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

(10) **DE 697 27 906 T2** 2005.02.03

(12) # Übersetzung der europäischen Patentschrift

(97) **EP 0 864 964 B1**
(21) Deutsches Aktenzeichen: **697 27 906.5**
(96) Europäisches Aktenzeichen: **97 116 646.7**
(96) Europäischer Anmeldetag: **24.09.1997**
(97) Erstveröffentlichung durch das EPA: **16.09.1998**
(97) Veröffentlichungstag
der Patenterteilung beim EPA: **03.03.2004**
(47) Veröffentlichungstag im Patentblatt: **03.02.2005**

(51) Int Cl.[7]: **G06F 3/12**

(30) Unionspriorität:
816978 **13.03.1997** **US**

(73) Patentinhaber:
Hewlett-Packard Co. (n.d.Ges.d.Staates Delaware), Palo Alto, Calif., US

(74) Vertreter:
Schoppe, Zimmermann, Stöckeler & Zinkler, 82049 Pullach

(84) Benannte Vertragsstaaten:
DE, FR, GB

(72) Erfinder:
Snyders, Lawrence M., Boise, Idaho 83702, US

(54) Bezeichnung: **Geschalteter Druckertreiber in Windows-Betriebssystem**

Beschreibung

Gebiet der Erfindung

[0001] Diese Erfindung bezieht sich auf ein System für ein Drukkertreiberschalten innerhalb einer Computernetzwerkumgebung, wie beispielsweise eines Windows-Betriebssystems, um Druckaufträge von einer einzigen Druckanforderung innerhalb einer Anwendung an eines oder mehrere ungleiche Geräte zu richten.

Hintergrund der Erfindung

[0002] Es sind zentralisierte und dezentralisierte Computernetzwerke mit einer breiten Vielfalt von Peripheriegeräten erhältlich, die miteinander verbunden sind, so daß dieselben miteinander kommunizieren können. Für Anwendungen, die die Ausgabe von Daten von einem Computersystem erfordern, sind eines oder mehrere Ausgabegeräte in der Form eines Monitors, eines Druckers, eines Plattenlaufwerks oder eines anderen Peripheriegeräts bereitgestellt. Für den Fall von Computernetzwerken kann die Anzahl und Vielfalt von erhältlichen Ausgabegeräten ziemlich groß sein, was in Datentransferinkompatibilitätsproblemen resultiert.

[0003] Ein Problem, das aktuellen Computernetzwerkumgebungen zugeordnet ist, besteht darin, daß dieselben das automatische Senden von elektronischem Material von einer Anwendung zu ungleichen Bestimmungsorten von einer einzigen Quellanwendung oder einem Dokument nicht ermöglichen. Zum Beispiel ermöglichen gegenwärtige Fenstertreiberlösungen nicht, daß ein Operator automatisch einen Druckauftrag von einem Drukkertreiber zu einem jeglichen von mehreren Empfangsgeräten oder Druckern sendet. Anstelle dessen muß der Operator das System rekonfigurieren, um eine Ausgabe zu einem spezifischen Drucktreiber eines ausgewählten Ausgabegeräts zu liefern.

[0004] Bei gegenwärtigen Fenstertreiberlösungen bildet jeder Drukker ein Ausgabegerät, das eine zweckgebundene Druckmaschine aufweist, die ein zweckgebundenes Codieren für den zugeordneten Druckprozessor erfordert. Zum Beispiel erzeugt ein Druckprozessor eine Datei von Zeichnungsbefehlen für einen konfigurierten Drucker. Normalerweise ist eine permanente Verbindung zwischen der Anwendung (z. B. einem Textverarbeitungsprogramm), der Zwischenzeichnungsdatei und einem einzigen zweckgebundenen Ausgabegerät bereitgestellt. Optional kann der Benutzer eines von mehreren zweckgebundenen Ausgabegeräten über ein fensterbasiertes Menü auswählen. Eine derartige Konfiguration ist jedoch zu einer Verwendung mit einem Ausgabegerät formatiert, das der Benutzer ausgewählt hat, und bleibt zu einem derartigen Gerät zweckgebunden, bis dieselbe manuell rekonfiguriert wird.

[0005] Das Dokument US 5,511,156 lehrt ein Druckauftragverteilungssystem, bei dem Druckdateien in einem PostScript-Dateiformat beschrieben sind, wobei die PostScript-Druckdatei zu einem parallelen Übersetzungsverarbeiten durch mehrere Computer in unabhängige Abschnitte geteilt ist.

[0006] Daher existiert ein Bedarf nach einem System, das Druckaufträge von einem Computer, der innerhalb einer Computernetzwerkumgebung wirksam ist, zu einem jeglichen von mehreren ungleichen Ausgabegeräten von einer einzigen Druckanforderung innerhalb einer Anwendung verteilt.

[0007] Ein anderes Problem, das aktuellen Computernetzwerkumgebungen zugeordnet ist, ist die Unfähigkeit, ausgehende Quelldaten mehrere Male zu mehreren Ausgabegeräten zu senden. Zum Beispiel ermöglichen aktuelle Fenstertreiberlösungen es nicht, daß ein Operator die Quelldaten mehrere Male parst bzw. syntaktisch analysiert, um die notwendigen Codierungen für jedes Ausgabegerät zu erhalten. Anstelle dessen muß der Operator das System rekonfigurieren, um jede Lieferung einer Ausgabe zu jedem spezifischen Drucktreiber jedes ausgewählten Ausgabegeräts durchzuführen.

[0008] Diese Erfindung bezieht sich auf eine Informationsverteilungsvorrichtung und ein Verfahren, das die obigen Nachteile überwindet. Die Informationsverteilungsvorrichtung dieser Erfindung verbessert eine Verteilung von Quellaufträgen für eine Ausgabe zu einem jeglichen von mehreren ungleichen Ausgabegeräten. Die Informationsverteilungsvorrichtung dieser Erfindung verbessert auch eine Verteilung von Quellaufträgen zu mehreren empfangenden Ausgabegeräten von einer/einem einzigen Quellanwendung/-dokument.

Zusammenfassung der Erfindung

[0009] Gemäß einem Aspekt dieser Erfindung ist die allgemein in **Fig.** 1–5 gezeigte Informationsverteilungsvorrichtung innerhalb einer Computernetzwerkumgebung wirksam. Die Informationsverteilungsvorrichtung ge-

mäß der Erfindung ist in dem beigefügten unabhängigen Anspruch 1 definiert.

[0010] Gemäß einem anderen Aspekt dieser Erfindung implementiert die allgemein in **Fig.** 1–5 gezeigte Informationsverteilungsvorrichtung ein Verfahren zum Wirksamsein innerhalb einer Computernetzwerkumgebung, wie es in dem beigefügten unabhängigen Anspruch 13 definiert ist.

[0011] Andere Aufgaben, Merkmale und Vorteile werden aus der unten gegebenen detaillierten Beschreibung der Vorrichtung und des Verfahrens dieser Erfindung und Ausführungsbeispielen von Systemen ersichtlich, die die Vorrichtung und das Verfahren dieser Erfindung eingliedern.

Beschreibung der Zeichnungen

[0012] **Fig.** 1 ist ein konzeptionelles Blockdiagramm einer Computernetzwerkumgebung zum Implementieren des Druckertreiberschaltmechanismus und -verfahrens dieser Erfindung, das einen Entwurf einer Mehrzahl von Computern und Ausgabegeräten zeigt, die innerhalb der Netzwerkumgebung konfiguriert sind, gemäß einem Ausführungsbeispiel dieser Erfindung.

[0013] **Fig.** 2 ist ein schematisches Blockdiagramm, das die Struktur und Operationen des Druckertreiberschaltmechanismus von **Fig.** 1 darstellt, wobei ein mehrfaches Parsen des ausgehenden Dokuments in ein Format für eine Einfach-Teilprozeß-Spool-Lieferung eines Quellauftrags zu mehreren Bestimmungsorten gezeigt ist.

[0014] **Fig.** 3 ist ein schematisches Blockdiagramm, das die Struktur und Operation des Druckertreiberschaltmechanismus von **Fig.** 1 darstellt, wobei ein mehrfaches Parsen des ausgehenden Dokuments in ein Format für eine Mehrfach-Teilprozeß-Spool-Lieferung eines Quellauftrags zu mehreren Bestimmungsorten gezeigt ist.

[0015] **Fig.** 4 ist ein schematisches Blockdiagramm des ersten Ausführungsbeispiels eines Gerätetreiberschaltmechanismus während einer ersten Durchgangsoperation von einem ursprünglichen Gerät/Treiber zu Spool-verbesserten Metadateidaten zu einem Speicher.

[0016] **Fig.** 5 ist ein schematisches Blockdiagramm des ersten Ausführungsbeispiels eines Gerätetreiberschaltmechanismus während einer zweiten Durchgangsoperation für ein ausspulmäßiges Drucken (Despooling) von Druckauftrags- und Druckerinformationen über einen Druckprozessor zu einem neuen Gerätetreiber.

[0017] **Fig.** 6 ist ein schematisches Blockdiagramm eines ersten Ausführungsbeispiels eines Gerätetreiberschaltmechanismus während einer Mehrfach-Durchgang-Druckauftrag-Verteilung.

[0018] **Fig.** 7 ist ein Flußdiagramm, das die Sequenz von Schritten darstellt, die bei einem automatischen Implementieren eines Mechanismus zu einem Druckertreiberschalten in Windows-Betriebssystemen verwendet wird, um eine Verteilung von Druckaufträgen auf mehrere ungleiche Aufträge innerhalb einer Druckanforderung innerhalb einer Anwendung zu ermöglichen, bei einem Ausführungsbeispiel der vorliegenden Erfindung.

[0019] **Fig.** 8 ist ein Flußdiagramm, das eine erste Operation darstellt, die durch den Druckprozessor von **Fig.** 7 implementiert ist.

[0020] **Fig.** 9 und 10 stellen ein Flußdiagramm dar, das eine zweite Operation zeigt, die durch den Druckprozessor von **Fig.** 7 implementiert ist.

[0021] **Fig.** 11 ist ein Flußdiagramm, das eine dritte Operation darstellt, die durch den Druckprozessor von **Fig.** 7 implementiert ist.

Detaillierte Beschreibung der Erfindung

[0022] **Fig.** 1 zeigt einen Computer **10**, der als „Computer A" etikettiert ist und zu einer Verteilung von Informationen innerhalb einer Computernetzwerkumgebung **12** konfiguriert ist. Der Computer **10** kann elektronisches Material in der Form eines Quelldokuments oder -auftrags an mehrere Empfangsgeräte innerhalb der Netzwerkumgebung senden. Das Quelldokument wird ansprechend darauf erzeugt, daß eine Anwendung auf einem Betriebssystem des Computers **10** betrieben wird. Genauer gesagt stellt eine Mehrzahl von Ausgabegeräten **14**, **16** und **18**, die als „Ausgabegerät A", „Ausgabegerät B" bzw. „Ausgabegerät C" etikettiert sind, die Empfangsgeräte zu einem Empfangen von Ausgabebefehlsdateien und zum Erzeugen einer Ausgabe bereit.

Zusätzlich ist der Computer **10** mit anderen Computern **20** und **22** verbunden, die ähnliche zugeordnete Ausgabegeräte (nicht gezeigt) aufweisen. Der Computer **10** ist zu einem Senden eines Quelldokuments zu einem jeglichen dieser Ausgabegeräte in der Lage. Der Computer **10** ist zu einer Signalkommunikation mit jedem der Ausgabegeräte **14**, **16** und **18** über eine jeweilige Kommunikationsverbindung **24** zusammen verbunden oder verbunden. Auf eine ähnliche Weise ist der Computer **10** zu einer Signalkommunikation mit anderen Computern **20** und **22** über eine jeweilige Kommunikationsverbindung **26** zusammen verbunden oder verbunden. Die Verbindungen **24** und **26** können aus einer jeglichen einer Anzahl von gegenwärtig erhältlichen Draht- oder drahtlosen Signalverbindungen gebildet sein, die bei einem Bilden von Computernetzwerkverbindungen verwendbar sind. Eine derartige Verbindung ist durch ein Festverdrahten jeweiliger Komponenten des Netzwerks **12** miteinander gebildet.

[0023] Gemäß der Netzwerkumgebung **12**, die in **Fig.** 1 dargestellt ist, ist klar, daß die Ausgabegeräte **14**, **16** und **18** und die Computer **10**, **20** und **22** zweckgebundene Prozessoren zum Durchführen von Verarbeitungs- und Kommunikationsbedürfnissen aufweisen, wenn Quellaufträge und Ausgabebefehle zwischen Geräten des Netzwerks übertragen werden. Die Computer **10**, **20** und **22** weisen ein Windows 95 Betriebssystem und einen kundenspezifischen Druckprozessor dieser Erfindung zu einer Verwendung mit Windows 95 auf. Die Ausgabegeräte **14**, **16** und **18** weisen den gleichen kundenspezifischen Druckprozessor auf, was ein Drucken direkt von einer Anwendung, die auf dem Betriebssystem des Computers **10** läuft, zu einem jeglichen der Ausgabegeräte **14**, **16** oder **18** ermöglicht.

[0024] Der Computer **10** enthält vorzugsweise ein Windows 95 Betriebssystem. Ein „Betriebssystem" ist ein Satz von Computerprogrammen, die spezifisch durch einen Computer zu einem Verwalten der Betriebsmittel desselben verwendet werden. Das Betriebssystem steuert alle Betriebsmittel des Computersystems, einschließlich einer Kommunikation zwischen einem Benutzer und dem Computer. Mehrere exemplarische Betriebssysteme umfassen DOS, Windows, OS/2, UNIX und das MacIntosh-System. Vorzugsweise umfaßt das Windows 95 Betriebssystem ferner die „Microsoft Windows 95 Device Driver Developers Kit" (Microsoft Windows 95 Gerätetreiberentwicklerausrüstung), die von Microsoft Corporation durch ein Bestellen von „Microsoft Developer Network Professional Subscription" derselben erhalten werden kann. „Microsoft Developer Network Professional Subscription" umfaßt CDROMs mit einem Muster-Quellcode und einer Dokumentation, die beschreibt, wie Anwendungen und Gerätetreiber zu entwickeln sind. Eine Komponente der Driver Developers Kit (DDK) (Treiberentwicklerausrüstung) schildert genau, wie Drucksystemkomponenten zu entwickeln sind. Ein Druckprozessor ist eine Drucksystemkomponente, die in der Treiberentwicklerausrüstung dokumentiert ist. Das Abonnement umfaßt alle Dokumentationen hinsichtlich wie die Entwicklerausrüstung zu verwenden ist.

[0025] Gemäß der Implementierung dieser Erfindung wird eine Anwendung auf dem Betriebssystem des Computers **10** betrieben, wie es in **Fig.** 1 gezeigt ist. Eine „Anwendung" ist ein Computerprogramm, das konfiguriert ist, um bei einem Durchführen einer bestimmten Art einer Arbeit zu unterstützen. Im Gegensatz dazu betreibt ein Betriebssystem einen Computer, ein Hilfsprogramm führt eine Wartung oder allgemeine Arbeiten durch und eine Sprache wird verwendet, um Computerprogramme zu erzeugen. Basierend auf dem speziellen Entwurf derselben kann eine Anwendung Text, Graphik, Zahlen oder eine Kombination dieser Elemente manipulieren.

[0026] Das Betriebssystem des Computers **10** und des Netzwerks **12** umfaßt eine Anwendungsprogrammierungsschnittstelle (API = application programming interface). Eine API weist einen definierten Satz von Funktionen auf, die durch das Betriebssystem zu einer Verwendung durch eine Anwendung bereitgestellt sind. Die Schnittstelle existiert in der Form eines definierten Satzes von Funktionen zu einer Verwendung, wo es notwendig ist, daß proprietäre Anwendungsprogramme mit einer Kommunikationssoftware sprechen oder konform zu Protokollen von einem Produkt eines anderen Verkäufers sind. Eine API stellt ein standardisiertes Verfahren von vertikalen Kommunikationen innerhalb und außerhalb des Computernetzwerks bereit.

[0027] Für den Fall, bei dem die Ausgabegeräte **14**, **16** und **18** Drukker sind, weist jeder Drucker einen Druckertreiber auf. Ein „Druckertreiber" ist ein Softwareprogramm, das es ermöglicht, daß andere Programme mit einem speziellen Drucker arbeiten, ohne sich mit den Spezifika der Hardware und internen Sprache des Druckers zu befassen. Jeder Drucker erfordert einen spezifischen Satz von Codes und Befehlen, um ordnungsgemäß wirksam zu sein und einen Zugriff auf spezielle Merkmale und Fähigkeiten bereitzustellen. Wo alternativ das Ausgabegerät kein Drucker ist, weist das Ausgabegerät einen Gerätetreiber auf, der wie ein Druckertreiber wirkt.

[0028] Ein „Treiber", wie bei einem Gerätetreiber oder einem Drukkertreiber darauf Bezug genommen wird, ist ein Programm oder Teilprogramm, das beschrieben ist, um entweder ein spezielles Hardwaregerät oder

eine andere Softwareroutine zu steuern. Der Ausdruck „Treiber" stammte von dem Konzept von Trabrenntreibern oder Automobiltreibern, die die Rösser oder Autos derselben auf Herz und Nieren prüften, um die Fähigkeiten derselben zu messen. Die allgemeinsten Beispiele eines Hardwaretreibers, einer, der eine Hardware steuert, sind diejenigen, die zu speziellen Marken und Modellen von Druckern gehören, die an Personalcomputern angeschlossen sind. Zum Beispiel ermöglicht es ein spezifischer Drukkertreiber, daß ein Textverarbeitungsprogramm mit einem Punktmatrixdrucker oder einem Laserdrucker eines speziellen Modells kommuniziert.

[0029] Ein anderer wichtiger Aspekt betrifft den Bedarf nach einer Software in der Form einer Druckertreibersoftware zu einem Unterstützen von Ausgabegeräten. Bis zu diesem Punkt wurden lediglich die mechanischen Aspekte eines Druckers detailliert geschildert. Auf einem anderen Pegel wird das Problem durch die große Vielfalt von Druckern mit allen Arten von speziellen Merkmalen komplexer gemacht, die durch eine breite Vielfalt von Herstellern hergestellt werden.

[0030] Bei der frühen Entwicklung von Druckern war die Situation ziemlich einfach. Die erhältlichen Drucker sprachen auf ASCII-Text- und -Steuerschriftzeichen an. Für jedes zu dem Drucker gesendete Schriftzeichen wurde ein Buchstabe erzeugt. Eine Vorbewegung zu der nächsten Zeile eines Texts wurde mittels eines Wagen-Rückkehr- und Zeilenzufuhr-Schriftzeichens vorgenommen. Falls ein Überdrucken für fettgedruckte Wirkungen benötigt wurde, wurde lediglich ein Wagen-Rückkehr-Schriftzeichen gesendet und die Zeile wurde wieder gedruckt. Schreibmaschinenähnliche Drucker waren ebenfalls erhältlich, die auf ein Rückwärtsschritt-Schriftzeichen ansprachen, so daß ein Fettdrucken und Unterstreichen ohne weiteres erzielt wurden.

[0031] In dem Maße wie eine Druckertechnologie mit der Entwicklung von Punktmatrixdruckern fortschritt, wurde die Fähigkeit für Schriftartänderungen und graphische Optionen machbar. Das Ergebnis dieses Fortschritts bestand darin, daß Textverarbeitungsprogramme einen Satz einer Software für jeden verfügbaren Drucker benötigten, da jeder Hersteller seine eigenen Gedanken hatte, welche Merkmale nützlich sein könnten, jeder Hersteller dieselben auf eine unterschiedliche Weise implementierte. Mit jeder neuen Druckerausführung oder jedem Modell wurde ein neuer Satz einer Software benötigt, die als Treibersoftware bezeichnet wird. Folglich war ein mißliches Problem sowohl für den Benutzer des Textverarbeitungsprogrammdruckers als auch für Textverarbeitungsprogrammverkäufer erzeugt.

[0032] In dem Maße wie der Bedarf nach komplizierteren Druckfähigkeiten wuchs, hat sich eine Anzahl von Lösungen entwickelt. Das Hauptproblem war ein Entscheiden, wohin die „Intelligenz" für den Druckprozeß zu setzen ist. Es gibt drei Möglichkeiten:

1. In das Textverarbeitungsprogramm selbst. Das Textverarbeitungsprogramm wird etwas schwierig zu verwenden, wenn keine schnelle CPU und keine gute Schnittstelle verfügbar sind. Unter anderem führt dies zu einem proprietärem Dateiformat für das Textverarbeitungsdokument. Es wird schwierig, stark formatierte Dokumente zwischen unterschiedlichen Plattformen oder Textverarbeitungsprogrammen zu bewegen, wenn nicht ein gemeinsames Datenformat verwendet werden kann, wie beispielsweise RTF oder DCA. Falls keine gemeinsame formatierte Datenstruktur gefunden werden kann, ist es eventuell möglich, die Dateien in einem „generischen" Format zu übertragen, bei dem eine Wagenrückkehr/Zeilenzuführung lediglich an dem Ende eines Absatzes verwendet wird. Es wird dann nötig, die Datei umzuformatieren.
2. In den Drucker. Dies führt zu dem, was als Seitenbeschreibungssprachen (PDLs = Page description languages) bekannt ist. Die häufigste derselben ist eine, die als PostScript bekannt ist. Der Drucker enthält einen Prozessor und ein Programm, um die Befehle zu interpretieren, die in der PostScript-Ausgabedatei enthalten sind. Die PostScript-Ausgabedatei ist eine ASCII-Textdatei, die falls nötig manuell editiert werden kann. Es sind auch andere proprietäre Seitenbeschreibungs- und Steuersprachen erhältlich, wie beispielsweise dieselben in der Hewlett-Packard Linie von Laserdruckern. Die PostScript-Implementierung wird durch ein Kaufen und Installieren eines PostScript-Sprachchips für den Drucker aktiviert.
3. Eine andere Alternative. Eine letzte Alternative, zwischen diesen zwei, ist ein Schriftsetzprogramm, wie beispielsweise TeX, das eine Datei erzeugt, die die Schriftsatzbefehle enthält. Diese Datei wird dann in ein geräteunabhängiges Format umgewandelt, das durch ein Treiberprogramm für den speziellen Drucker geleitet werden muß.

[0033] Zusätzlich zu einer Plazierung der Intelligenz ist eine Graphikunterstützung auf zwei allgemeine Weisen implementiert:

1. Im wesentlichen eine Graphik-„Abladung" des Textverarbeitungsbildschirms. Der notwendige Druckertreiber wird wiederum für den verwendeten Drucker benötigt. Es ist auch möglich, das Werk innerhalb des Dokuments selbst einzubetten, wie beispielsweise bei Microsoft Word.
2. Eine „verkapselte" PostScript-Datei. Dies enthält den durch das Zeichnungsprogramm erzeugten Code

in einem Format, das interpretiert werden soll. Eine verkapselte PostScript-Datei kann ferner ein Vorschaubild des Bilds in einem Rasterformat, wie beispielsweise TIFF, oder einem Vektorformat, wie beispielsweise einer Windows-Metadatei für eine einfache aber direkte Bildschirmmanipulation des Bilds enthalten.

[0034] Welches Verfahren gewählt wird, ist wiederum eine Frage einer persönlichen Präferenz und der speziellen erhältlichen Implementierung. Zu Implementierungszwecken des Geräts und des Verfahrens von **Fig.** 1–11 der Anmelderin wird ein kundenspezifischer Druckprozessor zu einer Verwendung mit einem Betriebssystem, wie beispielsweise Windows 95, mit einem Druckertreiber für das Betriebssystem verwendet, das ein Erzeugen von verbesserten Windows-Metadateien unterstützt.

[0035] **Fig.** 2 und 3 stellen Weisen zu einem Liefern eines Quellauftrags **28** zu einem oder mehreren Bestimmungsorten **30**, **32** und **34** dar, wobei ein Beispiel eines Bestimmungsorts die Ausgabegeräte **14**, **16** und **18** (von **Fig.** 1) sind. Ein anderes Beispiel eines Bestimmungsorts könnten die Computer **20** und **22** sein. Derzeitige Windows-Treiberlösungen ermöglichen kein Senden von einer Anwendung zu mehreren Bestimmungsorten. Der Treiber muß jedoch in der Lage sein, elektronisches Material zu mehreren Empfangsgeräten von einer/einem einzigen Quellanwendung/-dokument zu senden, um wirksam zu sein. Dies erfordert ein Sichern der ausgehenden Dokumentdaten oder des Quellauftrags **28** in einem Format, das mehrere Male geparst werden kann, um die erwünschten Codierungen zu erhalten, wie es in **Fig.** 2 gezeigt ist. Falls das Quelldokument oder der Quellauftrag **28** mehrere Seiten ist, dann muß der Mechanismus zu einem Senden ein Senden aller Seiten eines Bestimmungsorts vor einem Übergehen zu dem nächsten Codiertyp unterstützen, wie es in **Fig.** 3 gezeigt ist. Ein Mehrfach-Teilprozeß-Lieferungsmechanismus ist notwendig um die Quellauftragslieferung zu mehreren Bestimmungsorten zu implementieren, wie es in **Fig.** 3 gezeigt ist.

[0036] Wie es in **Fig.** 2 und 3 gezeigt ist, bestehen mehrere Optionen zu einem Liefern eines Quellauftrags **28** zu mehreren Bestimmungsorten **30**, **32** und **34**. Das Folgende ist eine Zusammenfassung der Optionen zu einem Senden:

1) Alle Seiten eines Auftrags können zu einem speziellen Bestimmungsort vor einer Lieferung zu anderen Bestimmungsorten gesendet werden. Diese Option wird als eine Seriell-Lieferung zu einem Bestimmungsort zu einer Zeit über eine Einfach-Teilprozeß-Spool-Lieferung bezeichnet. Eine derzeitige Windows-Treiberarchitektur verwendet eine derartige Einfach-Teilprozeß-Spool-Lieferung.

2) Eine Lieferung kann als eine Seite zu einer Zeit zu jedem (allen) Bestimmungsort(en) vor einem Übergehen zu der nächsten Seite des Quellauftrags implementiert sein. Probleme können jedoch eine Lieferung verzögern, was bedeutet, daß dasselbe oft keine gute Lieferungswahl ist.

3) Es kann ein Teilprozeß für jeden Bestimmungsort vorgesehen sein (Mehrfach-Teilprozeß), wobei der gesamte Auftrag durch jeden Teilprozeß geliefert wird. Auf diese Weise können alle Teilprozesse simultan laufen, was in einer parallelen Ausführung resultiert. Dies erfordert ein Ändern von Windows-Despooling-Mechanismen, aber bietet die beste Lieferung zu allen Empfängern.

[0037] Die Implementierung eines Zwischendateiformats, vorzugsweise in einer Form als eine verbesserte Metadatei (EMF = enhanced metafile) ausgeführt, würde die mehreren notwendigen Durchläufe ermöglichen, um die oben erwähnten Lieferungsmerkmale von **Fig.** 2 und 3 zu implementieren. Eine verbesserte Metadatei wird jedoch typischerweise über einen Spooler zu einem Speicher geschrieben und eine Spooler-Wiedergabe einer verbesserten Metadatei ermöglicht keine Mehrfach-Durchlauf-Lieferung von dem Speicher. Eine verbesserte Metadatei (EMF) ist ein verbessertes Dateiformat, das eine Reihe von graphischen Operationen in einem geräteunabhängigen Datenformat auf hoher Ebene beschreibt.

[0038] Um ein derartiges Zwischendateiformat zu implementieren, wird ein spezifischer Druckprozessor eines Entwurfs der Anmelderin dieses Problem durch ein Kopieren der verbesserten Metadatei lösen. Wie es unten mit Bezug auf **Fig.** 4–6 gezeigt und beschrieben wird, wird ein derartiger kundenspezifischer Druckprozessor ermöglichen, daß mehrere Durchläufe von einem Speicher über einen Spooler/Despooler geliefert werden, um einen Quellauftrag zu mehreren Bestimmungsorten über die Lieferungsschemata zu liefern, die oben mit Bezug auf **Fig.** 2 und 3 offenbart sind. Ein „Spooler" ist eine Komponente, die eine anwendungserzeugte Ausgabe, die für einen Drucker bestimmt ist, nimmt und dieselbe temporär auf einer Platte speichert. Ein „Despooler" ist eine Systemkomponente, die für Daten in Spool-Dateien und ein Übergeben derselben zu der Software verantwortlich ist, die zu einem Schreiben derselben zu einem Ausgabegerät verantwortlich ist.

[0039] Um dies jedoch zu machen, benötigt das Gerät dieser Erfindung eine zweckgebundene Spool-Auftrag-Kopfblock-Datei, um die Bestimmungsorte und aufbereiteten Codierungen für jeden Bestimmungsort zu beschreiben. Wenn eine Despool-Operation implementiert ist, sendet eine Graphikgerätschnittstelle (GDI = Graphics Device Interface) graphische Funktionen innerhalb eines Gerätekontexts (DC = device context) durch

den Druckertreiber zu dem Spooler. Eine „GDI" ist eine Graphikgerätschnittstelle (GDI = Graphics Device Interface), die Komponente von Windows, die zu einem Implementieren der graphischen Funktionen verantwortlich ist, wie beispielsweise einem Linienzeichnen und einer Farbverwaltung. Eine GDI ist eine DLL (Dynamic Link Library = dynamische Verknüpfungsbibliothek), die alle der graphischen Anwendungsprogrammierungsschnittstellen (APIs) in Windows umfaßt. Ein Gerätekontext (DC) ist eine GDI-Datenstruktur, die den aktuellen Zustand eines Geräts oder einer Zeichnungsoberfläche beschreibt, ein logisches Objekt, das auf der Anwendungsebene zu finden ist. Genauer gesagt ist ein Gerätekontext eine Struktur, die intern in einer GDI zu dem Zweck eines Anzeigens der graphischen Daten (Malen auf dem Bildschirm oder Drucken von Seiten) beibehalten wird. Ein jeglicher Anwendungsentwickler, der sich mit einem Drucken befaßt, ist mit dem Konzept eines Gerätekontexts vertraut. Eine dynamische Verknüpfungsbibliothek (DLL) ist eine Bibliothek von gemeinschaftlich verwendeten Funktionen, mit denen sich Anwendungen zu einer Ausführungszeit verbinden, im Gegensatz zu einer Kompilierzeit. Eine einzige speicherinterne Kopie der dynamischen Verknüpfungsbibliothek (DLL) befriedigt Anforderungen von allen rufenden Teilnehmern.

[0040] Bei Quelldokumenten, die lediglich eine Seite in einer Länge sind, ist es möglich, anstelle der verbesserten Metadatei (EMF) alternativ Rohdaten zu verwenden, die in die Sprache des Druckers formatiert sind. In diesem Fall ist es jedoch notwendig, daß alle Bestimmungsorte die gleiche Codierung unterstützen und wählen. Um dies zu machen, wird eine Unterstützung der verbesserten Metadatei in einer GDIINFO-Struktur markiert.

[0041] Die Vorrichtung und das Verfahren dieser Erfindung liefern Daten zu Geräten, einschließlich Ausgabegeräten, wie beispielsweise Drucker, Scanner und Plotter, in einer Verteilungsliste. Die Verteilungsliste ist eine Liste von Geräten, die eine Ausgabe empfangen können. Eine Verteilungsliste kann durch eine getrennte Anwendung gesteuert sein, mit der der Druckprozessor kommuniziert. Die getrennte Anwendung könnte eine jegliche einer Anzahl von Geräten sein, die innerhalb einer Computernetzwerkumgebung vorhanden sind. Der Druckprozessor verwendet dann die Liste von Geräten bei einem Implementieren einer Operation eines Treiberschaltens gemäß dem Gerät und dem Verfahren dieser Erfindung.

[0042] Die Anmelderin hat die Operation des Treiberschaltens gemäß dieser Erfindung durch ein Betreiben des Programms und der Datei, die in Tabelle 1 unten bereitgestellt sind, getestet. Weitere Details der Implementierung werden unten mit Bezug auf **Fig.** 4–6 und das Flußdiagramm von **Fig.** 7–11 erörtert. Die Anmelderin betrieb einen derartigen Test auf einem Windows 95 System. Ein EMF-Auftrag wurde in eine Warteschlange mit fünf Treibern gegeben, dann wurden die Treiber durch eine Benutzerschnittstelle (UI = user interface) zu einem Bitabbildungstreiber geschaltet. Der resultierende Quellauftrag wurde verarbeitet, geschaltet und freigegeben, wobei der Druckauftrag ordnungsgemäß zu einer Datei druckt.

[0043] Das unten in Tabelle 1 aufgelistete C-Code-Programm liefert einen „kundenspezifischen Druckprozessor zu einem Treiberschalten" gemäß dieser Erfindung. Dieser Code, wie derselbe in den Flußdiagrammen von **Fig.** 7–11 genau geschildert ist, ermöglicht das Schalten von Komponenten in der Form von Ausgabegeräten zu einem Empfangen eines Druckauftrags. Das Schalten von Komponenten ist über einen kundenspezifischen Druckprozessor für Windows 95, einen Treiber, der eine Windows-Bitabbildung erzeugt, und einen Treiber, der eine HP PCL5-Druckersprache erzeugt, implementiert. Um dies wirksam vorzunehmen, muß ein Druckertreiber für Windows 95 oder NT ein Erzeugen von verbesserten Windows-Metadateien (EMFs) unterstützen. Ein Operator muß lediglich eine Windows-Anwendung betreiben und mit dem erwünschten anfänglichen Druckertreiber das erwünschte Druckobjekt auswählen. Dieser Treiber muß jedoch den kundenspezifischen Druckprozessor aufweisen, der demselben zugeordnet ist. Der Operator, oder das Netzwerk über ein gewisses Prioritätensetzungsschema, kann dann den Druckauftrag direkt von der Anwendung drucken. Der resultierende Druckauftrag wird temporär in dem Spooler gespeichert. Wenn der Auftrag zeitplanmäßig drucken soll, ruft der Spooler den kundenspezifischen Druckprozessor auf, um den Auftrag zu drucken. Der Druckprozessor fragt das System ab, um zu bestimmen, ob die Treiber existieren. Derselbe öffnet dann den durch die Anwendung ausgewählten Treiber und erlangt die aktuelle PRINTER.INFo.2-Struktur, um den Treibernamen und andere Informationen über den Drucker zu erlangen.

[0044] Der Druckprozessor ändert dann den Treibernamen durch ein Verwenden von „SetPrinter API" mit der PRINTER.INFO.2-Struktur und einem neuen Treibernamenfeld in den neuen erwünschten Namen. Dann erlangt der Druckprozessor unter Verwendung von „Document Properties" die DEVMODE-Struktur des neuen Treibers, so daß dieselbe durch den neuen Treiber verwendet und verstanden wird. Dies wird auch auf der PRINTER.INFO.2-Struktur gesichert. Das Obige wird alles gemacht, wenn dem Druckprozessor durch den Spooler befohlen wird, „den Prozessor zu öffnen". Der Prozessor wird dann durch den Spooler angerufen, um die Daten zu verarbeiten. Der Prozessor ruft die Graphikgerätschnittstelle (GDI) an, um die Daten der verbes-

serten Metadatei (EMF) in dem neuen Treiber zu verarbeiten. Am Ende des Auftrags wird der Treiber zu dem Ursprung rückgesetzt, so daß dem Benutzer immer die gleichen Informationen vorgelegt werden.

[0045] Um den obigen Prozeß zu implementieren, müssen mehrere Beschränkungen angewendet werden. Erstens kann sich der DEVMODE für einen jeglichen verwendeten Treiber nicht auf private DEVMODE-Daten stützen, wenn nicht alle Treiber die gleichen privaten Daten unterstützen, wie dieselben in dem Verbesserte-Metadatei- (EMF-) Auftrag enthalten sind. Zweitens sollte die Benutzerschnittstelle für alle Treiber gleich aussehen, weil der Benutzer jederzeit aufgerufen werden könnte.

[0046] Um ein Treiberschalten gemäß der Vorrichtung und dem Verfahren dieser Erfindung besser zu verstehen, wird ein Schritt-für-Schritt-Fluß durch das System, wie es über den Algorithmus von Tabelle 1 implementiert ist, mit Bezug auf **Fig.** 4–6 durchgesehen. Ein grundlegender Vorteil des Treiberschaltens gemäß dieser Erfindung besteht darin, eine Systemwahl eines unterschiedlichen Druckertreibers als dem einen zu ermöglichen, der verwendet wird, um einen Druckauftrag in Windows 95 oder NT zu erzeugen (dies gilt konzeptionell auch für den OS/2-Queue Processor (Wartschlangentreiber)). Ein Wählen eines unterschiedlichen Treibers ermöglicht, daß der einfach-gespoolte Auftrag zu mehreren ungleichen Geräten gesendet wird, wie beispielsweise einem HP DeskJet Drucker, einem LaserJet-Drucker oder einem Plotter. Derselbe kann auch mehrere Male verarbeitet werden, um zu einer Verteilung von Geräten zu senden, wie beispielsweise vernetzte Drucker oder Internetgeräte.

[0047] Eine Analyse des Treiberschaltauftragsflusses, der in **Fig.** 4–6 dargestellt ist, beginnt mit einer Erstdurchlauf-Druckanforderung, die über eine Anwendung **36** eingeleitet wird. Die Anwendung **36** erzeugt einen Druckauftrag oder einen Quellauftrag, der zu der dynamischen Verknüpfungsbibliothek (DLL) einer Graphikgerätschnittstelle (GDI) geliefert wird. Die Graphikgerätschnittstelle (GDI) implementiert gemeinschaftlich verwendete Funktionen zu einer Verbindungszeit zwischen der Anwendung **36** und einem ursprünglichen Treiber **38**. Die Graphikgerätschnittstelle (GDI) enthält alle graphischen Anwendungsprogrammierungsschnittstellen (APIs) in Windows, wobei gespoolte Daten in der Form einer verbesserten Metadatei **42** erzeugt werden.

[0048] Gemäß einer Zweitdurchlaufoperation des Treiberschaltens stellt **Fig.** 5 einen Spooler **46** dar, der über einen Spool-Kopfblock **48** die Speicherung und Wiedererlangung der gespoolten Druckdaten oder der verbesserten Metadatei **42** von einem Speichergerät **44** anweist. Der Spooler **46** leitet Druckauftrags- und Druckerinformationen zu dem kundenspezifischen Druckprozessor **50** dieser Erfindung. Genauer gesagt setzt der Druckprozessor **50** über die Codeimplementierung von Tabelle 1 und **Fig.** 7–11 den Treiber von dem ursprünglichen Treiber **38** (von **Fig.** 4) rück. Eine Registerdatenbank **54** speichert Hardware- und Softwarekonfigurationsinformationen, die durch die Systemanwendungsprogrammierungsschnittstellen (APIs) wiedererlangt und verwendet werden, um den Druckprozessor **50** bei einem Rücksetzen des Treibers zu unterstützen. Der Druckprozessor **50** weist die Graphikgerätschnittstelle (GDI) an, den Druckauftrag auf einem neuen oder rückgesetzten Treiber zu liefern. Die verbesserte Metadatei (EMF) **42** wird durch die Graphikgerätschnittstelle (GDI) von dem Speichergerät **44** wiedererlangt, um einen Teil des „GDI-Abspielen-Spool-Stroms (GDI play spool stream)" zu bilden. Die Graphikgerätschnittstelle (GDI) weist eine Kommunikation mit den neuen Treiber **56** und einem Tor-Überwachungs/Druck-Anbieter **58** eines Ausgabegeräts **14** an. Das Gerät **14** empfängt nachfolgend Ausgabebefehle von dem Treiber **56**, die von der Ausgabebefehlsdatei umgewandelt sind, die eine Zeichnungsbefehlsdatei aufweist, die Zeichnungsbefehle enthält, die von dem Druckprozessor **50** empfangen werden. Der Druckprozessor **50** führt die Ausgabebefehlsdatei basierend auf einem Prioritätensetzungsschema dem Ausgabegerätetreiber **56** zu.

[0049] Ein geeignetes Prioritätensetzungsschema betrifft ein Richten der Ausgabebefehlsdatei von dem Druckprozessor **50** zu einem des zumindest einen Ausgabegeräts basierend auf den Druckfähigkeiten des Ausgabegeräts **14**. Eine andere Weise besteht darin, ein Richten der Ausgabebefehlsdatei zu einem Ausgabegerät auf eine relative Verfügbarkeit des Ausgabegeräts **14** und/oder des verwandten Gerätetreibers **56** zu basieren. Zum Beispiel könnten ein spezieller Treiber **56** und der Drucker **14** bereits zu viele Druckaufträge in einer Warteschlange enthalten. Alternativ könnten der Treiber **56** und der Drucker **14** kein Papier oder keine Tinte mehr haben. Des weiteren könnten der spezielle Treiber **56** und der Drucker **14** erwünschte Druckfähigkeiten nicht aufweisen, wie beispielsweise eine Farbe, ein korrekt proportioniertes Papier, eine Bildauflösung, etc., was es vernünftiger macht, den Prozessor **50** aufzuweisen, um den Druckauftrag zu einem anderen geeigneteren Treiber und Gerät basierend auf einem oder einem jeglichen einer Anzahl von bekannten Prioritätssetzungsschemata zu liefern.

[0050] In Anbetracht eines speziellen Treibers **56** und eines Geräts **14**, die verwendet werden, hängt die Tatsache, wie jedes Verfahren arbeitet, von dem durch das Textverarbeitungsprogramm verwendeten Druckertrei-

ber ab. Ein Treiber ist eine Erweiterung zu dem Betriebssystem und ist für ein spezifisches Stück einer Hardware maßgeschneidert, wie beispielsweise einen Hewlett-Packard LaserJet III Drucker oder einer Super-VGA-Anzeige. Der Druckertreiber nimmt die Informationen, die durch den Reihenstrom von Daten oder die Seitenbeschreibungssprache geliefert werden, und wandelt dieselben in Befehle auf niedriger Ebene um, die durch den Drucker erkannt werden. (Ein Bildschirmtreiber macht das gleiche mit dem Videoadapter, um Text auf dem Monitor anzuzeigen.) Durch ein Arbeiten mit unterschiedlichen Treibern kann ein Textverarbeitungsprogramm mit dem gleichen Dokument an einer Vielfalt von Anzeigen und Druckern arbeiten. Eine „Registerdatenbank" ist eine strukturierte Datei in Windows, die indexierte Informationen speichert, die die Hardware des Hostsystems, Benutzerpräferenzen und andere Konfigurationsdaten beschreiben. Die Registerdatenbank dient dazu, die starke Vermehrung von Konfigurationsdateien zu reduzieren, die eine Windows-Maschine stören kann.

[0051] **Fig.** 6 stellt ein Treiberschalten zu einem Verteilen von Mehrfachdurchlaufaufträgen an eines oder mehrere Ausgabegeräte dar. Der Druckprozessor **50** wird gemäß Schritt „S7.20", der unten mit Bezug auf **Fig.** 9 beschrieben ist, durch den Spooler **46** „geöffnet", um die ursprüngliche verbesserte Metadatei (EMF) **42** zu kopieren, die in dem Speichergerät **44** gespeichert ist. Über einen Block **50** und einen Schritt „S7.30" von **Fig.** 11 weist der Spooler **46** ferner den Druckprozessor an, das Dokument an dem Druckprozessor zu drukken. Der Operationsblock **50** implementiert eine geschlossene Schleife **62**, die zu einem neuen Druckauftrag **64** zeigt, und erlangt die verbesserte Metadatei **42** wieder und schreibt dieselbe zu dem Speicher **44**, die über den Druckprozessor einer Lieferung zu jedem einer Reihe von Ausgabegeräten gemäß der Schleife **62** und einem Zeiger **64** wieder wiedererlangt wird.

Logisches Flußdiagramm

[0052] Gemäß **Fig.** 7 ist ein „Druckprozessor" als ein logisches Flußdiagramm einer ersten Ebene für das Programmieren eines Computers zu einem Wirksamsein innerhalb einer Computernetzwerkumgebung offenbart, um Druckaufträge zu mehreren Ausgabegeräten aufzubereiten. Tabelle 1 schildert ferner genau die Implementierung des Druckprozessors, aufbereitet in C-Code-Sprache, wie es oben mit Bezug auf **Fig.** 5 und 6 gezeigt ist. Der „Druckprozessor" bildet ein Gerät, das ein Druckertreiberschalten zu einer Verwendung mit einem Computer implementiert, der innerhalb einer Computernetzwerkumgebung wirksam ist. Der „Druckprozessor" kann gemäß dem logischen Flußdiagramm von **Fig.** 7–11 automatisch implementiert sein. Alternativ kann der kundenspezifische Druckprozessor HP PCL5 erzeugen. Auf diese Weise unterstützt der Druckprozessor das Erzeugen von verbesserten Windows-Metadateien (EMFs). Ein Benutzer betreibt eine Windows-Anwendung und wählt mit einem erwünschten ursprünglichen Druckertreiber ein erwünschtes Druckobjekt aus. Die Gerätetreiber, die in der Computernetzwerkumgebung wirksam sind, müssen den kundenspezifischen Druckprozessor aufweisen, der denselben zugeordnet ist. Druckaufträge werden von der Anwendung gedruckt. Der resultierende Druckauftrag wird über den Spooler temporär in einem Speicher gespeichert. Wenn der Auftrag zeitplanmäßig drucken soll, ruft der Spooler den kundenspezifischen Druckprozessor auf, den Auftrag zu drucken. Der Druckprozessor fragt das System ab, um zu bestimmen, ob die Treiber existieren. Derselbe öffnet dann den durch die Anwendung ausgewählten Drucker und erlangt die aktuelle PRINTER.INFO.2-Struktur, um den Treibernamen und andere Informationen über den Drucker zu erlangen. Der Druckprozessor ändert dann den Druckernamen in einen neuen erwünschten Namen, wie es vorhergehend offenbart ist.

[0053] Gemäß einem Schritt „S1" werden die Betriebssysteme auf den Computern innerhalb des Computerbetriebsnetzwerks und die Betriebssysteme und Gerätetreiber der Ausgabegeräte gestartet. Zum Beispiel startet auf ein Hochfahren jedes Geräts in dem Netzwerk hin das System-BIOS das Betriebssystem und den Spooler, die verwendet werden können, um das Flußdiagramm von **Fig.** 7 und 8 automatisch zu initialisieren oder die Einleitung desselben zu bewirken, was bewirkt, daß der Computer eine verbesserte Metadatei erzeugt, ursprüngliche Druckertreibergeräteinformationen sammelt und den Drucker (oder das Ausgabegerät) zu einem Empfangen der Metadatei schaltet. Nach einem Durchführen des Schritts „S1" geht der Prozeß zu einem Schritt „S2" über.

[0054] Bei dem Schritt „S2" leitet die Anwendung, die auf dem Betriebssystem läuft, einen Druckauftrag ein. Ein Weg besteht darin, automatisch eine Druckanforderung zu erzeugen. Ein anderer Weg besteht darin, daß ein Benutzer einen Druckauftrag anfordert. Nach einem Durchführen des Schritts „S2" geht der Prozeß zu einem Schritt „S3" über.

[0055] Bei dem Schritt „S3" leitet die Anwendung, die auf dem Betriebssystem läuft, den Druckauftrag zu der Graphikgerätschnittstelle (GDI) weiter, eine Komponente des Windows-Betriebssystems, die für ein Implemen-

tieren der graphischen Funktionen verantwortlich ist, wie beispielsweise ein Linienzeichnen und eine Farbverwaltung. Die GDI erzeugt Daten in Zwischenzeichnungsbefehlen in der Form von verbesserten Metadateien. Nach einem Durchführen des Schritts „S3" geht der Prozeß zu einem Schritt „S6" über.

[0056] Bei einem Schritt „S4" liefert der ursprüngliche Druckertreiber Geräteinformationen, die notwendig sind, um das zugeordnete Peripheriegerät zu betreiben, wie beispielsweise einen Drucker, einen Monitor oder ein anderes Ausgabegerät. Der Druckertreiber liefert Informationen über Fähigkeiten des Geräts, das derselbe darstellt. Zum Beispiel sind die Größe eines verwendeten Papiers, eine Unterstützung von Farben, Schwarz-Weiß, etc. einige der Informationen über Gerätefähigkeiten, die geliefert werden können. Nach einem Durchführen von Schritten „S4" und „S5" geht der Prozeß zu Schritten „S5" und „S6" über.

[0057] Bei dem Schritt „S5" überträgt die GDI die erzeugten Zwischenzeichnungsbefehle in der Form einer verbesserten Metadatei, wo dieselbe in einem Speicher gespeichert wird.

[0058] Bei dem Schritt „S6" weist der Prozeß den Spooler an, einen zweiten Durchlauf an dem Druckauftrag einzuleiten, um den speichergespeicherten Druckauftrag zu einem Ausgabegerät zu senden, wie es durch die Anwendung bestimmt ist. Nach einem Durchführen des Schritts „S6" geht der Prozeß zu einem Schritt „S7" über.

[0059] Bei dem Schritt „S7" leitet der Prozeß eine Implementierung des Druckprozessors ein. Der Druckprozessor wird auf eine von drei Weisen implementiert: der Druckprozessor wird über eine Operation des in **Fig.** 8 gezeigten Flußdiagramms initialisiert; der Druckprozessor wird geöffnet, Druckertreiberdetails werden wiedererlangt und neue Druckertreibereinstellungen werden in der Systemdatenbank gesichert, um eine Druckprozessordatenstruktur zu erreichen; und der Druckprozessor führt das Drucken (oder die Ausgabe) eines Druckauftrags an dem neuen Ausgabegerät über einen zugeordneten Gerätetreiber aus. Nach einem Durchführen des Schritts „S7" (eine der Routinen von **Fig.** 8, 9 und 10 zusammen, oder 11) geht der Prozeß zu einem Schritt „S8" über.

[0060] Bei dem Schritt „S8" bereitet die GDI den Druckauftrag unter Verwendung eines neuen Druckertreibers auf. Nach einem Durchführen des Schritts „S8" geht der Prozeß zu einem Schritt „S10" über. Ein eingekreistes Zeichen 1 stellt eine Verbindung zu einem Schritt „S7.36" zu einem Leiten eines Puffers zu der GDI gemäß den Operationen von **Fig.** 11 her. Nach einem Durchführen des Schritts „S8" geht der Prozeß zu einem Schritt „S10" über.

[0061] Bei einem Schritt „S9" liefert der durch die Anwendung ausgewählte neue Druckertreiber Ausgabegerätinformationen zu der GDI, was ermöglicht, daß die GDI den Druckauftrag aufbereitet, um eine Ausgabe des Druckauftrags über den Schritt „S10" zu ermöglichen.

[0062] Bei dem Schritt „S10" wird der Druckauftrag als ein Dokument gedruckt. An diesem Punkt endet der Prozeß.

[0063] Gemäß **Fig.** 8 implementiert der Druckprozessor eine erste Operation. Gemäß einem Schritt „S7.10" initialisiert der Prozeß die Druckprozessorfunktionszeiger und weist einen Speicher zu. Nach einem Durchführen des Schritts „S7.10" geht der Prozeß zu einem Schritt „S7.11" über.

[0064] Bei dem Schritt „S7.11" implementiert der Druckprozessor eine erste Operation. Gemäß dem Schritt „S7.10" initialisiert der Prozeß die Druckprozessorfunktionszeiger und weist einen Speicher zu. Nach einem Durchführen des Schritts „S7.10" geht der Prozeß zu einem Schritt „S7.11" über.

[0065] Bei dem Schritt „S7.11" leitet der Prozeß eine Rückkehr zu dem Spooler ein. Nach einem Durchführen des Schritts „S7.11" geht der Prozeß zu einem Schritt „S7.12" über.

[0066] Bei dem Schritt „S7.12" endet der Prozeß. Der Prozeß geht dann zu der Implementierung des Flußdiagramms von **Fig.** 9 und 10 über.

[0067] Gemäß dem Prozeß von **Fig.** 9 und 10 implementiert der Prozeß eine zweite Operation durch den Druckprozessor. Gemäß einem Schritt „S7.20" öffnet der Prozeß den Druckprozessor. Nach einem Durchführen des Schritts „S7.20" geht der Prozeß zu einem Schritt „S7.21" über.

[0068] Bei dem Schritt „S7.21" öffnet der Druckprozessor den Drukker. Gemäß dem Schritt „S7.10" leitet der

Prozeß das Öffnen über „openprinter API" ein. „Openprinter API" ist der Name der Teilroutine, die der Druckprozessor in der „Microsoft Windows 95 Device Driver Developers Kit" anruft. Die „openprinter API" existiert in dem Spooler, um Funktionen durchzuführen, die durch den Spooler benötigt werden. Nach einem Durchführen des Schritts „S7.21" geht der Prozeß zu einem Schritt „S7.22" über.

[0069] Bei dem Schritt „S7.22" erlangt der Druckprozessor Druckerdetails unter Verwendung von „printerinfo2 structure" wieder. Gemäß dem Schritt „S7.22" werden die Details über „getprinter API" wiedererlangt. „Getprinter API" ist der Name einer Teilroutine in dem Spooler, die Informationen von dem Spooler wiedererlangt. Nach einem Durchführen des Schritts „S7.22" geht der Prozeß zu einem Schritt „S7.23" über.

[0070] Bei dem Schritt „S7.23" ändert der Druckprozessor den Drukkertreibernamen in einer „printinfo2"-Struktur in einen neuen Gerätetreiber. Nach einem Durchführen des Schritts „S7.23" geht der Prozeß zu einem Schritt „S7.24" über.

[0071] Bei dem Schritt „S7.24" sichert der Druckprozessor die neuen Druckerinformationen in der Systemregisterdatenbank. Derartiges wird über „setprinter API" implementiert. „Setprinter API" ist eine Teilroutine, die in dem Spooler existiert und Informationen über den Drucker oder das Ausgabegerät sichert, so daß der Spooler eine Kenntnis der Informationen aufweist. Nach einem Durchführen des Schritts „S7.24" geht der Prozeß zu einem Schritt „S7.25" über.

[0072] Bei dem Schritt „S7.25" erlangt der Druckprozessor die Druckerdokumenteigenschaften wieder. Die Wiedererlangung ist als „documentproperties API" implementiert. „Documentproperties API" erhält Informationen von dem Spooler über Dokumente, die erzeugt werden. Nach einem Durchführen des Schritts „S7.25" geht der Prozeß zu einem Schritt „S7.26" über.

[0073] Bei dem Schritt „S7.26" ändert der Druckprozessor die Dokumenteigenschaften, um mit neuen Treibereinstellungen übereinzustimmen. Der Druckprozessor aktualisiert im wesentlichen die devmode-Struktur. „Devmode structure" ist eine Struktur, die in dem Spooler existiert und die Fähigkeiten der innerhalb der Computernetzwerkumgebung verfügbaren Druckertreiber beschreibt. Nach einem Durchführen des Schritts Schritts „S7.26" geht der Prozeß zu einem Schritt „S7.27" über.

[0074] Bei dem Schritt „S7.27" sichert der Druckprozessor die neuen Druckerdokumenteigenschaften in der Systemregisterdatenbank. Um dies vorzunehmen, ändert der Druckprozessor den Treibernamen unter Verwendung von „setprinter API" in den neuen erwünschten Namen. „Setprinter API" ist eine Teilroutine, die in dem Spooler existiert und Informationen über den Drucker oder das Ausgabegerät sichert, so daß der Spooler eine Kenntnis der Informationen aufweist. Nach einem Durchführen des Schritts „S7.27" geht der Prozeß zu einem Schritt „S7.28" über.

[0075] Bei dem Schritt „S7.28" weist der Druckprozessor Prozessordatenstrukturen zu und initialisiert dieselben. Nach einem Durchführen des Schritts „S7.28" geht der Prozeß zu einem Schritt „S7.29" über.

[0076] Bei dem Schritt „S7.29" leitet der Druckprozessor eine Rückkehr zu dem Spooler ein. Nach einem Durchführen des Schritts „S7.29" geht der Prozeß zu einem Schritt „S7.291" über.

[0077] Bei dem Schritt „S7.291" beendet der Druckprozessor die Teilroutine von **Fig.** 9 und 10 und bewegt sich vorzugsweise zu einer Implementierung der Teilroutine von **Fig.** 11 weiter.

[0078] Gemäß dem Prozeß von **Fig.** 11 implementiert der Prozeß eine dritte Operation durch den Druckprozessor. Gemäß einem Schritt „S7.30" druckt der Prozeß ein Dokument von der verbesserten Metadatei. Genauer gesagt richtet der Spooler ein Druckdokument auf den Druckprozessor. Nach einem Durchführen des Schritts „S7.30" geht der Prozeß zu einem Schritt „S7.31" über.

[0079] Bei dem Schritt „S7.31" validiert der Druckprozessor eingehende Parameter. Zum Beispiel validiert der Druckprozessor die Druckerkennung und den Datentyp. Nach einem Durchführen des Schritts „S7.31" geht der Prozeß zu einem Schritt „S7.32" über.

[0080] Bei dem Schritt „S7.32" öffnet der Druckprozessor den Drukker. Nach einem Durchführen des Schritts „S7.32" geht der Prozeß zu einem Schritt „S7.33" über.

[0081] Bei dem Schritt „S7.33" setzt der Druckprozessor den Spooler, um zu starten, und liest das Dokument.

Genauer gesagt wird die „startdocprinter API"-Datei implementiert. „Startdocprinter API" ist eine Funktion, die in dem Druckprozessor existiert und wiederum durch den Spooler angerufen wird, um dem Druckprozessor den Start eines Dokuments anzugeben. Nach einem Durchführen des Schritts „S7.33" geht der Prozeß zu einem Schritt „S7.34" über.

[0082] Bei dem Schritt „S7.34" liest der Druckprozessor einen Puffer einer verbesserten Metadatei. Nach einem Durchführen des Schritts „S7.34" geht der Prozeß zu einem Schritt „S7.36" über.

[0083] Bei einem Schritt „S7.35" überträgt der Druckprozessor die verbesserte Metadatei zu einem Puffer, wo dieselbe gelesen wird. Nach einem Durchführen des Schritts „S7.35" geht der Prozeß zu dem Schritt „S7.36" über.

[0084] Bei dem Schritt „S7.36" leitet der Druckprozessor den Puffer zu der GDI, um durch den neuen Treiber aufbereitet zu werden. Die „GDIplayspoolstream API"-Datei wird implementiert. „GDIplayspoolstream API" ist eine Teilroutine, die in der GDI existiert. Nach einem Durchführen des Schritts „S7.36" geht der Prozeß zu einem Schritt „S7.37" über.

[0085] Bei dem Schritt „S7.37" fährt der Druckprozessor fort, bis derselbe das Ende der Datei erreicht. Nach einem Durchführen des Schritts „S7.37" geht der Prozeß zu einem Schritt „S7.38" über.

[0086] Bei dem Schritt „S7.38" schließt der Prozessor den Drucker. Nach einem Durchführen des Schritts „S7.38" geht der Prozessor zu einem Schritt „S7.39" über.

[0087] Bei dem Schritt „S7.39" beendet der Druckprozeß die Teilroutine von **Fig.** 11 und kehrt zu einer Implementierung der Teilroutine von **Fig.** 7 zurück.

[0088] In Übereinstimmung mit den Vorschriften wurde die Erfindung hinsichtlich struktureller und methodischer Merkmale in einer mehr oder weniger spezifischen Sprache beschrieben. Es ist jedoch klar, daß die Erfindung nicht auf die spezifischen gezeigten und beschriebenen Merkmale begrenzt ist, da die hierin offenbarten Einrichtungen bevorzugte Formen eines Ausführens der Erfindung aufweisen. Die Erfindung wird daher in einer jeglichen der Formen oder Modifizierungen derselben innerhalb des ordnungsgemäßen Schutzbereichs der beigefügten Ansprüche beansprucht, die gemäß der Doktrin von Äquivalenten geeignet interpretiert werden.

TABELLE 1

```
/*Der folgende Code basiert auf einem Musterdruckprozessor,
der durch Microsoft Corporation in der Windows 95 Geräte-
treiberentwicklerausrüstung derselben geliefert wird. Es
wird angenommen, daß der Mustercode zu einer Modifizierung
und Anpassung durch Gerätehersteller für Geräte, die auf
dem Microsoft Windows Betriebssystem verwendet werden sol-
len, frei verfügbar ist, wenn notwendig. Modifizierungen,
die vorgenommen sind, um den Gedanken eines
DRUCKERTREIBERSCHALTENS zu testen, sind unten mit dem fol-
genden Kommentar vor und nach der Modifizierung angemerkt.
Die Microsoft-Urheberrechtaussage unten gilt nicht für ei-
nen Code innerhalb der Kommentare.*/

/***>> Modifizierung, um Druckertreiberschalten zu testen.
Beginn/Ende <<***/
/***>> L. Snyders 29.08.96 Copyright 1996 Hewlett-Packard
Company <<***/


/**********************************************************
*****************
*                          *
DIESER CODE UND DIE INFORMATIONEN SIND BEREITGESTELLT "WIE
SIE SIND", OHNE GARANTIE EINER JEGLICHEN ART ENTWEDER
AUSDRÜCKLICH ODER IMPLIZIERT, EINSCHLIESSLICH ABER NICHT
BEGRENZT AUF DIE IMPLIZIERTEN GARANTIEN EINER
MARKTGÄNGIGKEIT UND/ODER EIGNUNG FÜR EINEN SPEZIELLEN
ZWECK.*
*                          *
*Copyright (C) 1993-95 Microsoft Corporation. Alle Rechte
vorbehalten. *
*                          *
**********************************************************
***************/
```

```
#define TIMING

#include <windows.h>
#include <wingdi.h>
#include <winspool.h>
#include <winsplp.h>
#include <winuser.h>
#include <winbase.h>

#include "local.h"
#include "winprint.h"

TCHAR FAR *szWinPrint = TEXT("WinPrint");
LPTSTR Datatypes[]={"RAW", "EMF", 0};
LPSTR pSimple = NULL
LPTSTR pFull = NULL;

/***>> Modifizierung, um Druckertreiberschalten zu testen.
Beginn <<***/
/***>> L. Snyders 29.08.96 Copyright 1996 Hewlett-Packard
Company <<***/


LPTSTR pNewPrinterName = NULL;
LPDEVMODE pDevMode = NULL;

/***>> Modifizierung, um Druckertreiberschalten zu testen.
Ende <<***/
/***>> L. Snyders 29.08.96 <<***/


HDC WINAPI gdiPlaySpoolStream(LPSTR, LPSTR, LPSTR, DWORD,
LPDWORD, HDC);

#define PRINTPROCESSOR_TYPE_RAW   0
#define PRINTPROCESSOR_TYPE_EMF   1
#define PRINTPROCESSOR_TYPE_NUM   2
```

```
#ifdef TIMING
HWND hWndBench = 0;
#endif

//--------------------------------------------
//
//--------------------------------------------
BOOL
WINAPI
InitializePrintProcessor (
     LPPRINTPROCESSOR  pPrintProcessor,
     DWORD   cbPrintProcessor
)
{
     char szBuf[MAX_PATH];

     pPrintProcessor->fpEnumDatatypes =
WinprintEnumPrintProcessorDatatypes;
     pPrintProcessor->fpOpenPrintProcessor =
WinprintOpenPrintProcessor;
     pPrintProcessor->fpPrintDocument =
WinprintPrintDocumentOnPrintProcessor;
     pPrintProcessor->fpClosePrintProcessor =
WinprintClosePrintProcessor;
     pPrintProcessor->fpControlPrintProcessor =
WinprintControlPrintProcessor;

     if (LoadString(hlnst, IDS_BANNERSIMPLE, szBuf,
sizeof(szBuf)))
     pSimple = AllocSplStr(szBuf);

     if(LoadString(hlnst, IDS_BANNERFULL, szBuf,
     sizeof(szBuf)))
     pFull = AllocSplStr(szBuf);

     return TRUE;
}
```

```
// --------------------------------------
//
//--------------------------------------
BOOL
WINAPI
WinprintEnumPrintProcessorDatatypes(
     LPTSTR pName,
     LPTSTR pPrintProcessorName,
     DWORD Level,
     LPSTR pDatatypes,
     DWORD cbBuf,
     LPDWORD pcbNeeded,
     LPDWORD pcReturned
)
}
     DATATYPES_INFO_1  FAR *pInfo1 = (DATATYPES_INFO_1 FAR
*)pDatatypes;
     LPTSTR  FAR *pMyDatatypes = Datatypes;
     DWORD  cbTotal=0;
     LPBYTE pEnd;

     *pcReturned = 0;

     pEnd = (LPBYTE)pInfo1 + cbBuf;

     while (*pMyDatatypes) {

     cbTotal += wcslen(*pMyDatatypes)*sizeof(TCHAR) +
          sizeof(TCHAR) + sizeof(DATATYPES_INFO_1);

     pMyDatatypes++;
     }

     *pcbNeeded = cbTotal;

     if (cbTotal <= cbBuf) {

     pMyDatatypes = Datatypes;
```

```
        while (*pMyDatatypes) {

        pEnd-=wcslen(*pMyDatatypes)*sizeof(TCHAR) +
        sizeof(TCHAR);
        wcscpy((LPTSTR)pEnd, *pMyDatatypes);
        pInfo1->pName = (LPSTR)pEnd;
        pInfo1 + +;
        (*pcReturned)+ +;

        pMyDatatypes+ +;
    }

    } else {

    SetLastError(ERROR_INSUFFICIENT_BUFFER);
    return FALSE;
    }

    return TRUE;
}

//----------------------------------------
//
//----------------------------------------
HANDLE
WINAPI
WinprintOpenPrintProcessor(
      LPTSTR  pPrinterName
)
{
      PRINTPROCESSORDATA pData;
      HANDLE hPrinter=NULL;
      HDC    hDC=0;

/***>> Modifizierung, um Druckertreiberschalten zu testen.
Beginn <<***/
```

```
/***>> L. Snyders 29.08.96 Copyright 1996 Hewlett-Packard
Company <<***/


     DWORD dwBytesNeeded = 0; /* for Get/SetPrinters */
     DWORD dwDMBytesNeeded = 0;
     DWORD dwModeFlag = DM_OUT_BUFFER;

     BOOL  bRetcode = TRUE;

     PRINTER_INFO_2 *pPrtInfo2 = NULL;
     char  bur[256]; /*buffer for debug strings */

     OutputDebugString("HPPRINTP: Request to Open
     printer:");
     OutputDebugString(pPrinterName);

/***>> Modifizierung, um Druckertreiberschalten zu testen.
Ende <<***/
/***>> L. Snyders 29.08.96 <<***/

     if(!OpenPrinter(pPrinterName, &hPrinter, NULL))
     {
          OutputDebugString(„HPPRINTP.DLL Open Printer
          failed");
          return FALSE;
     }

/***>> Modifizierung, um Druckertreiberschalten zu testen.
Beginn <<***/
/***>> L. Snyders 29.08.96 Copyright 1996 Hewlett-Packard
Company <<***/

//
//GetPrinter zuerst aufrufen, um Bytezählwert zu erlangen,
der für Puffer benötigt wird
//
```

```
     GetPrinter (hPrinter, 2, 0, 0, &dwBytesNeeded);
//
//Speicher für Drucker-Info-Puffer zuweisen
//

if(/!(pPrtInfo2 = (PRINTER_INFO_2 *) AllocSplMem (dwBytes-
Needed)))
{
     OutputDebugString("HPPRINTP.DLL AllocSplMem
     failed\n\r");
     return FALSE;
}
//
//GetPrinters Stufe 2
//
     bRetcode = GetPrinter (hPrinter, 2, (LPBYTE)pPrtInfo2,
     dwBytesNeeded, &dwBytesNeeded);
     if (bRetcode = = FALSE)
     {
          OutputDebugString(HPPRINTP.DLL Second GetPrinter
          failed\n\r");
          return FALSE;
     }
//
// Den Treiber in den Mono-Bitabbildung-Drucker ändern
//

     pPrtInfo2->pDriverName = AllocSplStr(„Mono Bitmap
Driver"); /* New Driver name */
     bRetcode = SetPrinter(hPrinter, 2, (LPBYTE)pPrtInfo2,
0;
     if (bRetcode = = FALSE)
     {
          OutputDebugString("HPPRINTP.DLL SetPrinter
failed\n\r);
          return FALSE;
     }
```

```
//
//Die devmode-Struktur des neuen Druckers einstellen, zu-
erst die Größe derselben erlangen
//
     dwDMBytesNeeded = DocumentProperties(NULL, hPrinter,
pPrinterName, NULL, NULL, 0);
     pDevMode = (LPDEVMODE)AllocSplMem(dwDMBytesNeeded);

//
// Dann die vorgegebenen devmode-Werte erlangen
//

     bRetcode = DocumentProperties(NULL, hPrinter, pPrin-
terName, pDevMode, NULL, dwModeFlag);
     if(bRetcode < 0)
     {
          OutputDebugString(„HPPRINTP.DLL DocProp
Failed\n\r");
          return(FALSE);
     }

//
// Nun vorgegebene Devmode-Werte in Registerdatenbank ein-
stellen
//

     pPrtInfo2->pDevMode = pDevMode;
     bRetcode = SetPrinter(hPrinter, 2, (LPBYTE)pPrtInfo2,
     0);

/***>> Modifizierung, um Druckertreiberschalten zu testen.
Ende <<***/
/***>> L. Snyders 29.08.96 <<***/


     pData =
(PPRINTPROCESSORDATA)AllocSplMem(sizeof(PRINTPROCESSORDATA)
);
```

```
    pData->cb      = sizeof(PRINTPROCESSORDATA);
    pData->signature =PRINTPROCESSORDATA_SIGNATURE;
    pData->hPrinter = hPrinter;
    pData->semPaused = CreateEvent(NULL, FALSE,
    TRUE,NULL);
    pData->pPrinterName = AllocSplStr(pPrinterName);

    return (HANDLE)pData;
}

//-------------------------------------------------
//
//-------------------------------------------------
UINT ValidateDatatype(LPTSTR pDatatype)
{
    LPTSTR  FAR *pMyDatatypes=Datatypes;
    DWORD  uDatatype=0;

    while (*pMyDatatypes && wcscmp(*pMyDatatypes,
    pDatatype))
    {
    pMyDatatypes+ +;
    uDatatype+ +;
    }
    return uDatatype;
}

//-------------------------------------------------
//
//-------------------------------------------------
DWORD
ValidateBannerType(LPTSTR IpBanner)
{
    if(IpBanner && *IpBanner)
    {
    if(!wcscmp(IpBanner, pFull))
        return BANNER_FULL;
```

```
    if (!wcscmp(IpBanner, pSimple))
        return BANNER_SIMPLE;

#ifdef TIMING
    if(!wcscmp(IpBanner, "Bench"))
    {
        hWndBench = FindWindow("BNCH32PRT", "PRTWIN");

        if/hWndBench)
        SendMessage(hWndBench, WM_USER + 801, 0, 0);

        return 0;
    }
    else
        hWndBench = 0
#endif

    return BANNER_CUSTOM;
    }

    return 0;
}

//--------------------------------------------
//
//--------------------------------------------
BOOL
WINAPI
WinprintPrintDocumentOnPrintProcessor(
    HANDLE hPrintProcessor,
    LPPRINTPROCESSORDOCUMENTDATA  IpDoc
)
{
    PRINTPROCESSORDATA pData;
    DOC_INFO_2 DocInfo;
    DWORD  LastError = 0;
    DWORD  NoRead, NoWritten;
    DWORD  iBannerType;
```

```
BYTE  Buffer[4096];
HANDLE  hPrinterRead;
HDC hDC = NULL;
LPBYTE pReadBuf;
DWORD cbReadBuf;
DWORD NoLeftOver = 0;
BOOL ret = TRUE;

if(!(pData = ValidateHandle(hPrintProcessor)))
{
SetLastError(ERROR_INVALID_HANDLE);
return FALSE;
}

pData->uDatatype = ValidateDatatype(IpDoc->pDatatype);

iBannerType = ValidateBannerType(IpDoc->pSepFile);

if (iBannerType)
InsertBannerPage(pData->hPrinter, IpDoc->JobId, IpDoc-
>pOutputFile, iBannerType, IpDoc->pSepFile);

DocInfo.pDocName = IpDoc->pDocumentName;
DocInfo.pOutputFile = IpDoc->pOutputFile;  //the spool
file

DocInfo.pDatatype = IpDoc->pDatatype;
DocInfo.JobId = IpDoc->JobId;

// Den gleichen Drucker zu einem Lesen der Spooldatei
öffnen.
if(!OpenPrinter(pData->pPrinterName, &hPrinterRead,
NULL))
return FALSE;

// Dies läßt ReadPrinter() die Spooldatei für uns le-
sen.
DocInfo.dwMode = DI_READ_SPOOL_JOB;
```

```
      if(!StartDocPrinter(hPrinterRead, 2,
      (LPBYTE)&DocInfo))
      {
      LastError = GetLastError();
      ret = FALSE;
      goto Exit_2;
      }

      if(pData->uDatatype = = PRINTPROCESSOR_TYPE_RAW)
      {
      //Direktes Schreiben zu Tor starten
      DocInfo.dwMode = DI_CHANNEL_WRITE;
      if(!StartDocPrinter(pData->hPrinter, 2,
      (LPBYTE)&DocInfo))
      {
           //SetJob(pData->hPrinter, IpDoc->JobId, 0, NULL,
JOB_CONTROL_CANCEL);
           LastError = GetLastError();
           ret = FALSE
           goto Exit_1;
     }
     }

     pReadBuf = (LPBYTE)Buffer;
     cbReadBuf = sizeof(Buffer);

     //Hier wird ReadPrinter() verwendet, um tatsächlich 4K
     //der Spooldatei zu lesen. Diese Daten werden dann
     //entweder direkt zu dem Drucker gesendet, falls RAW,
     //oder zu der GDI, falls EMF. Bei dem letzteren Fall
     //werden die Metadateidaten auf dem Drucker-DC
     //wiedergegeben und dann zu dem Drucker gesendet (all
     // dies wird vorgenommen durch gdiPlaySpoolStream()).
     while (((ReadPrinter(hPrinterRead, pReadBuf, cbRead-
     Buf, &NoRead)) && NoRead) || NoLeftOver)
     {
     //gdiPlaySpoolStream spielt nun eine Seite zu einer
     //Zeit ab. Somit gibt ein Abspielen von EMF
```

```
//Gelegenheit, bei jeder Seite anzuhalten.

if(pData->fsStatus & PRINTPROCESSOR_PAUSED)
     WaitForSingleObject(pData->semPaused, INFINITE);

if(pData->fsStatus & PRINTPROCESSOR_ABORTED)
{
     //wir können nicht nur für EMF anhalten
     //wir müssen den DC bereinigen und etc.

     if(pData->uDatatype = = PRINTPROCESSOR_TYPE_EMF)
     gdiPlaySpoolStream(NULL, NULL, IpDoc-
     >pSpoolFileName, 0, 0, hDC);

     break;
}

//Überprüfen, ob RAW oder EMF, und entsprechend zu dem
//richtigen Ort senden.
if(pData->uDatatype = = PRINTPROCESSOR_TYPE_RAW)
     WritePrinter(pData->hPrinter, Buffer, NoRead,
     &NoWritten);
else
{
     NoRead + = NoLeftOver;
     NoWritten = NoRead;

     SetLastError(ERROR_SUCCESS);

     //hDC ist NULL, wenn man das erste Mal hierher
     kommt
     //dies ist wo die Metadatei abgespielt wird.
     hDC = gdiPlaySpoolStream(
     pData->pPrinterName, IPDoc->pOutputFile,
     Buffer, IpDoc->JobId, &NoRead, hDC);

     //Auf eine Rückkehr hin ist NoRead die Anzahl
     //von Bytes, die in dem vorhergehenden Puffer
```

```
            //verarbeitet werden. Und dieselbe darf nicht
            //größer als NoWritten sein.

            if(hDC && (NoWritten > = NoRead))
            {
            NoLeftOver = NoWritten - NoRead;

            //es kann einen unvollständigen sp-Block an dem
            //Ende geben, der nicht verarbeitet wurde und den
            //wir übertragen müssen

            if(NoLeftOver)
                 CopyMemory(Buffer, Buffer + NoRead, NoLeft-
                 Over);

            pReadBuf = Buffer + NoLeftOver;
            cbReadBuf = sizeof(Buffer) - NoLeftOver;
            }
            else
            {
            //wir haben versagt

            //das ~EMF????.TMP ??? löschen
            //oder wollen wir die EMF dort belassen und einem
            //Benutzer ermöglichen, zu wiederholen ???

            LastError = GetLastError();

            DBGMSG(DBG_ERROR, ("WinprintPrintDoc: gdiPlay-
SpoolStream failed %d\n", LastError));

            //Benutzer zu wiederholen/abbrechen/ok auffor-
dern?

            gdiPlaySpoolStream(NULL, NULL, IpDoc-
>pSpoolFileName, 0, 0, 0);
            ret = FALSE;
```

```
            break;
            }
        }
        }

        if(pData->uDatatype = = PRINTPROCESSOR_TYPE_RAW)
        EndDocPrinter(pData->hPrinter);
Exit_1:
        EndDocPrinter(hPrinterRead);
Exit_2:
        ClosePrinter(hPrinterRead);

        if(LastError)
        SetLastError(LastError);

#ifdef TIMING
        if(hWndBench)
        SendMessage(hWndBench, WM_USER + 802, 0, 0);
#endif

        return ret;
}

//--------------------------------------------
//
//--------------------------------------------
BOOL
WINAPI
WinprintClosePrintProcessor(
        HANDLE hPrintProcessor
)
{
        PRINTPROCESSORDATA pData;

        pData = ValidateHandle(hPrintProcessor);

        if (!pData)
        {
```

```
        SetLastError(ERROR_INVALID_HANDLE);
        return FALSE;
        }

        pData->signature = 0;

        /*Jegliche zugewiesene Betriebsmittel freigeben*/

        if(pData->hPrinter)
        ClosePrinter(pData->hPrinter);

        CloseHandle(pData->semPaused);

        if(pData->pPrinterName)
        FreeSplStr(pData,->pPrinterName);

        FreeSplMem(pData, pData->cb);

        return TRUE;
}

//------------------------------------------------
//
//------------------------------------------------
BOOL
WINAPI
WinprintControlPrintProcessor(
      HANDLE hPrintProcessor,
      DWORD  Command,
      DWORD  JobID,
      LPTSTR pDatatype,
      LPTSTR pSpoolFile
)
{
      PRINTPROCESSORDATA pData;
      PRINTPROCESSORDATA Data;

    if(hPrintProcessor)
```

```
pData = ValidateHandle(hPrintProcessor);
else
{
if (Command ! = JOB_CONTROL_CANCEL)
     return FALSE;

Data.uDatatype = ValidateDatatype(pDatatype);
if (Data.uDatatype > = 0)
     pData = &Data;
else
     pData = 0;
}

if(pData)
{
switch(Command)
{
case JOB_CONTROL_PAUSE:

     ResetEvent(pData->semPaused);
     pData->fsStatus | = PRINTPROCESSOR_PAUSED;
     return TRUE;

case JOB_CONTROL_CANCEL:

     if(!hPrintProcessor)
     {
     //wir löschen einen Auftrag, der ein Drucken
     //nicht begonnen hat

     if(pData->uDatatype = = PRINTPROCESSOR_TYPE_EMF)
     {
          return (BOOL)gdiPlaySpoolStream(NULL, NULL,
          pSpoolFile, 0, 0, 0);
     }

     return TRUE;
     }
```

```
            pData->fsStatus | = PRINTPROCESSOR_ABORTED;

            /*Durchfallen, um Auftrag freizugeben, falls an-
            gehalten*/

        case JOB_CONTROL_RESUME:

            if (pData->fsStatus & PRINTPROCESSOR_PAUSED)
            {
            pData->fsStatus &= ~PRINTPROCESSOR_PAUSED;
            SetEvent(pData->semPaused);
            }
            return TRUE;

        default:

            break;
        }

        }

        return FALSE;
}

//------------------------------------------------
//
//------------------------------------------------
BOOL
WINAPI
WinprintInstallPrintProcessor(
     HWND  hWnd
)
{
     return TRUE;
}

//------------------------------------------------
```

```
//
//--------------------------------------------------
PRINTPROCESSORDATA
ValidateHandle(
     HANDLE  hQProc
)
{
     PRINTPROCESSORDATA pData = (PRINTPROCESSORDATA)hQProc;

     if (pData && pData->signature = =
     PRINTPROCESSORDATA_SIGNATURE)
     return( pData );

     return( NULL );
}

/* Dateiname: local.h   */

/*****************************************************************
*****************
*                                    *

* THIS CODE AND INFORMATION IS PROVIDED "AS IS" WITHOUT
WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED,
INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF
MERCHANTABILITY AND/OR FITNESS FOR A PARTICULAR PURPOSE. *
*                                              *
*Copyright © 1993-95 Microsoft Corporation. Alle Rechte
vorbehalten.     *
*                                               *
*****************************************************************
******************/

//-----------------------------------
//WINPRINT
#define IDS_BANNERTITLE1      521
#define IDS_BANNERTITLE2      522
#define IDS_BANNERJOB         523
```

```
#define IDS_BANNERDATE          524
#define IDS_BANNERSIMPLE        526
#define IDS_BANNERFULL          527
//-------------------------------------

//-------------------------------------
//WINPRINT
#define IDC_STANDBAN      600
#define RT_CLIPFILE       601
//-------------------------------------


//-------------------------------------
//EXTERN VARIABLES
//-------------------------------------

extern HANDLE  hlnst;


//-------------------------------------
//DEBUG STUFF
//-------------------------------------
#ifdef DEBUG

extern DWORD SplDbgLevel;

VOID cdecl DbgMsg( LPSTR MsgFormat, …);

/*Diese Flags werden nicht als Argumente zu dem DBGMSG-
Makro verwendet.
*Man muß das hohe Wort der globalen Variable setzen, um zu
bewirken, daß dieselbe anhält.
*Dasselbe wird ignoriert, falls mit DBGMSG verwendet.
*(Hier hauptsächlich zu erläuternden Zwecken.)
*/
#define DBG_BREAK_ON_WARNING  (DBG_WARNING << 16)
#define DBG_BREAK_ON_ERROR    (DBG_ERROR << 16)
```

```
#define DBG_NONE        0
#define DBG_INFO        1
#define DBG_TRACE         DBG_INFO
#define DBG_WARING         2
#define DBG_ERROR       4

/*Hierfür werden doppelte Klammern benötigt, z. B.:
*
*  DBGMSG( DBG_ERROR, („Error code %d", Error));
*
*Dies ist so, weil man keine variablen Parameterlisten bei
Makros verwenden kann.
*Die Aussage wird in einem Nicht-Bereinigungsmodus zu einem
Strichpunkt vorverarbeitet.
*
*Die globale Variable GLOBAL_DEBUG_FLAGS über den Bereini-
ger einstellen.
*Ein Setzen des Flag bei dem niedrigen Wort bewirkt, daß
dieser Pegel gedruckt wird;
*ein Setzen des hohen Worts bewirkt ein Anhalten bei dem
Bereiniger.
*Z. B. wird ein Setzen desselben auf 0x00040006 alle Warn-
und Fehlernachrichten ausdrucken und Fehler unterbrechen.
*/

#define DBGMSG( Level, MsgAndArgs ) {if (Level > =
SplDbgLevel) {DbgMsg MsgAndArgs;}}
#define DBGBREAK() {DebugBreak();}
#define ASSERT( Expr, MsgAndArgs ) {if (!Expr) {DbgMsg
MsgAndArgs; DebugBreak();}}
VOID SpllnSem(VOID);
VOID SplOutSem(VOID);

#else

#define DBGMSG( Level, MsgAndArgs )
#define DBGBREAK()
#define ASSERT( Expr, MsgAndArgs )
```

```
#define SpllnSem()
#define SplOutSem()

#endif

//--------------------------------------
//FUNCTION PROTOTYPE
//--------------------------------------

#define AllocSplMem(a)    LocalAlloc( LPTR, a )
#define FreeSplMem(a, b)  LocalFree( a )

LPVOID
ReallocSplMem(
     LPVOID IpOldMem,
     DWORD cbOld,
     DWORD cbNew
);

LPTSTR
AllocSplStr(
     LPTSTR IpStr
);

BOOL
FreeSplStr(
     LPTSTR IpStr
);

BOOL
ReallocSplStr(
     LPTSTR FAR *plpStr,
     LPTSTR IpStr
);

//--------------------------------------
//UNICODE TO ANSI MACRO
//???!!! wir sollten diese früher oder später loswerden
```

```
//-----------------------------------------
#include <string.h>
#include <stdlib.h>
#include <stdio.h>

#ifndef UNICODE

LPSTR
mystrstr(
     LPSTR cs,
     LPSTR ct
);

LPSTR
mystrrchr(
     LPSTR cs,
     char c
];

LPSTR
mystrchr(
     LPSTR cs,
     char c
);

int
mystrnicmp(
     LPSTR cs
     LPSTR ct
     int n
);

#define wcscat(a, b) lstrcat(a, b)
#define wcscmp(a, b) lstrcmp(a, b)
#define wcscpy(a, b) lstrcpy(a, b)
#define wcslen(a) lstrlen(a)

#undef wcsicmp
```

```
#define wcsicmp(a, b) lstrcmpi(a, b)

#define wcschr(a, b) mystrchr(a, b)
#define wcsrchr(a, b) mystrrchr(a, b)
// #define wcsncmp(a, b, c) strncmp(a, b, c)

#undef wcsnicmp
#define wcsnicmp(a, b, c) mystrnicmp(a, b, c)

#define wcsstr(a, b) mystrstr(a, b)

#endif // UNICODE // ccteng
```

Patentansprüche

1. Eine Informationsverteilungsvorrichtung, die innerhalb einer Computernetzwerkumgebung (**12**) verwendbar ist und die folgende Merkmale aufweist:
einen Computer (**10**), der ein Betriebssystem aufweist und konfiguriert ist, um innerhalb der Computernetzwerkumgebung (**12**) wirksam zu sein;
eine Anwendung (**36**), die konfiguriert ist, um über das Betriebssystem auf dem Computer (**10**) zu laufen, wobei die Anwendung (**36**) konfiguriert ist, um einen Quellauftrag (**28**) in der Form eines Zwischendateiformats zu erzeugen, das eine Ausgabebefehlsdatei (**42**) aufweist;
einen Druckprozessor (**50**) in der Form eines ausführbaren Zwischencodes zu einem Wirksamsein an der Ausgabebefehlsdatei (**42**); und
einen Drucker (**14**, **16**, **18**), der einen Druckertreiber (**56**) aufweist, der konfiguriert ist, um die Ausgabebefehlsdatei (**42**) in Ausgabebefehle umzuwandeln, die durch den Drucker (**14**, **16**, **18**) verwendbar sind, um eine Ausgabe zu erzeugen;
wobei der Druckprozessor (**50**) wirksam ist, um Druckerdetails für einen identifizierten Druckertreibernamen von einer Speicherposition in einem Speichergerät (**44**) des Computers (**10**) wiederzuerlangen, den identifizierten Druckertreibernamen in einen unterschiedlichen identifizierten Druckertreibernamen zu ändern, die Druckerdetails des unterschiedlichen identifizierten Druckertreibernamens in der Form von neuen Druckerinformationen in einer Systemregisterdatenbank (**54**) zu speichern, Druckerdokumenteigenschaften der gespeicherten Druckerdetails von dem Speichergerät wiederzuerlangen, die wiedererlangten Druckerdokumenteigenschaften zu verändern, um dieselben an neue Druckertreibereinstellungen anzupassen, die neuen Druckerdokumenteigenschaften in der Systemregisterdatenbank (**54**) zu speichern und Druckprozessordatenstrukturen zuzuteilen und zu initialisieren, die verwendbar sind, um einen Druckauftrag (**28**) auf dem Drucker (**14**, **16**, **18**) auszuführen.

2. Die Informationsverteilungsvorrichtung gemäß Anspruch 1, bei der die Ausgabebefehlsdatei (**42**) eine Zeichnungsbefehlsdatei aufweist, die Zeichnungsbefehle umfaßt.

3. Die Informationsverteilungsvorrichtung gemäß Anspruch 1, bei der der Druckertreiber (**56**) das Zwischendateiformat von Ausgabebefehlen empfängt, um die Befehle zu verarbeiten.

4. Die Informationsverteilungsvorrichtung gemäß Anspruch 1, bei der das Zwischendateiformat eine verbesserte Metadatei (**42**) aufweist.

5. Die Informationsverteilungsvorrichtung gemäß Anspruch 1, bei der der Druckprozessor (**50**) die Ausgabebefehlsdatei (**42**) dem Druckertreiber (**56**) des Druckers (**14**, **16**, **18**) über einen Seriell-Sendevorgang zuführt, was ein Aufbereiten der Ausgabebefehlsdatei (**42**) für mehrere Ausgabegeräte (**14**, **16**, **18**) ermöglicht.

6. Die Informationsverteilungsvorrichtung gemäß Anspruch 1, bei der der Druckprozessor (**50**) die Ausgabebefehlsdatei (**42**) dem Druckertreiber (**56**) des Druckers (**14**, **16**, **18**) über einen Mehrfach-Teilprozeß-Parallel-Sendevorgang zuführt, um die Ausgabebefehlsdatei (**42**) für mehrere Ausgabegeräte (**14**, **16**, **18**) aufzubereiten.

7. die Informationsverteilungsvorrichtung gemäß Anspruch 1, bei der der Druckprozessor (**50**) die Ausgabebefehlsdatei (**42**) dem Druckertreiber (**56**) des Druckers (**14**, **16**, **18**) über einen Mehrfach-Teilprozeß in einem Parallel-Sendevorgang zuführt, um die Ausgabebefehlsdatei (**42**) für mehrere Ausgabegeräte (**14**, **16**, **18**) aufzubereiten.

8. Die Informationsverteilungsvorrichtung gemäß Anspruch 1, die ferner eine Registerdatenbank (**54**) und eine Systemanwendungsprogrammierungsschnittstelle (API = application programming interface) (**52**) aufweist, wobei die Registerdatenbank (**54**) durch die Anwendungsprogrammierungsschnittstelle (API) (**52**) an dem Druckprozessor (**50**) wirksam ist, um den Drucker (**14**, **16**, **18**) zu konfigurieren, um die Ausgabebefehlsdatei (**42**) zu empfangen.

9. Die Informationsverteilungsvorrichtung gemäß Anspruch 1, die ferner ein Speichergerät (**44**) und einen Spooler (**46**) zu einem Nehmen und einem Speichern der anwendungserzeugten Ausgabebefehlsdatei (**42**) aufweist.

10. Die Informationsverteilungsvorrichtung gemäß Anspruch 9, die ferner eine Graphikgerätschnittstelle (GDI = graphics device interface) (**40**) zu einem Implementieren von graphischen Funktionen und einem dynamischen Verbinden einer graphischen Systemanwendungsprogrammschnittstelle (API) (**52**) mit dem Druckertreiber (**56**) des Druckers (**14**, **16**, **18**), das Speichergerät (**44**) und einen ursprünglichen Druckertreiber (**38**) aufweist, der verwendet wird, um die Ausgabebefehlsdatei (**42**) zu erzeugen.

11. Die Informationsverteilungsvorrichtung gemäß Anspruch 9, die ferner einen Spool-Kopfblock (**48**) aufweist, der durch den Spooler (**46**) erzeugt und an die Ausgabebefehlsdatei (**42**) übertragen wird, wobei der Spool-Kopfblock (**48**) eine Spool-Auftrag-Kopfblock-Datei zu einem Beschreiben der Bestimmungsorte und der aufbereiteten Daten und/oder codierten Daten für jeden Bestimmungsort für die anwendungserzeugte Ausgabebefehlsdatei (**42**) aufweist.

12. Die Informationsverteilungsvorrichtung gemäß Anspruch 11, bei der die Ausgabebefehlsdatei (**42**) eine verbesserte Metadatei (**42**) aufweist, der Druckprozessor (**50**) die ursprüngliche Kopie der verbesserten Metadatei (**60**) in das Speichergerät (**44**) kopiert, nachdem der Druckprozessor (**50**) ansprechend auf den Spooler (**46**) ein Druckdokument einleitet, und eine Rückkopplungsschleife (**62**) mehrere Druckdokumentaufgaben einleitet, die mehrere Durchgangsaufträge (**64**) zu einer Verteilung aufweisen.

13. Ein Verfahren zu einer Verwendung bei einem System zum Verteilen von Druckaufträgen (**28**) von einem Computer (**10**), der verwendbar ist, um in einer Computernetzwerkumgebung (**12**) wirksam zu sein, von dem Typ, der ein Betriebssystem; eine Anwendung (**36**), die zu einem Laufen auf dem Betriebssystem und einem Erzeugen eines Quellauftrags (**28**) in der Form eines Zwischendateiformats konfiguriert ist, das eine Ausgabebefehlsdatei (**42**) aufweist; und einen Drucker (**14**, **16**, **18**) aufweist, der einen Druckertreiber (**56**) zum Empfangen der Ausgabebefehlsdatei (**42**) zum Erzeugen einer Ausgabe aufweist, wobei das Verfahren folgende Schritte aufweist:
Bereitstellen eines Druckprozessors (**50**) in der Form eines ausführbaren Zwischencodes zu einem Wirksamsein an der Ausgabebefehlsdatei (**42**);
Wiedererlangen von Druckerdetails für einen Namen eines identifizierten Druckertreibers (**38**) von einer Speicherposition in einem Speichergerät (**44**) des Computers (**10**);
Ändern des Namens des identifizierten Druckertreibers (**38**) in einen unterschiedlichen identifizierten Drukkertreiber (**56**);
Speichern der Details des Druckers (**14**, **16**, **18**) des Namens des unterschiedlichen identifizierten Druckertreibers (**56**) in der Form von neuen Druckerinformationen in einer Systemregisterdatenbank (**54**);
Wiedererlangen von Druckerdokumenteigenschaften der Details des gespeicherten Druckers (**14**, **16**, **18**) von dem Speichergerät (**44**);
Verändern der wiedererlangten Druckerdokumenteigenschaften, um dieselben an neue Einstellungen des Drukkertreibers (**56**) anzupassen;
Speichern der neuen Druckerdokumenteigenschaften in der Systemregisterdatenbank (**54**); und
Zuteilen und Initialisieren von Datenstrukturen des Druckprozessors (**50**), die verwendbar sind, um einen Druckauftrag (**28**) auf dem Drucker (**14**, **16**, **18**) auszuführen.

14. Das Verfahren gemäß Anspruch 13, das nach dem Schritt des Initialisierens der Datenstrukturen des Druckprozessors (**50**) ferner den Schritt eines Zurückkehrens zu einem Spooler (**46**) für eine Neuinitialisierung des Systems aufweist, um einen anderen Druckauftrag (**28**) über die zuvor erwähnten Schritte neuauszuführen, um mehrere Durchgangsdruckaufträge (**28**) für eine Verteilung auszuführen.

15. Das Verfahren gemäß Anspruch 13, bei dem die Ausgabebefehlsdatei in der Form einer verbesserten Metadatei (**42**) ausgeführt ist, wobei das Verfahren ferner den Schritt eines Bereitstellens eines Spoolers (**46**) zu einem Nehmen und Speichern der verbesserten Metadatei (**42**) von einer Anwendung (**36**), eines Erzeugens eines Spool-Kopfblocks (**48**) durch den Spooler (**46**) und eines Zuweisens des Spool-Kopfblocks (**48**) an die verbesserte Metadatei (**42**) aufweist, wobei der Spool-Kopfblock (**48**) eine Spool-Auftrag-Kopfblock-Datei zum Beschreiben der Bestimmungsorte und der aufbereiteten Daten und/oder codierten Daten für jeden Bestimmungsort für die durch die Anwendung (**36**) erzeugte verbesserte Metadatei (**42**) aufweist.

Es folgen 9 Blatt Zeichnungen

Anhängende Zeichnungen

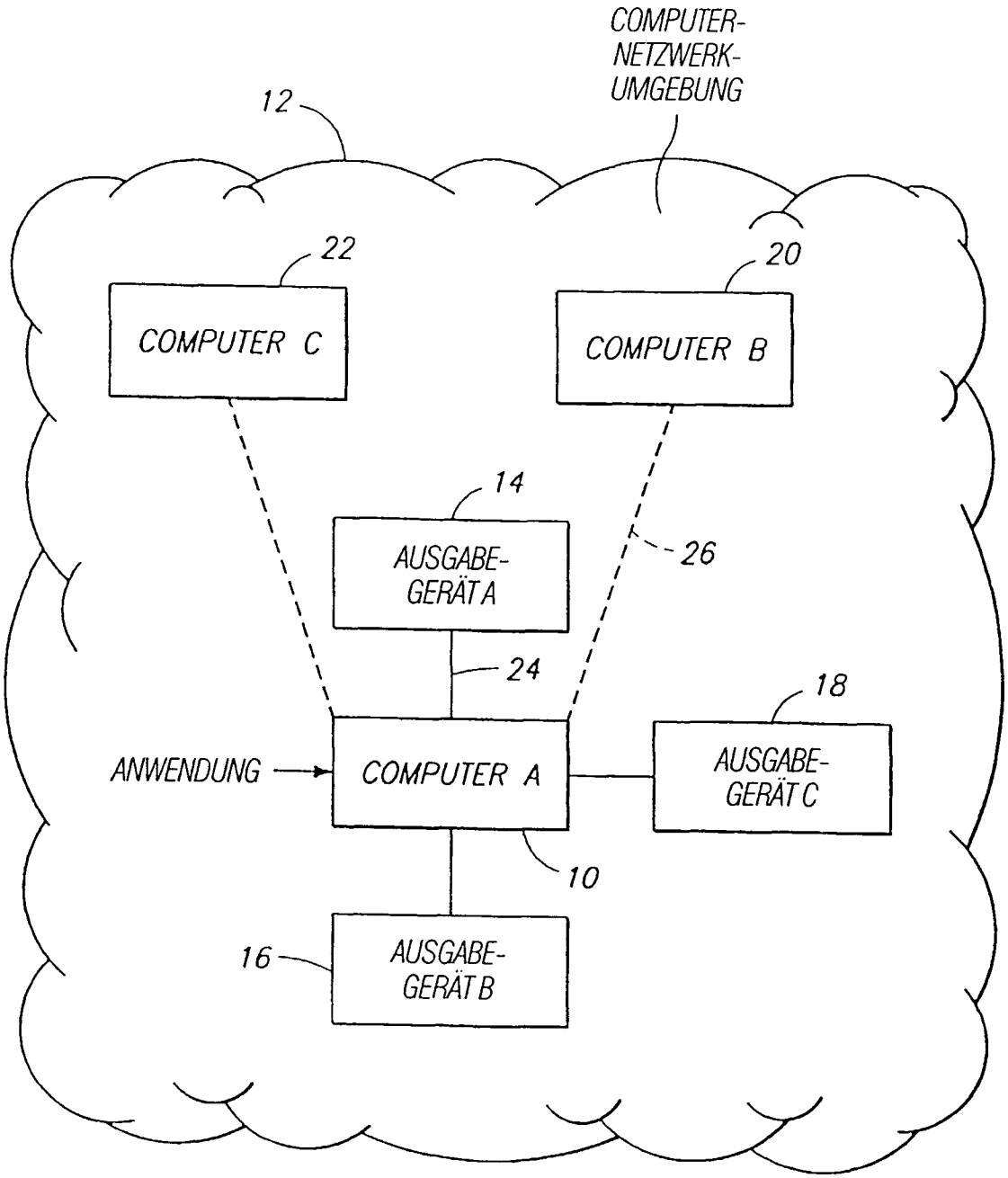

FIG. 1

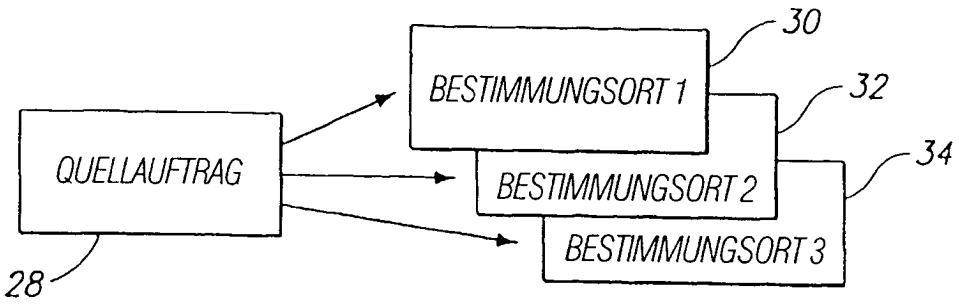
30
BESTIMMUNGSORT 1
32
34
QUELLAUFTRAG
BESTIMMUNGSORT 2
BESTIMMUNGSORT 3
28

Fig 2

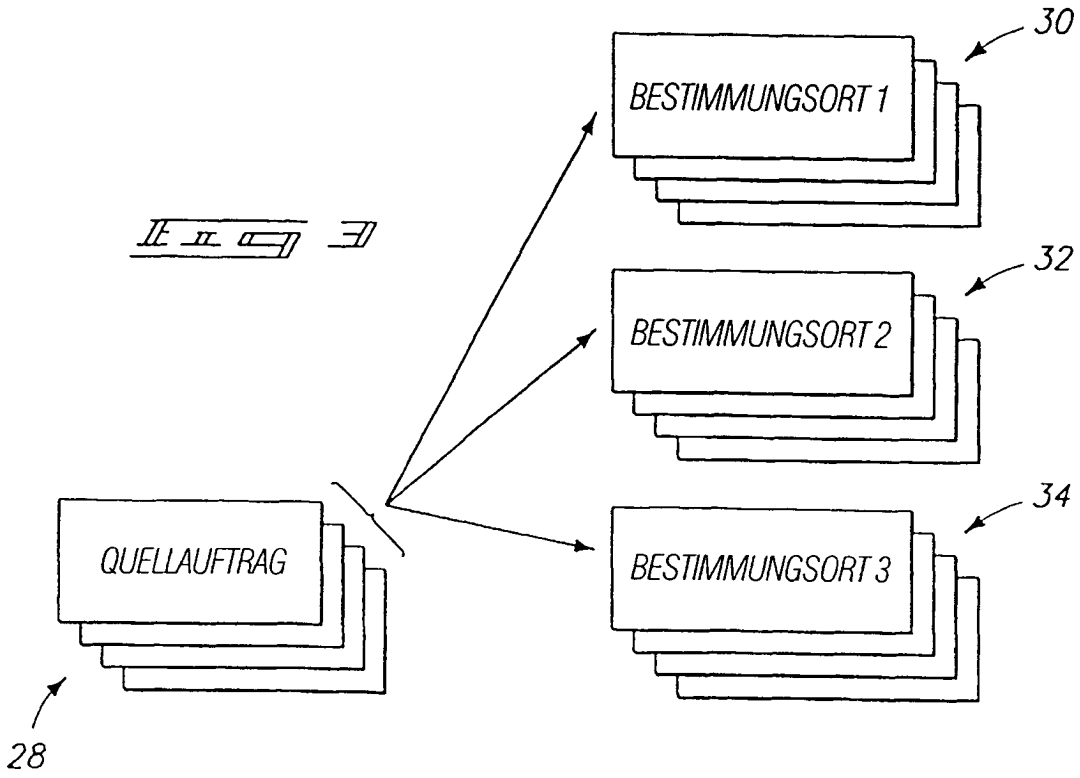
30
BESTIMMUNGSORT 1
32
BESTIMMUNGSORT 2
34
QUELLAUFTRAG
BESTIMMUNGSORT 3
28

Fig 3

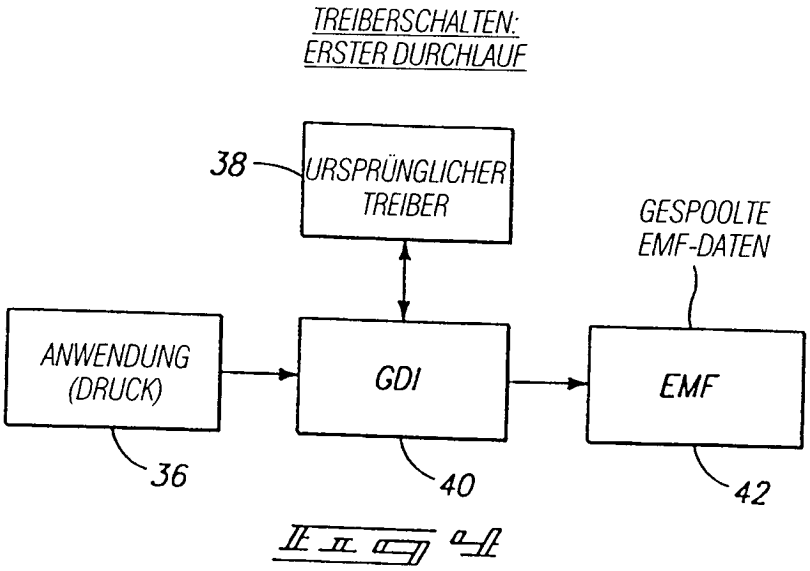
TREIBERSCHALTEN:
ERSTER DURCHLAUF
38
URSPRÜNGLICHER TREIBER
GESPOOLTE EMF-DATEN
ANWENDUNG (DRUCK)
GDI
EMF
36
40
42

Fig. 4

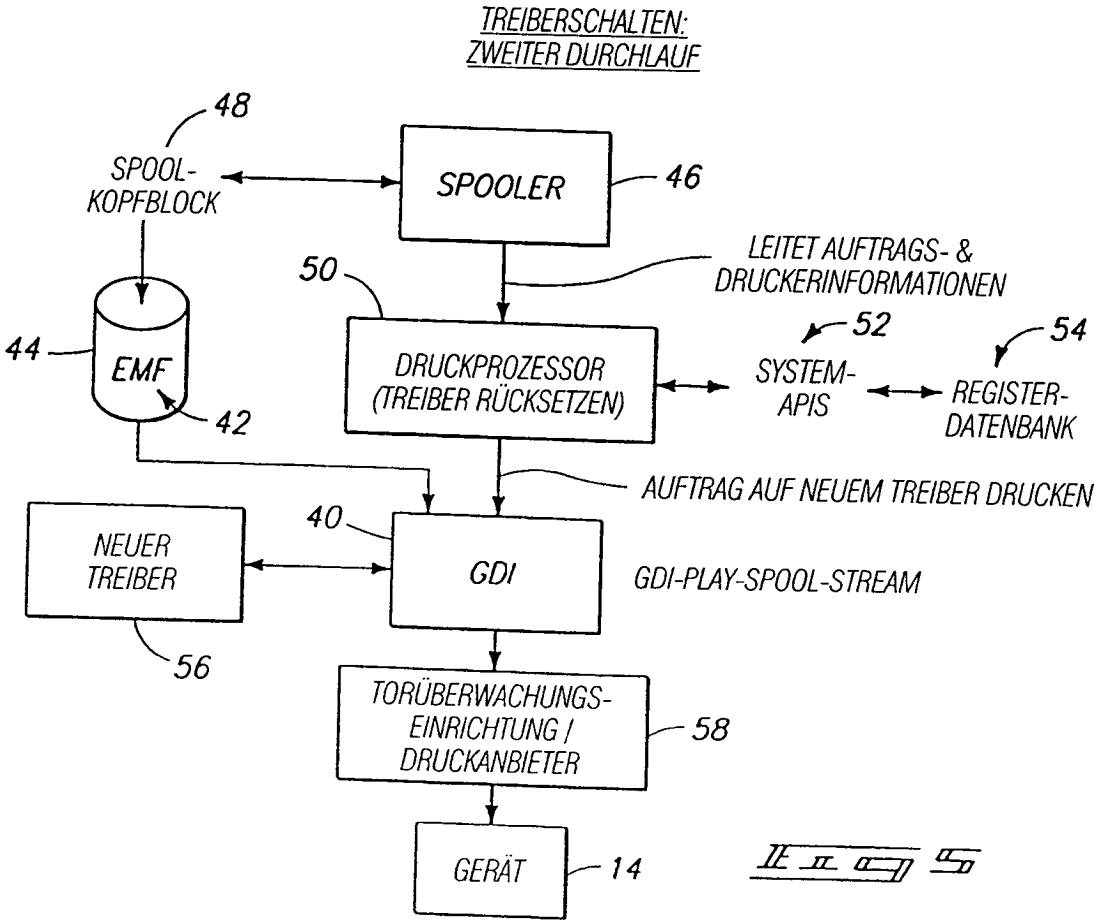
TREIBERSCHALTEN:
ZWEITER DURCHLAUF
48
SPOOL-KOPFBLOCK
SPOOLER
46
LEITET AUFTRAGS- & DRUCKERINFORMATIONEN
50
44
EMF
42
DRUCKPROZESSOR (TREIBER RÜCKSETZEN)
52
SYSTEM-APIS
54
REGISTER-DATENBANK
AUFTRAG AUF NEUEM TREIBER DRUCKEN
NEUER TREIBER
40
GDI
GDI-PLAY-SPOOL-STREAM
56
TORÜBERWACHUNGS-EINRICHTUNG / DRUCKANBIETER
58
GERÄT
14

Fig. 5

TREIBERSCHALTEN: MEGHRFACHDURCHLAUFAUFTRÄGE FÜR VERTEILUNG

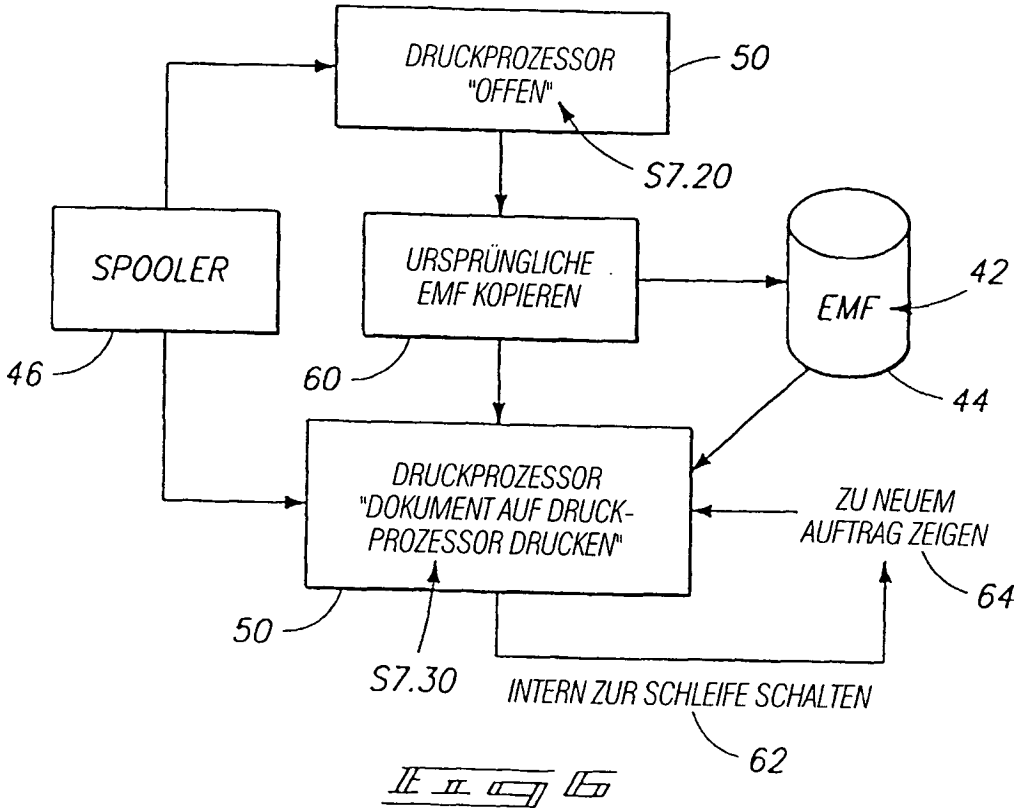

Fig 6

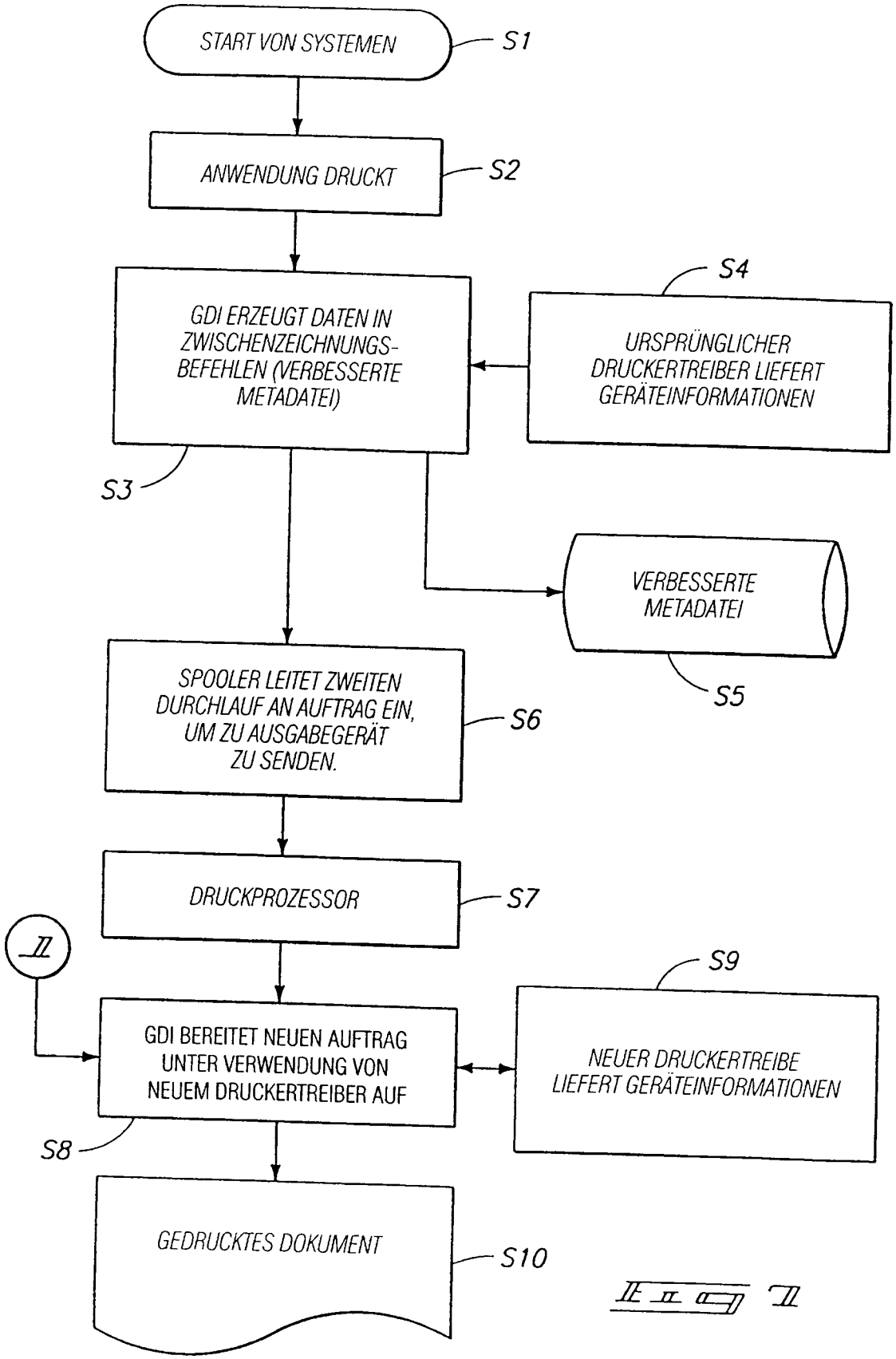
START VON SYSTEMEN
S1
ANWENDUNG DRUCKT
S2
GDI ERZEUGT DATEN IN ZWISCHENZEICHNUNGS-BEFEHLEN (VERBESSERTE METADATEI)
S4
URSPRÜNGLICHER DRUCKERTREIBER LIEFERT GERÄTEINFORMATIONEN
S3
VERBESSERTE METADATEI
S5
SPOOLER LEITET ZWEITEN DURCHLAUF AN AUFTRAG EIN, UM ZU AUSGABEGERÄT ZU SENDEN.
S6
DRUCKPROZESSOR
S7
II
S9
GDI BEREITET NEUEN AUFTRAG UNTER VERWENDUNG VON NEUEM DRUCKERTREIBER AUF
NEUER DRUCKERTREIBE LIEFERT GERÄTEINFORMATIONEN
S8
GEDRUCKTES DOKUMENT
S10

Fig. 11

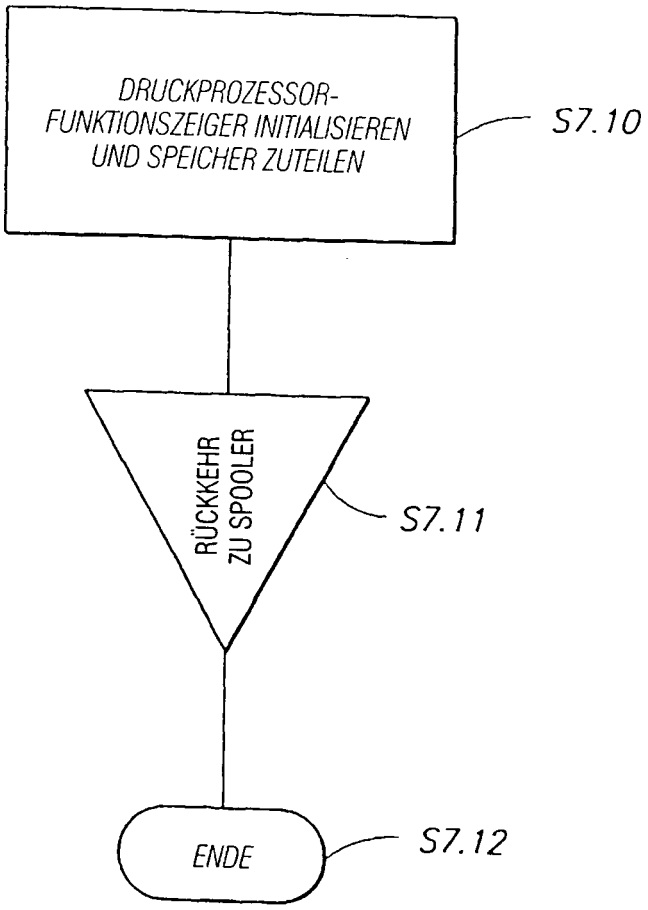
DRUCKPROZESSOR-
FUNKTIONSZEIGER INITIALISIEREN
UND SPEICHER ZUTEILEN
S7.10
RÜCKKEHR
ZU SPOOLER
S7.11
ENDE
S7.12

Fig. 8B

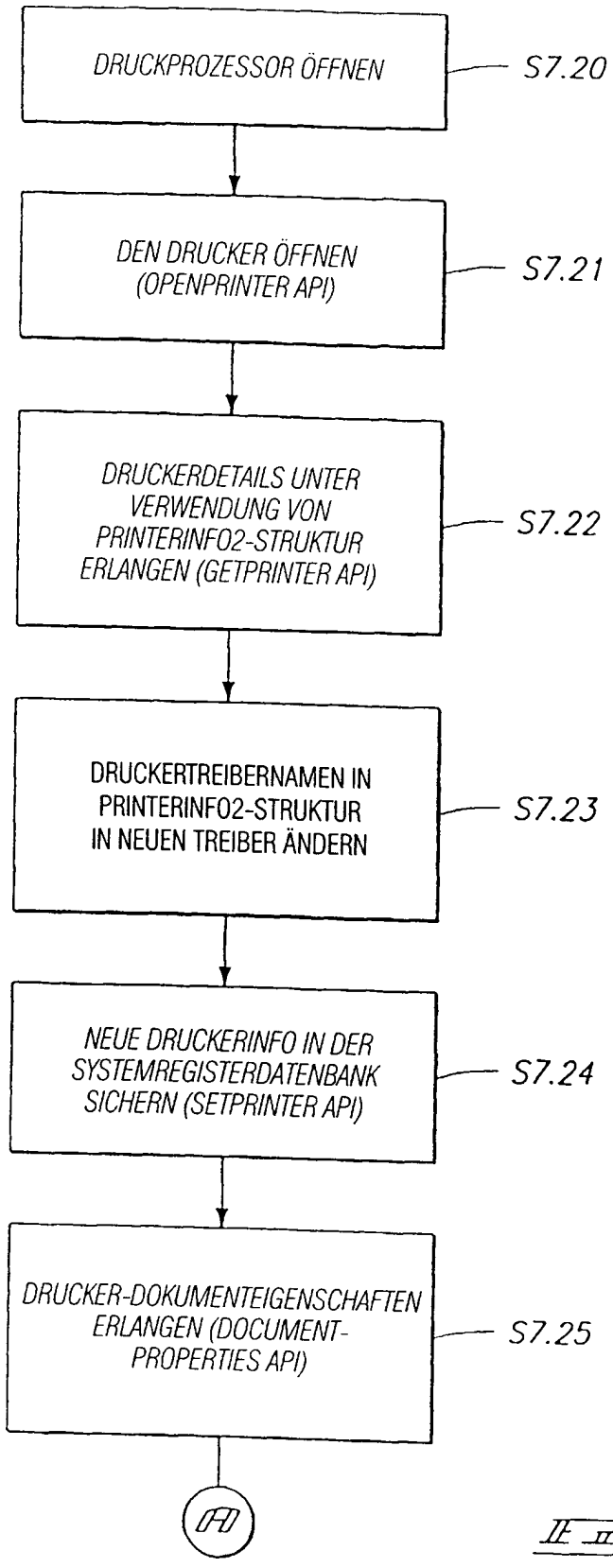
DRUCKPROZESSOR ÖFFNEN
S7.20
DEN DRUCKER ÖFFNEN (OPENPRINTER API)
S7.21
DRUCKERDETAILS UNTER VERWENDUNG VON PRINTERINFO2-STRUKTUR ERLANGEN (GETPRINTER API)
S7.22
DRUCKERTREIBERNAMEN IN PRINTERINFO2-STRUKTUR IN NEUEN TREIBER ÄNDERN
S7.23
NEUE DRUCKERINFO IN DER SYSTEMREGISTERDATENBANK SICHERN (SETPRINTER API)
S7.24
DRUCKER-DOKUMENTEIGENSCHAFTEN ERLANGEN (DOCUMENT-PROPERTIES API)
S7.25
A

Fig 9

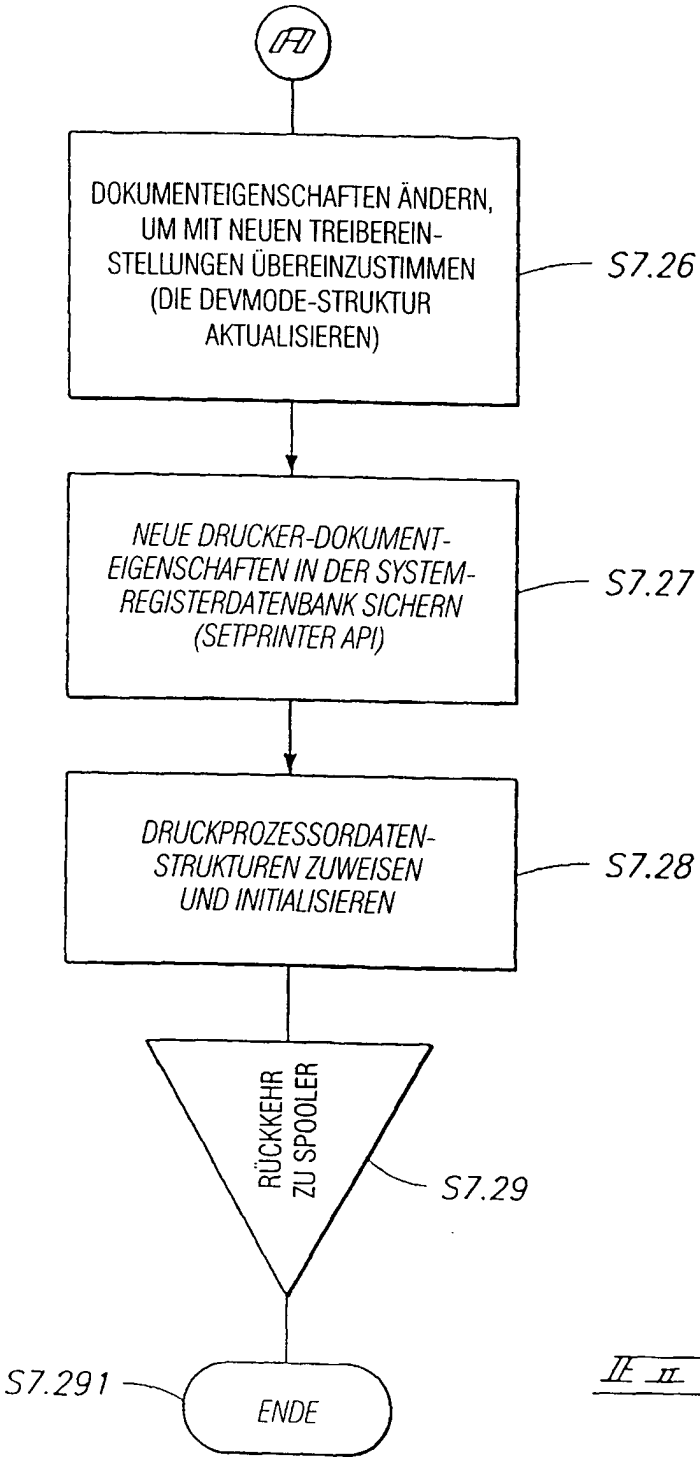
A
DOKUMENTEIGENSCHAFTEN ÄNDERN, UM MIT NEUEN TREIBEREIN-STELLUNGEN ÜBEREINZUSTIMMEN (DIE DEVMODE-STRUKTUR AKTUALISIEREN)
S7.26
NEUE DRUCKER-DOKUMENT-EIGENSCHAFTEN IN DER SYSTEM-REGISTERDATENBANK SICHERN (SETPRINTER API)
S7.27
DRUCKPROZESSORDATEN-STRUKTUREN ZUWEISEN UND INITIALISIEREN
S7.28
RÜCKKEHR ZU SPOOLER
S7.29
S7.291
ENDE

Fig 10

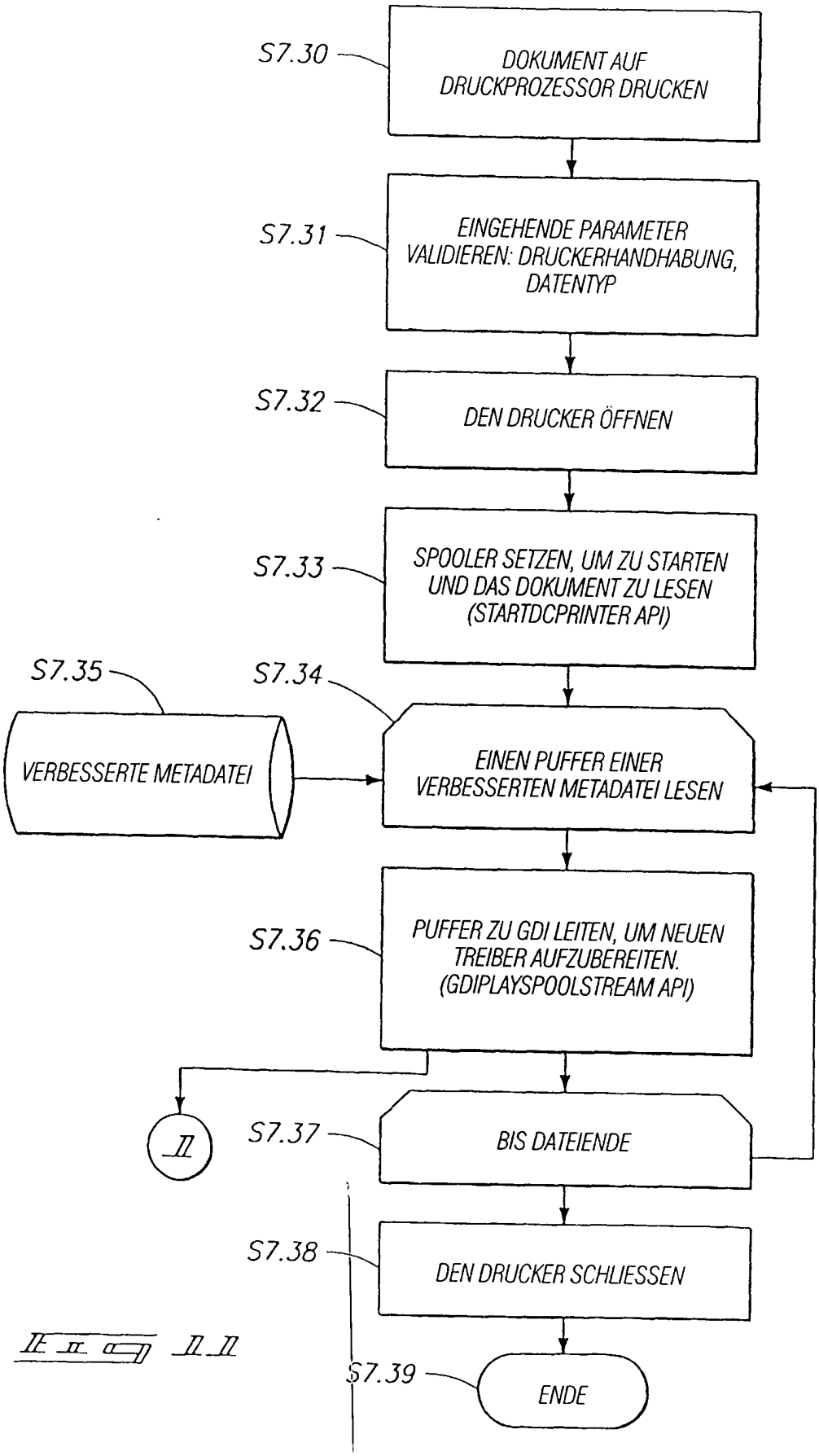
S7.30
DOKUMENT AUF DRUCKPROZESSOR DRUCKEN
S7.31
EINGEHENDE PARAMETER VALIDIEREN: DRUCKERHANDHABUNG, DATENTYP
S7.32
DEN DRUCKER ÖFFNEN
S7.33
SPOOLER SETZEN, UM ZU STARTEN UND DAS DOKUMENT ZU LESEN (STARTDCPRINTER API)
S7.35
S7.34
VERBESSERTE METADATEI
EINEN PUFFER EINER VERBESSERTEN METADATEI LESEN
S7.36
PUFFER ZU GDI LEITEN, UM NEUEN TREIBER AUFZUBEREITEN. (GDIPLAYSPOOLSTREAM API)
II
S7.37
BIS DATEIENDE
S7.38
DEN DRUCKER SCHLIESSEN
S7.39
ENDE

Fig 11