	--	--	--
:	--	--	--	--	--	--	--
:	--	--	--	--	--	--	--
:	--	--	--	--	--	--	--
:	--	--	--	--	--	--	--
30	--	--	--	--	--	--	--
31	--	--	--	--	--	--	--

[0214] Der RAM-Speicher ist vorzugsweise so organisiert, daß ein Block von 19 Byte Daten jedem potentiellen PCN-Knoten **220** zugeordnet ist. Die Daten sind durch den Host-Prozessor les- und schreibbar. Ein PCN-Knoten **220** kann jedem Datenblock zugoerdnet sein, obgleich die PCN-Knoten **220**, auf die am meisten zugegriffen wird, an der obersten Stelle in dem RAM-Speicher gespeichert werden sollten.

[0215] Das Netzwerkadressenbyte enthält die Netzwerkadresse eines PCN-Knotens **220**, der einem Speicherblock zugeordnet ist. Hierdurch kann der Anwender die PCN-Knoten **220** nicht sequentiell adressieren und den RAM-Speicher in Abhängigkeit von der Zugriffshäufigkeit der PCN-Knoten **220** organisieren. Wie in Fig. 35 gezeigt ist, enthält das Tick-Statusbyte 2 Informationseinheiten, die Aktualisierungsrate und die Zeit bis zur Aktualisierung. Die Aktualisierungsrateneinheit bestimmt, wie oft PCN-Knoten **220** aktualisiert werden, während die Zeit bis zur Aktualisierung als Zähler durch die Zugriffsteuerungs-Zustandsmaschine benützt wird, um zu erfassen, wann die Aktualisierung durchzuführen ist.

[0216] Die Aktualisierungsrate sollte durch den Host-Prozessor festgelegt sein, und sie bestimmt, wie oft innerhalb der Tick-Zyklen ein PCN-Knoten **220** aktualisiert wird. Ein Wert von 0 würde Aktualisierung bei jedem Tick-Zyklus anzeigen, ein Wert von 1 würde Aktualisierung bei jedem anderen Tick-Zyklus anzeigen, usw.. Die Aktualisierungsraten können während des Betriebs modifiziert werden, jedoch ist vorsichtig vorzugehen, wenn sich die PCN-Knoten **220** im direkten Modus befinden, da die Zeit zwischen den Aktualisierungen der PCN-Knoten **220** unterhalb des modifizierten PCN-Knotens **220** asymmetrisch sein kann.

[0217] Die Zeit bis zur Aktualisierung wird vorzugsweise aktiv durch die Zugriffssteuerungs-Zustandsmaschine während jedes Tick-Zyklus modifiziert. Bei jedem Tick-Zyklus wird diese Einheit untersucht. Ist sie 0, so wird der PCN-Knoten **220** aktualisiert, und die Einheit wird zu der Aktualisierungsrate rückgesetzt. Im anderen Fall wird der Wert der Einheit dekrementiert, und der PCN-Knoten **220** wird nicht aktualisiert. Der Anfangswert dieser Einheit kann durch den Host-Prozessor bestimmt werden, um die Zugriffe zwischen den PCN-Knoten **220** zu verteilen. Es ist auf ein Aufrechterhalten des Schedulings zu achten, wenn sich die Aktualisierungsrate verändert.

[0218] Wie in Fig. 36 gezeigt ist, stellt das Datenstatusbyte Informationen für die Daten für einen besonderen der PCN-Knoten **220** zur Verfügung, Es kann benützt werden, um zu gewährleisten, daß ungültige oder veränderte Daten nicht zu dem PCN-Knoten **220** übertragen oder durch den Prozessor gelesen werden. Vorzugsweise werden 4 Bits zum Spezifizieren des Datenstatus für einen besonderen PCN-Knoten **220** benützt.

[0219] Das Fehlerbit, ERR, wird vorzugsweise durch eine Zustandsmaschine des PCN-Knotens **220** gesetzt, wenn ein Fehler beim letzten Aktualisieren des PCN-Knotens **220** aufgetreten ist.

[0220] Das CPU-Modifizierungsbit, CM, kann durch den Host-Prozessor gesetzt werden, um eine Übetragung von Daten an den **220** PCN-Knoten **220** zu verhindern, während er modifiziert wird. Ein spezieller Rahmen wird zu dem PCN-Knoten **220** gesendet, durch den Register des PCN-Knotens **220** nicht aktualisiert werden, durch den jedoch der Status des PCN-Knotens **220** aktualisiert wird. Wird dieses Bit gesetzt, so ist es durch den Host-Prozessor nach der Modifikation der Daten in den PCN-Knoten **220** zu löschen. Der Einsatz dieses Bits ist optional.

[0221] Das Empfangs-Toggle-Bit, RT, wird vorzugsweise durch die Zugriffsteuerungs-Zustandsmaschine getogglet, und zwar jedesmal dann, wenn Daten von einem PCN-Knoten **220** empfangen werden. Der Host-Prozessor kann dieses Bit vor und nach dem Zugriff auf die Daten für den PCN-Knoten **220** untersuchen, um zu bestimmen, ob sich irgendwelche der Daten verändert haben, seitdem der Prozessor mit dem Zugriff auf die Daten begonnen hat.

[0222] Das Bit zum Anzeigen, daß zuvor keine Übertragung erfolgte, NT, wird vorzugsweise durch die Zugriffsteuerungs-Zustandsmaschine gesetzt, wenn ein PCN-Knoten **220** im letzten vorgesehenen Zeitpunkt nicht aktualisiert wurde und zwar aufgrund des CM-Bits, das durch den Host-Prozessor gesetzt wird.

[0223] Wie in Fig. 37 gezeigt ist, die die Meldungslänge zeigt, legt die Zahl der zu übertragenden Bytes fest, wieviele der 8 Bytes in dem Ausgangspuffer zu einem PCN-Knoten **220** zu übertragen sind. Die Zahl der durch den PCN-Knoten **220** zu empfangenden Bytes ist in der höherwertigen Einheit empfangen. Dieser Wert sollte vorzugsweise durch den Host-Prozessor gesetzt werden, und er ist eine Funktion der Funktions-ID-Kennung des PCN-Knotens **220**.

[0224] Die Fig. 38 zeigt die Zugriffsteuerungs-Zustandsmaschine zum Steuern sämtlicher Zugriffe auf das

Netzwerk. Durch die Zugriffsteuerungs-Zustandsmaschine wird das Scheduling der PCN-Knoten **220** aufrecht erhalten, die Kommunikation der PCN-Knoten **220** durchgeführt und der DCN-Zugriffssteuerrahmen (DAC) erzeugt. Signale zwischen dem Kernkommunikationsblock und der Zugriffsteuerungs-Zustandsmaschine ermöglichen der Zustandsmaschine die Steuerung dieser Datenkommunikationsvorgänge.

[0225] Der Beginn einer Tick-Periode initialisiert die Zugriffsteuerungs-Zustandsmaschine, die die geeigneten Kommunikationsvorgänge zwischen den PCN-Knoten **220** auf Grundlage der Daten in den PCN-RAM-Speicher durchführt. Bei – Abschluß einer Kommunikation mit einem PCN-Knoten **220** bestimmt die Zugriffsteuerungs-Zustandsmaschine, ob genügend Zeit in der Tick-Periode für einen DAC-Rahmen und die zugeordnete Kommunikation mit einem DCN-Knoten verbleibt. Ist dies der Fall, so kann der CCN-Knoten **220** eine Meldung übertragen, oder er kann den DAC-Rahmen übertragen. Bei Abschluß der Meldung des CCN-Knotens **210** oder des DAC-Rahmens und einer möglichen Meldung eines DCN-Knoten **310** führt die Zugriffsteuerungs-Zustandsmaschine eine erneute Prüfung durch, um festzustellen, ob ausreichend Zeit in der Tick-Periode für einen DAC-Rahmen und eine Kommunikation mit einem DCN-Knoten **310** verbleibt. Dieser Prozeß wird solange fortgesetzt, bis keine ausreichende Zeit in der Tick-Periode verbleibt, und an diesem Punkt wartet die Zustandsmaschine bis zum Beginn der nächsten Tick-Periode, um die Kommunikationsvorgänge mit den PCN-Knoten **220** zu beginnen.

[0226] Eine Mikroprozessorschnittstelle steuert vorzugsweise den Zugriff zwischen den Registern des CCN-Knotens **220**, den PCN-RAM-Speichern, den FIFO-Speichern der DCN-Knoten **310** und den Statusregistern. Alle Status- und Interrupt-Register sind in diesem Block enthalten. Die Schnittstelle ermöglicht dem Mikroprozessor einen transparenten Zugriff auf alle internen RAM-Speicher und Register.

[0227] Ferner erzeugt eine Gruppe von Status-LED-Dioden vorzugsweise Information für den Zweck der Überprüfung der Platine und des Systems. Sie zeigen die folgenden Zustände an: FPGA-Konfiguration abgeschlossen, dritter Draht in Aktion, dritter Draht nicht in Aktion, RS485-Freigabe aktiv, Übertragen der FIFO-Freigabe durch DCN-Knoten zum Übertragen und Empfang der Mitteilung durch DCN-Knoten, daß der FIFO-Speicher eine Meldung empfangen hat.

[0228] Schließlich stellt ein DIP-Schalter vorzugsweise die Funktions-ID-Kennung und die Adresse des CCN-Knotens **210** ein. Diese ist üblicherweise zu Null festgelegt, kann jedoch auf einen unterschiedlichen Wert verändert werden, damit die Platine des CCN-Knotens **210** als DCN-Knoten **310** für Überprüfungszwecke funktioniert.

[0229] Demnach ist zu erkennen, daß ein flexibles Bewegungssteuer-Kommunikationssystem mit serieller Kommunikation geschaffen wird, das eine wirksame Kommunikation zwischen unterschiedlichsten Elementen in einem Bewegungssteuersystem ermöglicht und das das einfache Einbeziehen modularer Zusätze bei einem Bewegungssteuersystem ermöglicht.

[0230] Das System gestattet auch eine Detekton des Verdrahtungsaufwands, der zum Verbinden der erforderlichen Elemente eines Bewegungssteuersystems erforderlich ist, und es ermöglicht die bessere Isolierung der Verdrahtung gegenüber einer elektromagnetischen und einer Hochfrequenz-Interferenz.

[0231] Das System ermöglicht auch die einfache Rekonfigurierung der Systeme, wodurch die Produktentwicklung beschleunigt wird, und es ergeben sich reduzierte Systemkosten durch Erhöhung der Zahl der gemeinsam benützten Teile.

[0232] Das System ermöglicht auch eine Unterstützung auf Anwendungsebene während eines Anlaufvorgangs, einschließlich der Herstellungs- und Servicediagnosevorgänge, und es ist ferner die automatische Konfigurierung eines Netzwerks nach einer erneuten Systemkonfiguration möglich.

[0233] Das System ermöglicht auch Knoten für das Motorsteuerkommunikationssystem, die Boundary-scanbar zum Testen sind und bei denen Boundary-Scan Information zu oder von einem einzigen Netzwerkverbindungspunkt übertragbar ist, damit ein Herstellungstest und eine Servicediagnose möglich ist.

[0234] Das System ermöglicht auch die Angleichung der Kommunikations-Baudrate zum Erzielen einer optimalen Abwägung zwischen Kosten/Bandbreite.

[0235] Das System ermöglicht auch ein Bewegungssteuersystem mit serieller Kommunikation, das die Kommunikation zwischen mehreren verteilten Steuerknoten ermöglicht, sowie zwischen mehreren peripheren

Steuerknoten und einem zentralisierten Steuerknoten, und zwar unter Einsatz eines seriellen Busses, der mit dem zentralisierten Steuerknoten kommuniziert, sowie mit den verteilten Steuerknoten und den pripheren Steuerknoten und es enthält eine Vorrichtung für die Kommunikation der Steuer- und Datensignale zwischen den zentalisierten Steuerknoten, den verteilten Steuerknoten und den peripheren Steuerknoten.

Patentansprüche

1. Bewegungssteuersystem (**10**) mit serieller Kommunikation, enthaltend einen zentralisierten Steuerknoten (**210**) und mehrere andere Steuerknoten (**220**, **310**) sowie einen seriellen Bus (**230**) zum Verbinden sämtlicher Knoten in dem System für die Kommunikation;
dadurch gekennzeichnet, daß
a) mehrere der anderen Steuerknoten ereignisgetriebene verteilte Steuerknoten (**310**) sind, mit einer Fähigkeit, entweder den zentralisierten Steuerknoten oder einen der anderen ereignisgetriebenen verteilten Steuerknoten in dem System auszuwählen und asynchron damit zu kommunizieren,
b) jeder der verteilten Steuerknoten (**310**) ein Knotenschlitzregister hat;
c) zumindest einer der anderen Knoten ein peripherer Steuerknoten (**220**) mit der Fähigkeit ist, synchron mit dem zentralisierten Steuerknoten zu kommunizieren; wobei
d) der zentralisierte Steuerknoten (**210**) betreibbar ist zum Runterladen einer eindeutigen Schlitzidentifizierungsnummer zu jedem verteilten Steuerknoten (**310**) zur Speicherung in dem Knotenschlitzregister davon;
e) jeder verteilte Steuerknoten (**310**) betreibbar ist auf der Basis der in seinem Knotenschlitzregister gespeicherten Nummer zu bestimmen, wie viele Schlitze er warten muss vor einem Initiieren einer Übertragung; und
f) während einem festgelegten Intervall der zentralisierte Steuerknoten (**210**) zum Initiieren einer synchronen Kommunikation mit dem peripheren Steuerknoten (**220**) in Übereinstimmung mit einem festgelegten Scheduling-Plan betreibbar ist, und, wenn nach dem Abschluß der synchronen Kommunikation, die in dem Intervall eingeteilt ist, ausreichend Zeit in dem Intervall verbleibt, ferner betreibbar ist zum Vermitteln einer asynchronen Kommunikation zwischen einem der verteilten Steuerknoten und entweder dem zentralisierten Steuerknoten oder einem anderen der anderen verteilten Steuerknoten, wie ausgewählt durch den die asynchrone Kommunikation durchführenden verteilten Steuerknoten, mit Nutzen des auf der Basis der in dem Knotenschlitzregister gespeicherten Nummer bestimmten Zeitschlitzes.

2. System nach Anspruch 1, dadurch gekennzeichnet, daß jeder der peripheren Steuerknoten enthält:
eine periphere Knotenschnittstellenvorrichtung (**750**) für die Schnittstellenbildung mit dem seriellen Bus; und
eine Eingabe/Ausgabeschaltung (**740**) für die Kommunikation mit der peripheren Knotenschnittstellenvorrichtung zum Durchführen einer Eingangsverarbeitung für die Hardwarekomponenten des Bewegungssteuersystems.

3. System nach Anspruch 1 oder 2, dadurch gekennzeichnet, daß zumindest einer der verteilten Steuerknoten enthält:
eine lokale Verarbeitungsvorrichtung (**750**) zum Bereitstellen lokaler Verarbeitungskapazitäten; und
eine Schnittstellenvorrichtung (**740**) des verteilten Steuerknotens zu dem seriellen Bus für die Schnittstelle zu dem seriellen Bus, wobei die Schnittstellenvorrichtung des verteilten Steuerknotens zu dem seriellen Bus mit der Verarbeitungsvorrichtung des verteilten Knotens kommuniziert.

4. System nach Anspruch 3, dadurch gekennzeichnet, daß der zentralisierte Steuerknoten ferner enthält:
eine Vorrichtung (**750**) zum Durchführen der Netzwerksteuerung und zum Ermöglichen einer Kommunikation zwischen gleichrangigen Knoten zwischen den verteilten Steuerknoten und zwischen den verteilten Steuerknoten und dem zentralisierten Steuerknoten über die Schnittstellenvorrichtung des verteilten Steuerknotens zu dem seriellen Bus sowie dem seriellen Bus; wobei
zumindest eine der lokalen Verarbeitungsvorrichtungen (**740**) eine Zentralverarbeitungseinheit enthält.

5. System nach Anspruch 1, dadurch gekennzeichnet, daß jeder der ereignisgetriebenen verteilten Steuerknoten eine zugeordnete lokale Prozessorvorrichtung (**740**) enthält und jeder der peripheren Steuerknoten enthält:
eine Schnittstellenvorrichtung des peripheren Knotens (**750**) für die Schnittstellenbildung zu dem seriellen Bus;
eine Eingabe/Ausgabeschaltung (**19**), die mit der peripheren Knotenschnittstellenvorrichtung zum Durchführen einer Eingabeverarbeitung der Hardwarekomponenten des Bewegungssteuersystems kommuniziert; und zumindest eine der folgenden Einheiten:
(1) eine Sensorgesamtheit (**22**) mit zumindest einem Sensor und einer Sensorschnittstellenvorrichtung für die Schnittstellenbildung zu den Sensoren, wobei die Sensorschnittstellenvorrichtung mit der peripheren Knotenschnittstellenvorrichtung kommuniziert,

(2) einen smarten Motor (**21**) und eine smarte Schnittstellenschaltung für die Schnittstellenbildung zu dem smarten Motor, wobei die smarte Schnittstellenschaltung mit der peripheren Knotenschnittstellenvorrichtung kommuniziert, und
(3) einen Solenoid (**20**) und eine Solenoidschnittstellenvorrichtung für die Schnittstellenbildung zu dem Solenoid, wobei die Solenoidschnittstellenvorrichtung mit der peripheren Knotenschnittstellenvorrichtung kommuniziert;
(4) eine ausgewiesene Eingabe/Ausgabevorrichtung (**14**) mit zumindest einer der folgenden Einheiten:
eine parallele Schnittstellenvorrichtung zum Senden und Empfangen paralleler Datensignale und paralleler Steuersignale, und
eine serielle Schnittstellenvorrichtung zum Senden und Empfangen serieller Datensignale und serieller Steuersignale.

6. Das System nach Anspruch 5, dadurch gekennzeichnet, daß jeder der verteilten Steuerknoten enthält:
eine Schnittstellenvorrichtung (**750**) des verteilten Steuerknotens zu dem seriellen Bus für die Schnittstellenbildung zu dem seriellen Bus, wobei die Schnittstellenvorrichtung des verteilten Steuerknotens zu dem seriellen Bus mit der lokalen Prozessorvorrichtung (**740**) des verteilten Knotens kommuniziert.

7. System nach Anspruch 1 oder nach Anspruch 6, dadurch gekennzeichnet, daß der zentralisierte Steuerknoten ferner enthält:
eine Generatorvorrichtung zum Generieren von Tick-Signalen bei regulären zeitlichen Tickintervallen; und
eine zentralisierte Knotensteuervorrichtung zum Generieren peripherer Knotensteuersignale, wobei jedem der peripheren Steuerknoten ein Zeitscheibenintervall innerhalb des Tickintervalls zugeordnet ist und jedes der Zeitscheibenintervalle der peripheren Steuerknoten die Länge einer Zeitscheibeneinheit aufweist und die Zeitscheibenintervalle die peripheren Steuerknoten eine Verzögerung relativ zu dem Ticksignal aufweisen, die ein ganzzahliges Vielfaches der Zeitscheibeneinheiten ist, wodurch die peripheren Knotensteuersignale, die im Zusammenhang mit den Kommunikationsvorgängen zwischen den zentralisierten Steuerknoten und einem der peripheren Steuerknoten stehen, voneinander um ein ganzzahliges Vielfaches der Tick-Intervalle getrennt sind; und
zumindest einer der peripheren Steuerknoten enthält:
eine Empfangsvorrichtung zum Empfangen der peripheren Knotensteuersignale in dem zugeordneten Zeitscheibenintervall des peripheren Steuerknotens;
und
eine Vorrichtung zum Bilden von Antwortsignalen in Ansprechen auf die peripheren Knotensteuersignale, die von der Empfangsvorrichtung empfangen werden.

8. System nach Anspruch 7, dadurch gekennzeichnet, daß der zentralisierte Steuerknoten ferner eine Vorrichtung zum dynamischen Ableiten eines neuen Scheduling-Plans zum Verändern der ganzzahligen Zahl der Tick-Intervalle aufweist.

9. System nach Anspruch 1 oder nach Anspruch 5, dadurch gekennzeichnet, daß der zentralisierte Steuerknoten und zumindest einer der mehreren verteilten Steuerknoten und der mehreren peripheren Steuerknoten über den seriellen Bus in einer Ringkonfiguration verbunden sind.

10. System nach Anspruch 1 oder nach Anspruch 5, dadurch gekennzeichnet, daß der zentralisierte Steuerknoten und mindestens einer der mehreren verteilten Steuerknoten und der mehreren peripheren Steuerknoten über den seriellen Bus in einer Mehrfachabzweig-Konfiguration verbunden sind.

11. System nach Anspruch 9 oder 10, dadurch gekennzeichnet, daß der zentralisierte Steuerknoten ferner eine Vorrichtung (**740**) zum Bestimmen der physikalischen Reihenfolge der mehreren der anderen Steuerknoten enthält.

12. System nach Anspruch 11, dadurch gekennzeichnet, daß der zentralisierte Steuerknoten ferner eine Vorrichtung zum Runterladen eindeutiger Adressen zu den peripheren Steuerknoten und den verteilten Steuerknoten enthält.

13. System nach Anspruch 9 oder 10, dadurch gekennzeichnet, daß es ferner eine Dritte-Draht-Vorrichtung für die Kommunikation mit den Knoten enthält und daß der zentralisierte Steuerknoten eine Vorrichtung zum Bestimmen der physikalischen Reihenfolge der mehreren anderen Steuerknoten enthält.

14. System nach einem der vorhergehenden Ansprüche, dadurch gekennzeichnet, daß der serielle Bus

ferner enthält:
einen Verbindungspunkt für die Kommunikation, Knotenidentifikationssignale, Testvektoren, und Scan-Modussignale, die von einem Tester für den zentralisierten Steuerknoten erzeugt werden; und
der zentralisierte Steuerknoten ferner enthält:
eine Vorrichtung zum Aufnehmen der Knotenidentifikationssignale, der Testvektoren und der Scan-Modussignale, wobei der zentralisierte Steuerknoten im wesentlichen diejenigen der peripheren Steuerknoten und der verteilten Steuerknoten in einen statischen Halbwertzustand versetzt, die zu testen sind;
eine Vorrichtung zum Übertragen der Testvektoren und der Scan-Modussignale zu der Scan-Logik derjenigen peripheren Steuerknoten und der verteilten Steuerknoten, die sowohl den Scan-Test unterstützen und die zu testen sind, zum Erzeugen eines Scan-Test-Ergebnisvektors;
eine Vorrichtung zum Empfangen des Scan-Testergebnisvektors der Scan-Logik; und
eine Vorrichtung zum Senden des Scan-Testergebnisvektors zu dem Tester.

Es folgen 21 Blatt Zeichnungen

Anhängende Zeichnungen

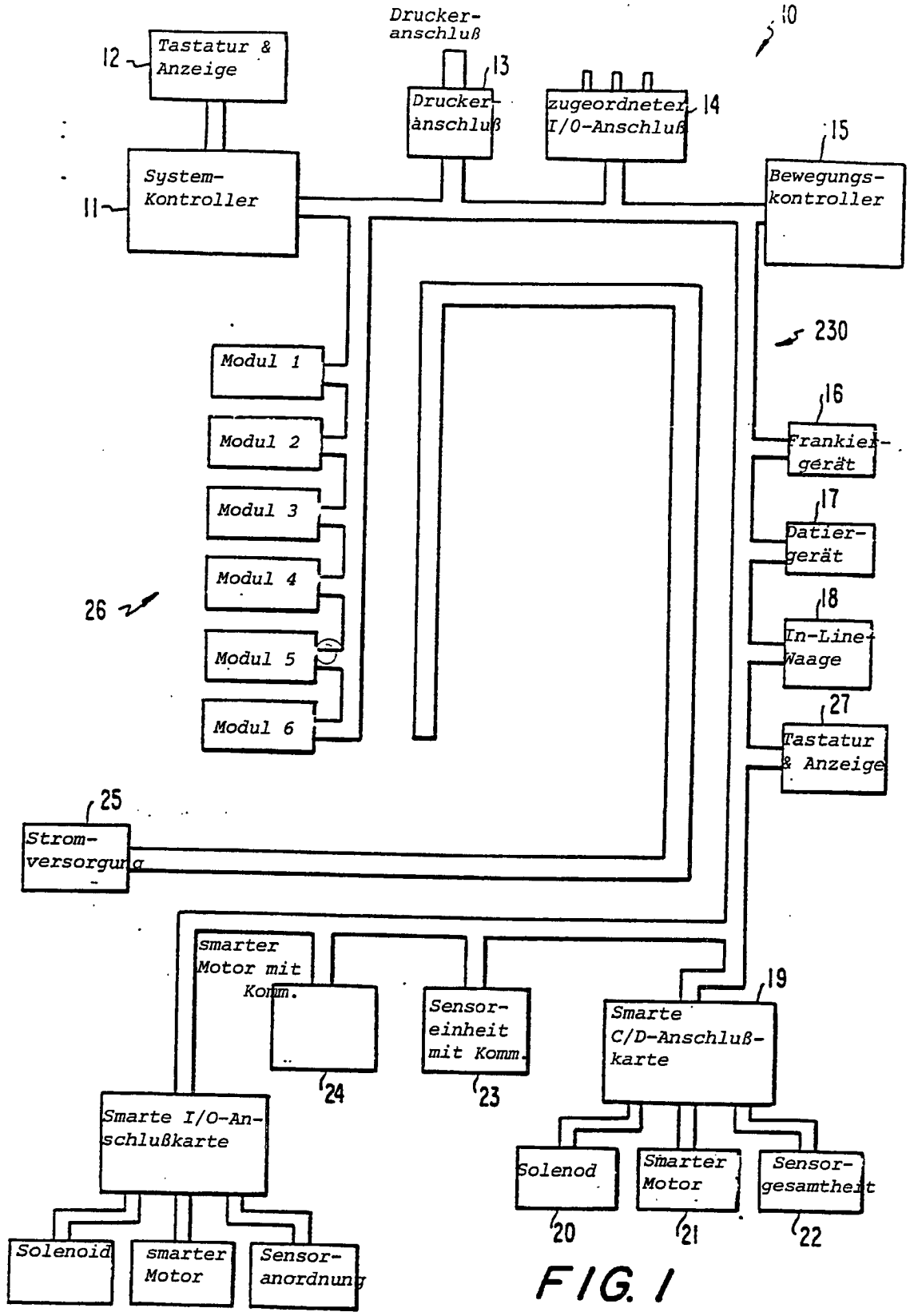


FIG. 1

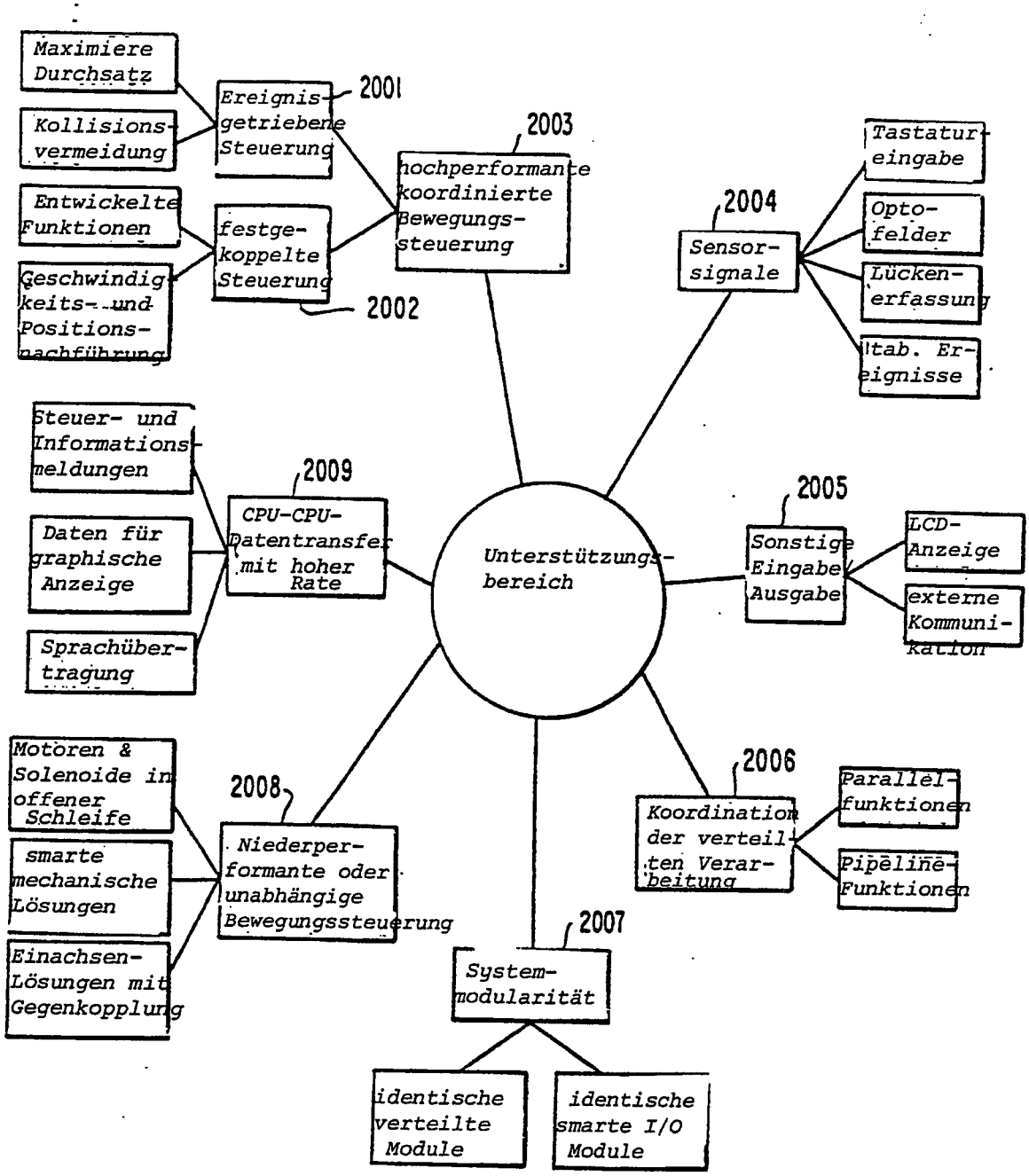

Unterstützungs-bereich
2001
Ereignis-getriebene Steuerung
Maximiere Durchsatz
Kollisions-vermeidung
2002
festge-koppelte Steuerung
Entwickelte Funktionen
Geschwindig-keits- und Positions-nachführung
2003
hochperformante koordinierte Bewegungs-steuerung
2004
Sensor-signale
Tastatur-eingabe
Opto-felder
Lücken-erfassung
Itab. Er-eignisse
2005
Sonstige Eingabe/ Ausgabe
LCD-Anzeige
externe Kommuni-kation
2006
Koordination der verteil-ten Verar-beitung
Parallel-funktionen
Pipeline-Funktionen
2007
System-modularität
identische verteilte Module
identische smarte I/O Module
2008
Niederper-formante oder unabhängige Bewegungssteuerung
Motoren & Solenoide in offener Schleife
smarte mechanische Lösungen
Einachsen-Lösungen mit Gegenkopplung
2009
CPU-CPU-Datentransfer mit hoher Rate
Steuer- und Informations-meldungen
Daten für graphische Anzeige
Sprachüber-tragung

FIG. 2

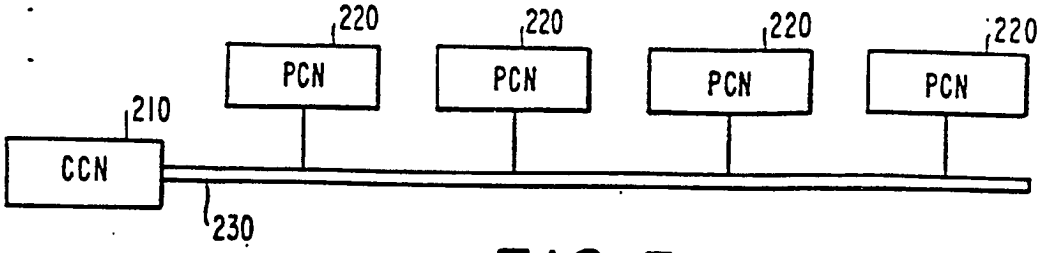


FIG. 3

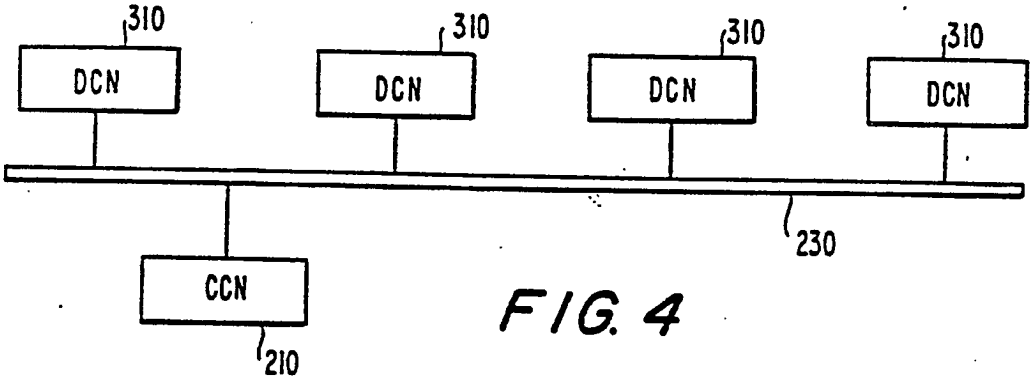


FIG. 4

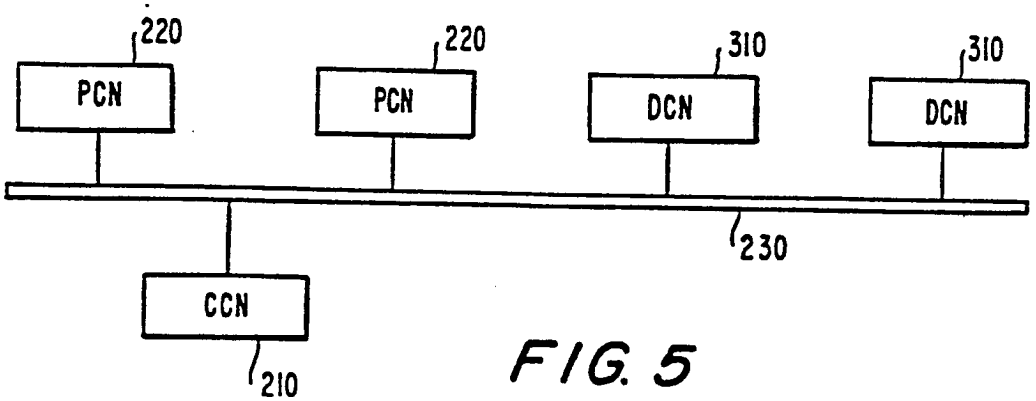


FIG. 5

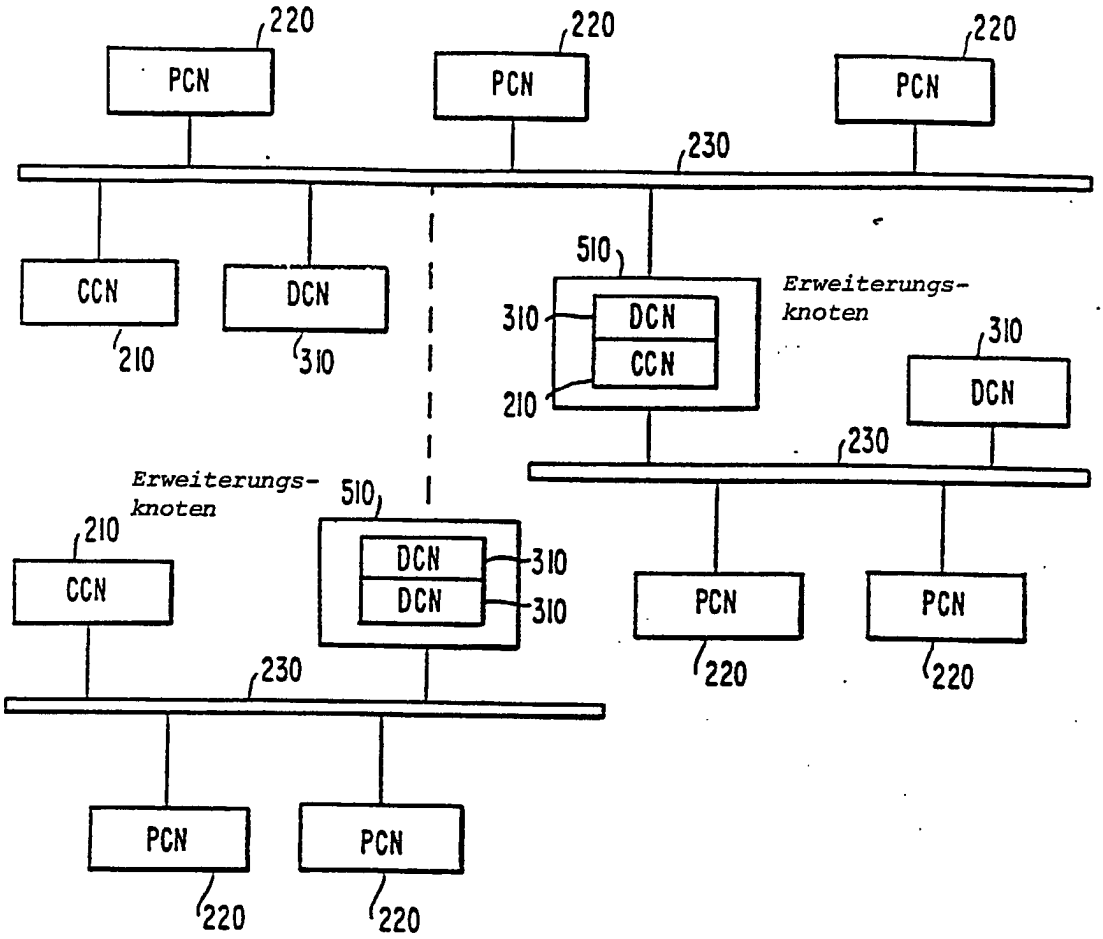

220
PCN
220
PCN
220
PCN
230
CCN
210
DCN
310
510
310
DCN
CCN
210
Erweiterungs-
knoten
310
DCN
230
Erweiterungs-
knoten
510
210
CCN
DCN
310
DCN
310
PCN
220
PCN
220
230
PCN
220
PCN
220

FIG. 6

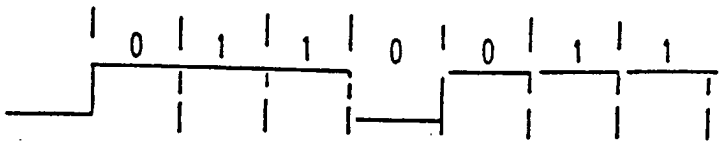


FIG. 7

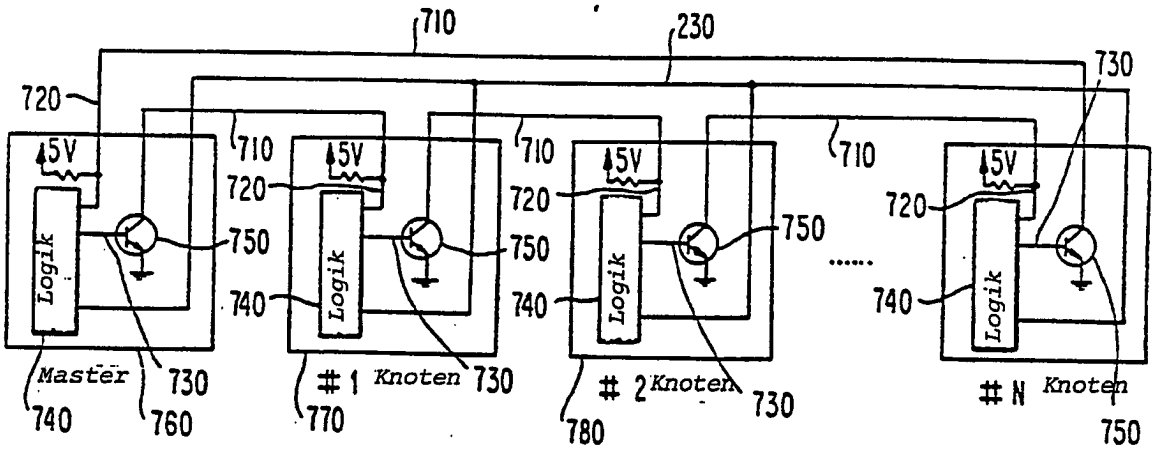


FIG. 8

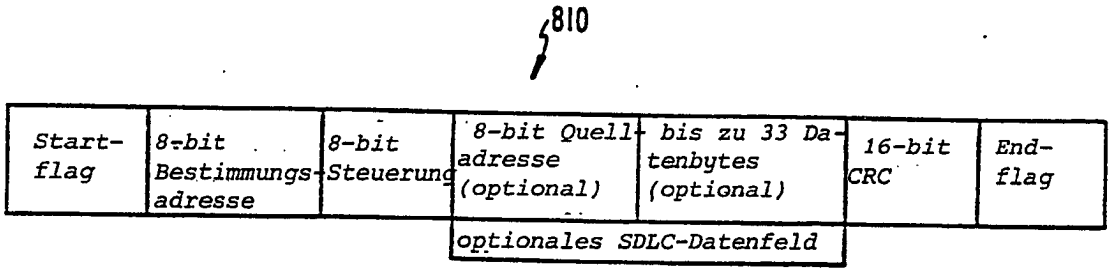


FIG. 9

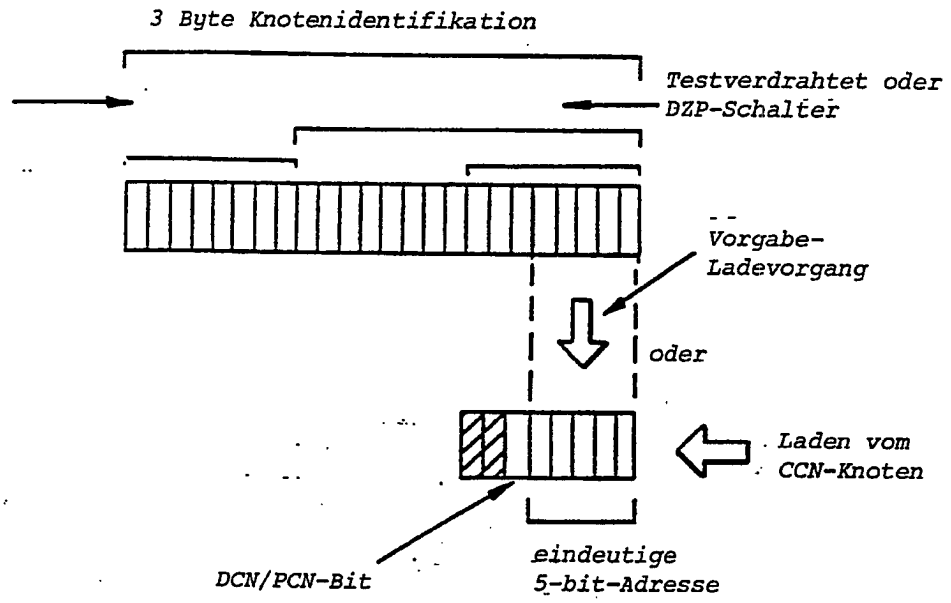

3 Byte Knotenidentifikation
Testverdrahtet oder
DZP-Schalter
Vorgabe-
Ladevorgang
oder
Laden vom
CCN-Knoten
DCN/PCN-Bit
eindeutige
5-bit-Adresse

FIG. 10

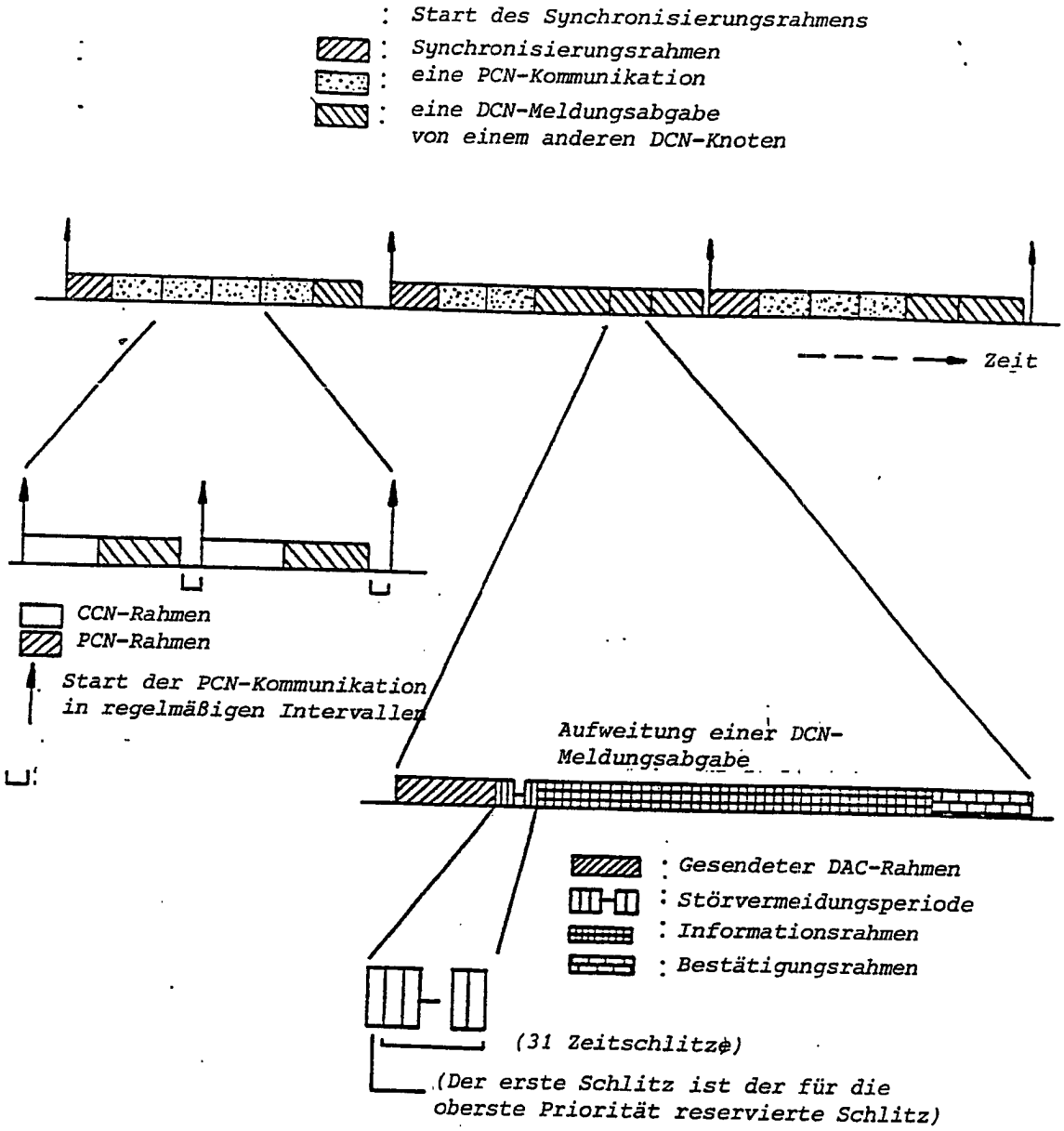

: Start des Synchronisierungsrahmens
: Synchronisierungsrahmen
: eine PCN-Kommunikation
: eine DCN-Meldungsabgabe von einem anderen DCN-Knoten
Zeit
CCN-Rahmen
PCN-Rahmen
Start der PCN-Kommunikation in regelmäßigen Intervallen
Aufweitung einer DCN-Meldungsabgabe
: Gesendeter DAC-Rahmen
: Störvermeidungsperiode
: Informationsrahmen
: Bestätigungsrahmen
(31 Zeitschlitze)
(Der erste Schlitz ist der für die oberste Priorität reservierte Schlitz)

FIG. 11

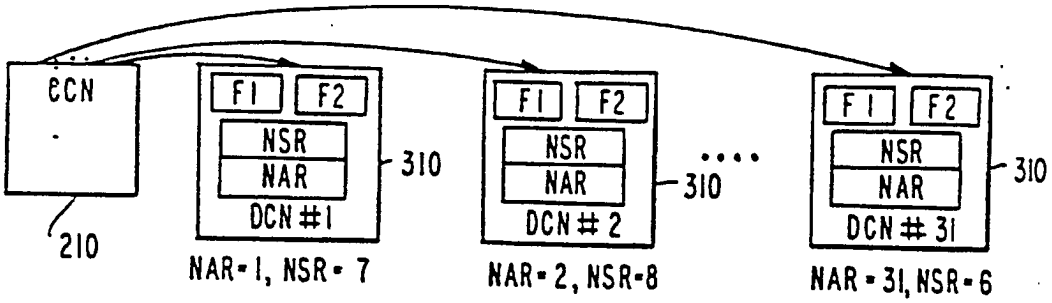

CCN
210
F1
F2
NSR
NAR
DCN #1
310
NAR=1, NSR=7
F1
F2
NSR
NAR
DCN #2
310
NAR=2, NSR=8
F1
F2
NSR
NAR
DCN #31
310
NAR=31, NSR=6

FIG. 12

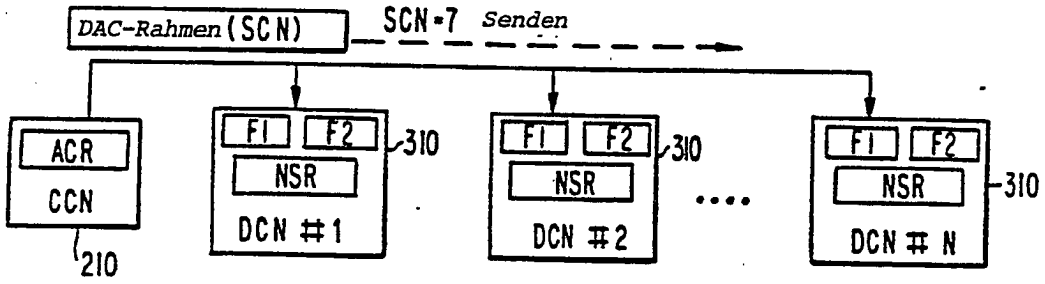

DAC-Rahmen (SCN)
SCN=7 Senden
ACR
CCN
210
F1
F2
NSR
DCN #1
310
F1
F2
NSR
DCN #2
310
F1
F2
NSR
DCN # N
310

FIG. 13

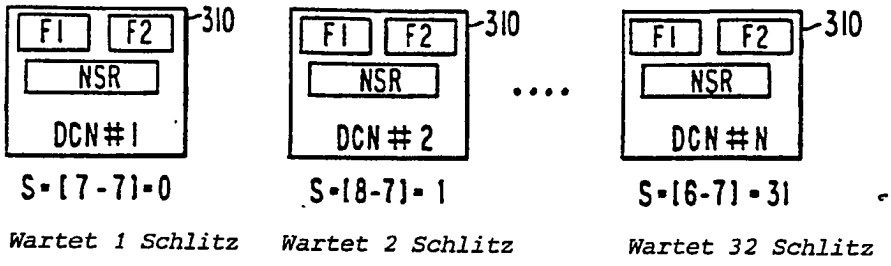

F1
F2
NSR
DCN#1
310
S-[7-7]-0
Wartet 1 Schlitz
F1
F2
NSR
DCN#2
310
S-[8-7]-1
Wartet 2 Schlitz
....
F1
F2
NSR
DCN#N
310
S-[6-7]-31
Wartet 32 Schlitz

FIG. 14

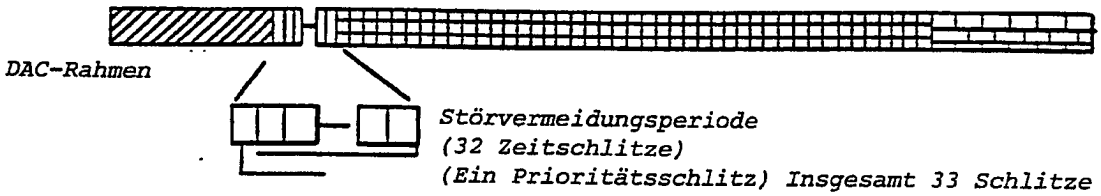

DAC-Rahmen
Störvermeidungsperiode
(32 Zeitschlitze)
(Ein Prioritätsschlitz) Insgesamt 33 Schlitze

FIG. 15

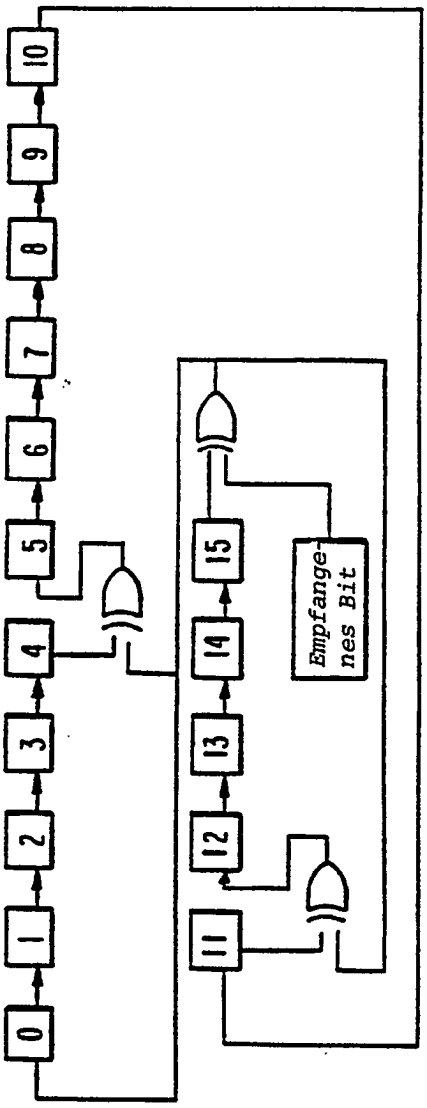


FIG. 16

Start-flag	8 Bit Bestimmungs-adresse	8 Bit Steuerung	Optimale Daten	16 Bit CRC	End-flag

FIG. 17

Startflag	8 Bit Bestimmungs-adresse	8 Bit Steuerung	Bis zu 8 Bytes	16 Bit CRC	Endflag

FIG. 18

Start-flag	8 Bit Bestimmungs-adresse	8 Bit Steuerung	8 Bit Quell-adresse	Bis zu 33 Bytes Daten	16 Bit CRC	Endflag
			SDLC-Datenfeld			

FIG. 19

Flag /BCA/DCF1/ BRS/CRC/Flag ----->

<----- keine Antwort

FIG. 20

CCN-Knoten PCN-Knoten oder DCN-Knoten

Flag /BCA/DCF2/ADD/CRC/Flag -->

<--Flag/CCN/DCF22/CRC /Flag

FIG. 21A

CCN-Knoten PCN-Knoten oder DCN-Knoten

Flag/ BCA/DCF3/ADD/CRC/Flag -->

<-- Flag/CCN/DCF33/CRC/Flag

FIG. 21B

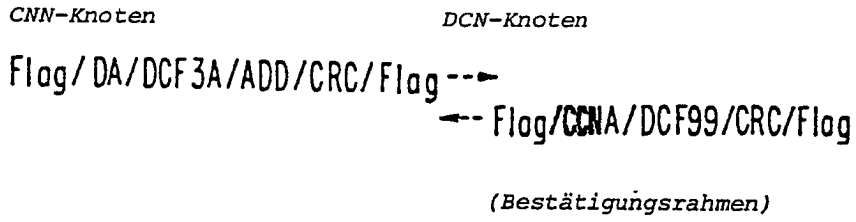


FIG. 22

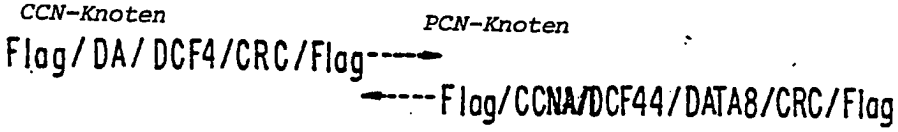


FIG. 23A

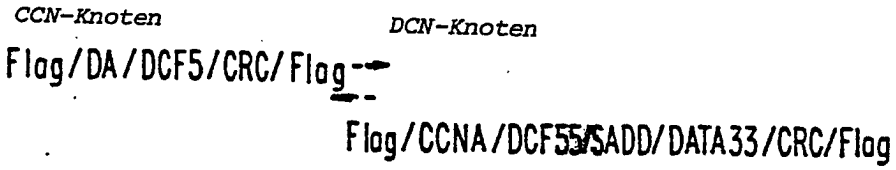


FIG. 23B

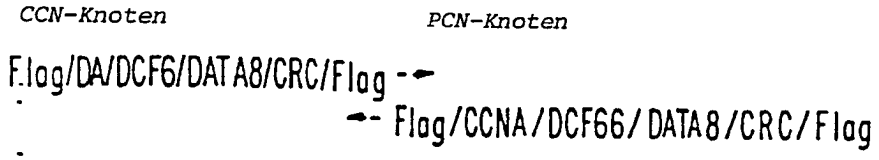


FIG. 24A

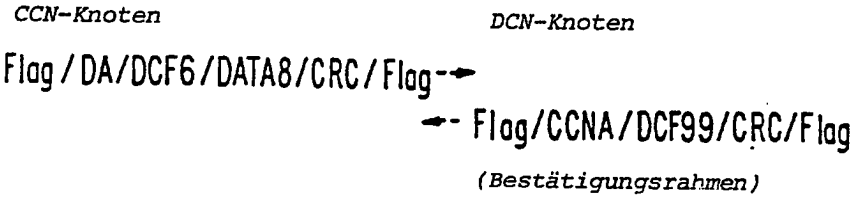


FIG. 24B

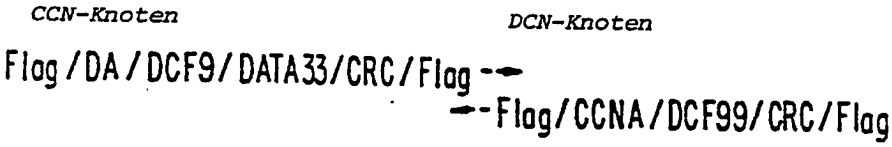


FIG. 25

Flag/BCA/DCF7/CRC/Flag ----->

<----- keine Antwort

FIG. 26

CCN-Knoten DCN-Knoten

Flag/DA/DCFA/DATA33/CRC/Flag ->

<- Flag/CCNA/DCF99/SADD/CRC/Flag

FIG. 27

CCN-Knoten DCN-Knoten

Flag/DA/DCFx/CRC/Flag ->

<- Flag/CCN/DCFx/CRC/Flag

FIG. 28

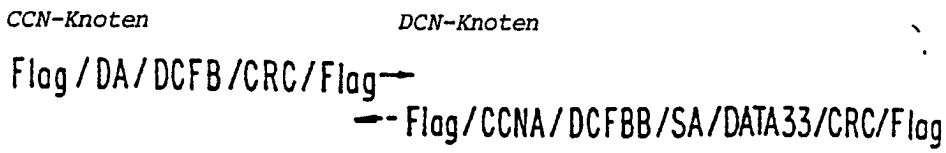


FIG. 29A

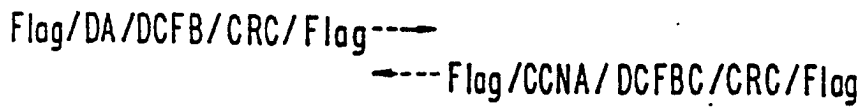


FIG. 29B

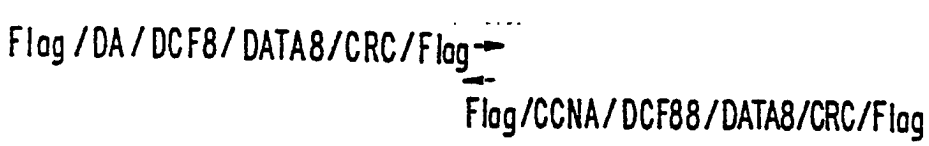


FIG. 30

Flag/DA/DCFC/DATA33/CRC/Flag -->
<-- Flag/CCNA/DCF99/CRC/Flag

FIG. 31

Flag/BRA/DCFD/SCN/CRC/Flag --> (CCN-Senden)
Flag/DCN1/DCFE/DCN2/DATA33/CRC/Flag (DCN1-Knoten sendet zu DCN2-Knoten)
<-- (Bestätigung vom DCN1-Knoten an DCN2-Knoten)

FIG. 32

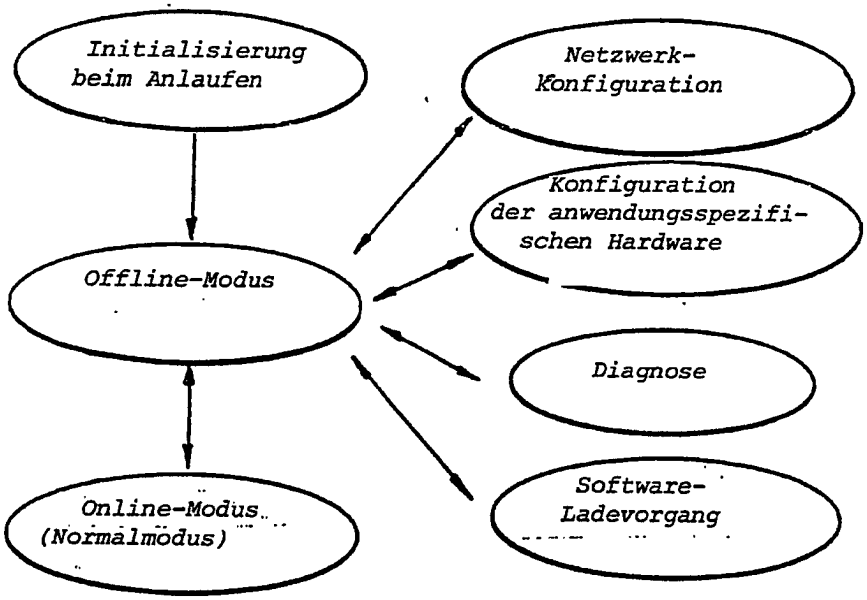

Initialisierung beim Anlaufen
Offline-Modus
Online-Modus (Normalmodus)
Netzwerk-Konfiguration
Konfiguration der anwendungsspezifischen Hardware
Diagnose
Software-Ladevorgang

FIG. 33

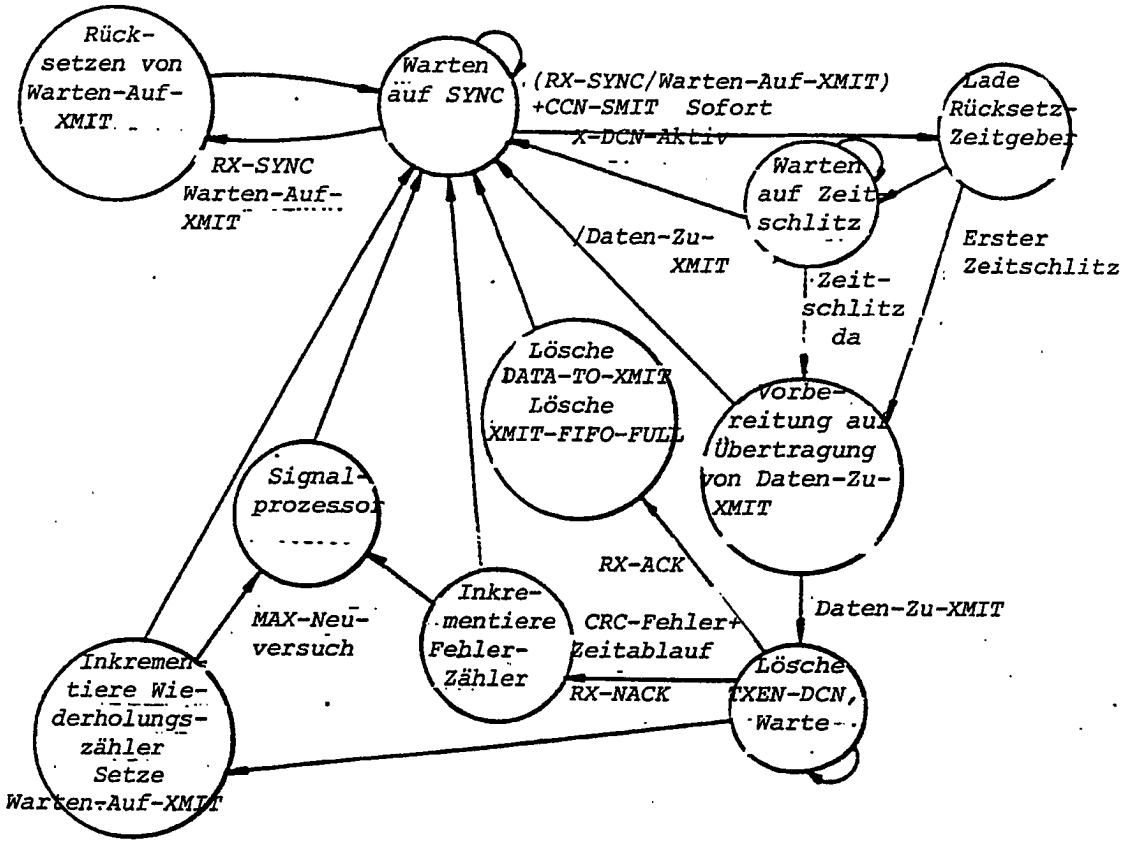

Rück-
setzen von
Warten-Auf-
XMIT
RX-SYNC
Warten-Auf-
XMIT
Warten
auf SYNC
(RX-SYNC/Warten-Auf-XMIT)
+CCN-SMIT Sofort
X-DCN-Aktiv
Lade
Rücksetz-
Zeitgeber
Warten
auf Zeit-
schlitz
/Daten-Zu-
XMIT
Erster
Zeitschlitz
Zeit-
schlitz
da
Lösche
DATA-TO-XMIT
Lösche
XMIT-FIFO-FULL
Vorbe-
reitung auf
Übertragung
von Daten-Zu-
XMIT
Signal-
prozessor
RX-ACK
Daten-Zu-XMIT
MAX-Neu-
versuch
Inkre-
mentiere
Fehler-
Zähler
CRC-Fehler+
Zeitablauf
RX-NACK
Lösche
TXEN-DCN,
Warte
Inkrementiere Wie-
derholungs-
zähler
Setze
Warten-Auf-XMIT

FIG. 34

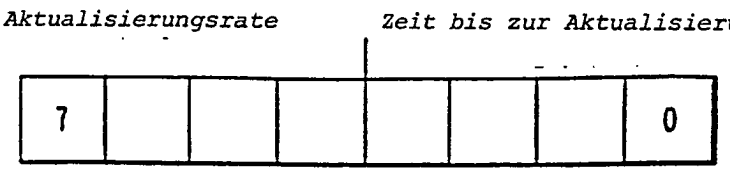
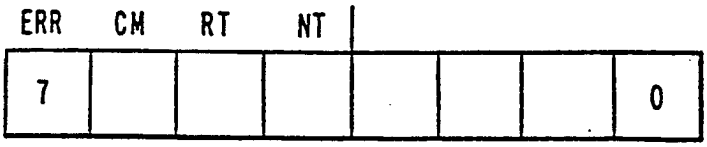


FIG. 35



ERR-Fehlerbit
CM-CPU-Modifizierungsbit
RT-Empfang des Togglebits
NT-Bit zum Anzeigen, daß zuvor keine Übertragung erfolgte

FIG. 36

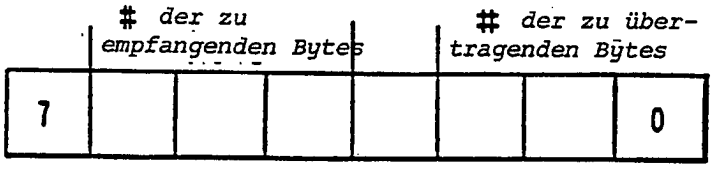


FIG. 37

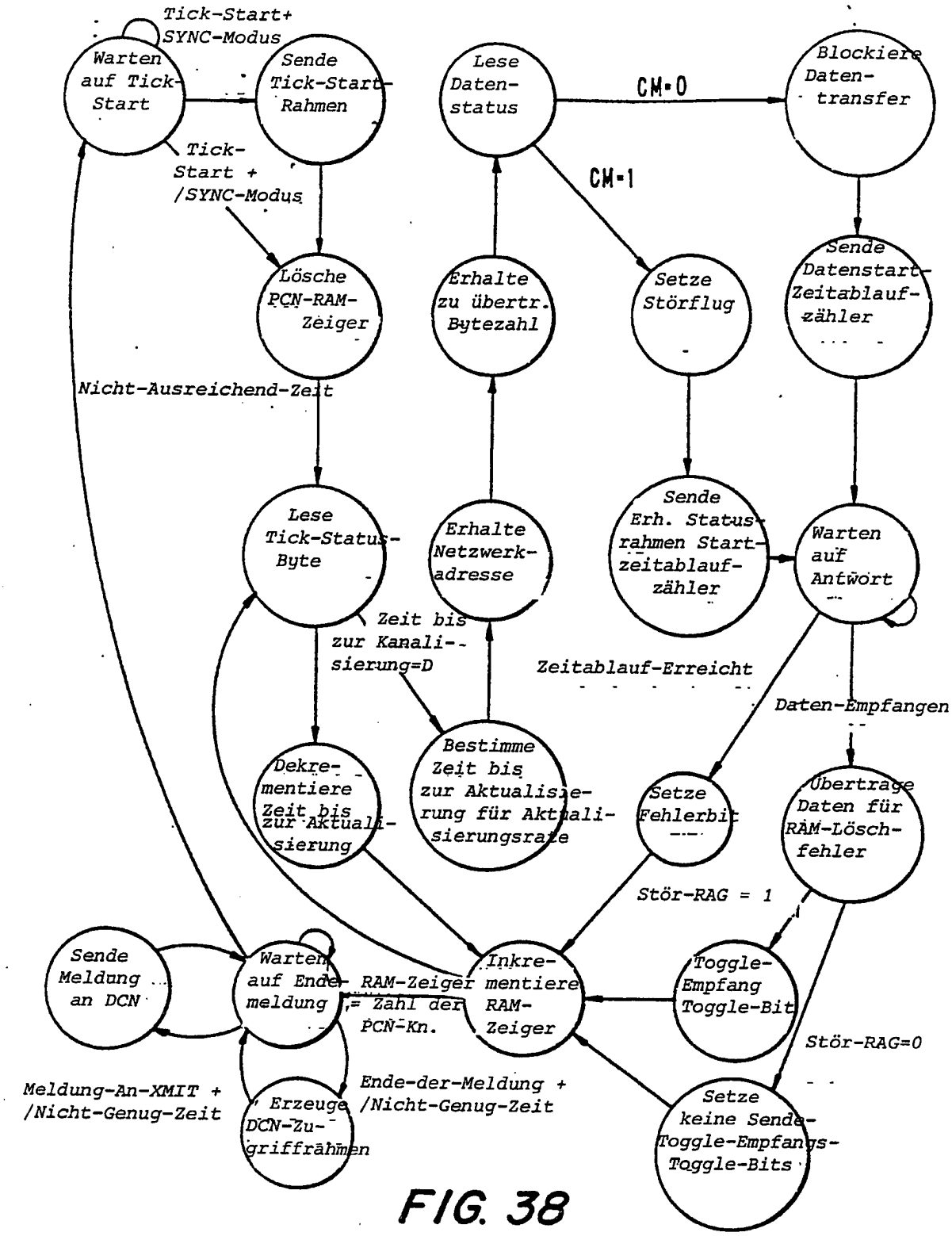

Tick-Start+
SYNC-Modus
Warten auf Tick-Start
Sende Tick-Start-Rahmen
Tick-Start + /SYNC-Modus
Lese Daten-status
CM=0
Blockiere Daten-transfer
CM=1
Lösche PCN-RAM-Zeiger
Erhalte zu übertr. Bytezahl
Setze Störflug
Sende Datenstart-Zeitablauf-zähler
Nicht-Ausreichend-Zeit
Lese Tick-Status-Byte
Erhalte Netzwerk-adresse
Sende Erh. Status-rahmen Start-zeitablauf-zähler
Warten auf Antwort
Zeit bis zur Kanali-sierung=D
Zeitablauf-Erreicht
Daten-Empfangen
Dekre-mentiere Zeit bis zur Aktuali-sierung
Bestimme Zeit bis zur Aktualisie-rung für Aktuali-sierungsrate
Setze Fehlerbit
Übertrage Daten für RAM-Lösch-fehler
Stör-RAG = 1
Sende Meldung an DCN
Warten auf Ende-meldung
RAM-Zeiger = Zahl der PCN-Kn.
Inkre-mentiere RAM-Zeiger
Toggle-Empfang Toggle-Bit
Stör-RAG=0
Meldung-An-XMIT + /Nicht-Genug-Zeit
Erzeuge DCN-Zu-griffrahmen
Ende-der-Meldung + /Nicht-Genug-Zeit
Setze keine Sende-Toggle-Empfangs-Toggle-Bits

FIG. 38