(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第3683925号
(P3683925)
(45)発行日　平成17年8月17日(2005.8.17)　(24)登録日　平成17年6月3日(2005.6.3)

(51)Int.Cl.[7]　F I
G06T 1/00　G06T 1/00　200D
G06F 17/30　G06F 17/30　210A
G06F 17/30　230A

請求項の数 26　(全 23 頁)

(21)出願番号　特願平6-285260
(22)出願日　平成6年11月18日(1994.11.18)
(65)公開番号　特開平8-147446
(43)公開日　平成8年6月7日(1996.6.7)
審査請求日　平成13年11月15日(2001.11.15)

(73)特許権者　000001007
キヤノン株式会社
東京都大田区下丸子３丁目３０番２号
(74)代理人　100090538
弁理士　西山　恵三
(74)代理人　100096965
弁理士　内尾　裕一
(72)発明者　横溝　良和
東京都大田区下丸子３丁目３０番２号キヤノン株式会社内
(72)発明者　杉浦　進
東京都大田区下丸子３丁目３０番２号キヤノン株式会社内

審査官　伊知地　和之

最終頁に続く

(54)【発明の名称】電子ファイリング装置

(57)【特許請求の範囲】

【請求項１】

電子ファイリングを行なうためのファイリングドライバーと、前記ファイリングドライバーを実行するコンピュータとを有する電子ファイリング装置であって、

前記ファイリングドライバーは、アプリケーションソフトウエアからの印刷指示に応じて起動され、

前記コンピュータは、当該起動されたファイリングドライバーを実行することにより、当該アプリケーションソフトウエアから伝達される印刷用の描画情報を受信し、当該受信した印刷用の描画情報を文書蓄積に適したフォーマットに変換し、当該変換された描画情報を電子ファイリング用のデータベースに蓄積することを特徴とする電子ファイリング装置。

【請求項２】

前記ファイリングドライバーは、プリンタードライバーをエミュレートしたドライバーであって、前記アプリケーションソフトウエアからの印刷指示に応じて起動され、前記コンピュータは、当該起動されたファイリングドライバーを実行することにより、前記変換された描画情報を前記電子ファイリング用のデータベースに蓄積すると、印刷完了メッセージを送ることを特徴とする請求項１に記載の電子ファイリング装置。

【請求項３】

前記コンピュータは、前記ファイリングドライバーを実行することにより、更に、前記描画情報からキーワード検索用の情報を抽出してキーワードとして登録することを特徴と

する請求項１に記載の電子ファイリング装置。
【請求項４】
前記コンピュータは、前記ファイリングドライバーを実行することにより、前記描画情報の中にビットマップ画像情報が含まれる場合には、該描画情報を蓄積する際に圧縮して蓄積することを特徴とする請求項１に記載の電子ファイリング装置。
【請求項５】
前記コンピュータは、前記ファイリングドライバーを実行することにより、更に、前記描画情報を、情報の種類に応じて分割し、
当該分割された情報毎に特徴を抽出し、
当該抽出された特徴を前記データベースに蓄積することを特徴とする請求項１に記載の電子ファイリング装置。
【請求項６】
前記コンピュータは、前記描画情報に含まれる情報の種類が、文字部分か、ベクトル描画部分か、ビットマップ画像部分かに応じて分割することを特徴とする請求項５に記載の電子ファイリング装置。
【請求項７】
前記コンピュータは、前記ファイリングドライバーを実行することにより、更に、前記描画情報のうち、ビットマップ画像部分を文字認識して文字コード化し、
前記文字認識して得た文字コードは、前記文字部分と同様に扱うことを特徴とする請求項６に記載の電子ファイリング装置。
【請求項８】
前記コンピュータは、前記ファイリングドライバーを実行することにより、更に、前記描画情報からタイトル部分のタイトル情報を抽出し、
前記抽出されたタイトル情報を、タイトルのキーワード検索用情報として登録することを有することを特徴とする請求項３に記載の電子ファイリング装置。
【請求項９】
前記コンピュータは、前記ファイリングドライバーを実行することにより、更に、前記描画情報から表部分の表情報を抽出し、
前記抽出された表情報を前記データベースに保存することを特徴とする請求項１に記載の電子ファイリング装置。
【請求項１０】
前記コンピュータは、前記ファイリングドライバーを実行することにより、前記表情報に基づいて該表の領域内に文字が入るか判断し、入ると判断された文字を、表中の文字であると識別する情報とともに登録することを特徴とする請求項９に記載の電子ファイリング装置。
【請求項１１】
前記コンピュータは、前記ファイリングドライバーを実行することにより、更に、前記描画情報から画像部分の画像情報を抽出し、
前記抽出された画像情報を圧縮して前記データベースに蓄積することを特徴とする請求項１に記載の電子ファイリング装置。
【請求項１２】
前記アプリケーションソフトウエアは、スキャナから画像を読み込むためのスキャナアプリケーションであって、
前記コンピュータは、前記ファイリングドライバーを実行することにより、前記スキャナから読み込まれた画像が前記スキャナアプリケーションから伝達された場合、当該伝達された画像に対して文字認識処理を行って文字情報を抽出し、当該伝達された画像を当該抽出された文字情報と共に前記データベースに蓄積することを特徴とする請求項１に記載の電子ファイリング装置。
【請求項１３】
更に、前記データベースに蓄積された描画情報を検索する検索手段を有することを特徴

とする請求項１に記載の電子ファイリング装置。
【請求項１４】
アプリケーションソフトウエアと前記アプリケーションソフトウエアからの指示に応じて起動されるファイリングドライバーと前記ファイリングドライバーを実行するコンピュータとを有する装置において、前記コンピュータにおいて前記ファイリングドライバーを実行させることにより、電子ファイリングを行なうように制御する電子ファイリング方法であって、
前記アプリケーションソフトウエアからの印刷指示に応じて起動されたファイリングドライバーを前記コンピュータに実行させることにより、当該アプリケーションソフトウエアから伝達される印刷用の描画情報を受信し、前記受信した印刷用の描画情報を文書蓄積に適したフォーマットに変換し、前記変換された描画情報を電子ファイリング用のデータベースに蓄積することを特徴とする電子ファイリング方法。
【請求項１５】
前記ファイリングドライバーは、プリンタードライバーをエミュレートし、前記アプリケーションソフトウエアからの印刷指示に応じて起動されるドライバーであって、
前記アプリケーションソフトウエアからの指示に応じて起動されたファイリングドライバーを前記コンピュータに実行させることにより、前記変換された描画情報を前記電子ファイリング用のデータベースに蓄積すると、印刷完了メッセージを送ることを特徴とする請求項１４に記載の電子ファイリング方法。
【請求項１６】
前記ファイリングドライバーを前記コンピュータに実行させることにより、更に、前記描画情報からキーワード検索用の情報を抽出してキーワードとして登録することを特徴とする請求項１４に記載の電子ファイリング方法。
【請求項１７】
前記ファイリングドライバーを前記コンピュータに実行させることにより、前記描画情報の中にビットマップ画像情報が含まれる場合には、該描画情報を蓄積する際に圧縮して蓄積することを特徴とする請求項１４に記載の電子ファイリング方法。
【請求項１８】
前記ファイリングドライバーを前記コンピュータに実行させることにより、更に、前記描画情報を、情報の種類に応じて分割し、
該分割された情報毎に特徴を抽出して、該抽出された特徴を前記データベースに蓄積することを特徴とする請求項１４に記載の電子ファイリング方法。
【請求項１９】
前記描画情報に含まれる情報の種類が、文字部分か、ベクトル描画部分か、ビットマップ画像部分かに応じて分割することを特徴とする請求項１８に記載の電子ファイリング方法。
【請求項２０】
前記ファイリングドライバーを前記コンピュータに実行させることにより、更に、前記描画情報のうち、ビットマップ画像部分を文字認識して文字コード化し、
前記文字認識して得た文字コードは、前記文字部分と同様に扱うことを特徴とする請求項１９に記載の電子ファイリング方法。
【請求項２１】
前記ファイリングドライバーを前記コンピュータに実行させることにより、更に、前記描画情報からタイトル部分のタイトル情報を抽出し、
前記抽出されたタイトル情報を、タイトルのキーワード検索用情報として登録することを有することを特徴とする請求項１６に記載の電子ファイリング方法。
【請求項２２】
前記ファイリングドライバーを前記コンピュータに実行させることにより、更に、前記描画情報から表部分の表情報を抽出し、
前記抽出された表情報を前記データベースに保存することを特徴とする請求項１４に記

載の電子ファイリング方法。
【請求項２３】
前記ファイリングドライバーを前記コンピュータに実行させることにより、前記表情報に基づいて該表の領域内に文字が入るか判断し、入ると判断された文字を、表中の文字であると識別する情報とともに登録することを特徴とする請求項２２に記載の電子ファイリング方法。
【請求項２４】
前記ファイリングドライバーを前記コンピュータに実行させることにより、更に、前記描画情報から画像部分の画像情報を抽出し、
前記抽出された画像情報を圧縮して前記データベースに蓄積することを特徴とする請求項１４に記載の電子ファイリング方法。
【請求項２５】
前記アプリケーションソフトウエアは、スキャナから画像を読み込むためのスキャナアプリケーションであって、
前記ファイリングドライバーを前記コンピュータに実行させることにより、前記スキャナアプリケーションから伝達された画像に対して文字認識処理を行って文字情報を抽出し、当該伝達された画像を当該抽出された文字情報と共に前記データベースに蓄積することを特徴とする請求項１４に記載の電子ファイリング方法。
【請求項２６】
更に、前記データベースに蓄積された描画情報を検索することを特徴とする請求項１４に記載の電子ファイリング方法。
【発明の詳細な説明】
【０００１】
【産業上の利用分野】
本発明は、文書を電子的な手段によりファイリング整理する電子ファイリング装置に関する。
【０００２】
本発明は、紙の原稿をスキャンした画像と、ＤＴＰ文書とを統合化し、共通に扱える電子ファイリング装置に関するものである。
【０００３】
【従来の技術】
従来の電子ファイリング装置は、大型で高価な専用機であった。また、操作方法も専用のものなので、専門的に教育されたオペレーターが必要であるなど、使いにくい面があった。また、従来の電子ファイリング装置はパーソナルコンピュータとは独立した装置になっており、ＤＴＰ文書と共通に管理することはできなかった。
【０００４】
【発明が解決しようとしている課題】
小型で安価な電子ファイリング装置を実現する為には、市販のパーソナルコンピュータ（以下パソコン）をベースにしたファイリング技術を実現する事が考えられる。電子ファイリング装置に必要な機能要素は、スキャナー、ＯＣＲ、キーワード検索機能、（光磁気）ディスク装置等であり、「紙」に印刷された文書をスキャナーで読んでＯＣＲで文字情報をコード化し、キーワードを自動的に拾い出してディスク装置に一緒に保存していた。専用機であれば、これだけの機能で十分であるが、パソコンをベースに電子ファイリングを行う際には、対象とする文書は「紙」に印刷された文書だけではなく、キーボードから入力して作成されたＤＴＰ文書も統一的に扱う必要性が出てくる。ところが、ＤＴＰ文書というのは、作成したアプリケーションによってファイルフォーマットが全く異なり、それらをファイルの形で保存すると以下の様な問題が生じる。
【０００５】
（１）ネットワーク上にあるサーバーに保存したＤＴＰ文書を複数のパソコンからアクセスする場合、全てのパソコンに作成時に用いたアプリケーションと同じアプリケーション

が存在するという保証は無い。バージョンが違うだけでも問題である。
【０００６】
（２）ネットワーク上にアーキテクチャーの異なるパソコンが存在する場合、たとえ同一アプリケーションの作成したファイルでもファイル形式が異なるのが普通なので、相互にアクセスできない。
【０００７】
（３）市販のアプリケーションには、キーワードを抽出する機能など無いのが普通であるから、電子ファイリング装置としては使用できない。
【０００８】
（４）専用の検索ソフトを用意し、市販のアプリケーションからは検索ソフトを介して間接的にファイリングをする場合、ファイルを媒介するか、カット＆ペースト機能を媒介して文書情報を伝送する訳だが、操作手順が繁雑で時間がかかり、自動化も困難である。
【０００９】
（５）従来の電子ファイリング装置はパーソナルコンピュータとは独立した装置になっており、ＤＴＰ文書と共通に管理する事はできなかった。
【００１０】
【課題を解決するための手段】
上記課題を解決する為に、本発明の電子ファイリング装置は、電子ファイリングを行なうためのファイリングドライバーと、前記ファイリングドライバーを実行するコンピュータとを有する電子ファイリング装置であって、前記ファイリングドライバーは、アプリケーションソフトウエアからの印刷指示に応じて起動され、前記コンピュータは、当該起動されたファイリングドライバーを実行することにより、当該アプリケーションソフトウエアから伝達される印刷用の描画情報を受信し、当該受信した印刷用の描画情報を文書蓄積に適したフォーマットに変換し、当該変換された描画情報を電子ファイリング用のデータベースに蓄積することを特徴とする。
【００１１】
上記課題を解決する為に、本発明の電子ファイリング方法は、アプリケーションソフトウエアと前記アプリケーションソフトウエアからの指示に応じて起動されるファイリングドライバーと前記ファイリングドライバーを実行するコンピュータとを有する装置において、前記コンピュータにおいて前記ファイリングドライバーを実行させることにより、電子ファイリングを行なうように制御する電子ファイリング方法であって、前記アプリケーションソフトウエアからの印刷指示に応じて起動されたファイリングドライバーを前記コンピュータに実行させることにより、当該アプリケーションソフトウエアから伝達される印刷用の描画情報を受信し、前記受信した印刷用の描画情報を文書蓄積に適したフォーマットに変換し、前記変換された描画情報を電子ファイリング用のデータベースに蓄積することを特徴とする。
【００４４】
【目的】
本発明は、従来ハード／ソフト共にスタンドアロンの機械として提供されていた文書蓄積／検索装置を、市販されているコンピュータをベースにし、市販されている幾多のアプリケーションをフロントエンドとする文書蓄積／検索装置を提供する事を目的とする。
【００４５】
本発明は、デスクトップのパソコンを介して簡単に文書の蓄積／検索ができる様にすることで、小型で安価な電子ファイリング方法及びその装置を提供することを目的とする。
【００４６】
本発明は、従来行われていたキーワード検索に加えて、ＤＴＰ文書やスキャンされた画像の中に含まれる「表」や「円グラフ」「タイトル」「ビットマップ画像」等の有無や個数等によっても検索時間の短縮あるいは検索精度を上げる事を目的とする。
【００４７】
【実施例】

以下、図に基づき本発明の実施例を詳細に説明する。

【００４８】

図１は本実施例の電子ファイリング装置のブロックダイヤグラムである。１００はホストコンピュータであり、枠で囲まれた機能を実行する。ここでは１つの枠で表したが、この枠内に表した各機能は１つ或いは複数のＣＰＵで実行しても良いし、１つ或いは複数のホストコンピュータで実行しても良い。同図に於て、１、２、３は市販のアプリケーションであり、例えばスキャナーアプリ１、ワープロアプリ２、表計算アプリ３等を示している。これらのアプリケーション間でデータのやり取りをするには、ファイル４、５を介して行うのは非常に困難である。なぜなら、スキャナーアプリ、ワープロアプリ、表計算アプリ等が生成するファイルのフォーマットはそれぞれのベンダーが独自に定めたものなので、互換性は全く無いからである。アプリケーションに依存しないファイルフォーマットとして「ＴＩＦＦ」「ＰＩＣＴ」「ＥＰＳ」「ＧＩＦ」「ＴＧＡ」等の業界標準フォーマットは全てのアプリケーションに対して万能ではない。

【００４９】

コンピューターがＭａｃｉｎｔｏｓｈ（登録商標）やＷｉｎｄｏｗｓ（登録商標）の場合には、「カット＆ペースト」の機能を用いてやり取りする仕組みが用意されているが、フォーマットは「ＰＩＣＴ」か「ＧＩＦ」に限定されてしまう。よって、コンピュータが限定された場合であってもカット＆ペーストでのデータのやり取りはできない。

【００５０】

一方、これらのアプリケーションが印刷する際にプリンタードライバーに引き渡すデータは、Ｍａｃｉｎｔｏｓｈの場合はＱｕｉｃｋＤｒａｗ（登録商標）、Ｗｉｎｄｏｗｓの場合はＧＤＩ（登録商標）に統一されている。図１はＭａｃｉｎｔｏｓｈのアプリケーション１、２、３がプリンタードライバー６に印刷データを渡している時の様子を示している。この時、Ｐｏｓｔｓｃｒｉｐｔアプリケーションの一部に例外はあるが、ほとんどのアプリケーションがＱｕｉｃｋＤｒａｗの描画コマンド７、８、９をプリンタードライバー６に渡している。本発明はこの特徴に着目し、プリンタードライバー６の中に文書変換／蓄積機能を備え、ＱｕｉｃｋＤｒａｗを処理対象とした。

【００５１】

１０－１はＱｕｉｃｋＤｒａｗ－共通文書変換装置である。ＱｕｉｃｋＤｒａｗの描画コマンドはコンピュータに依存した描画コマンドであり、Ｍａｃｉｎｔｏｓｈのみに限定すれば殆んどのアプリケーションに共通の言語であるが、ＷｉｎｄｏｗｓではＧＤＩコマンドが用いられており、互換性は無い。従ってこのＱｕｉｃｋＤｒａｗ－共通文書変換装置１０－１によりデータベース１３にセーブしやすい一般的なドキュメントの形に変換を行う。

【００５２】

共通文書１２、１４はドローであってもペイントであっても、また混在していてもどの場合でも各々処理できる。ドローとは描画の線の太さ、色、座標点を指定する数値で表現する描画方法で、それを受け取った側も画像におとすにはその数値をデコードする機能がなければならない。一方、ペイントとはビットマップ画像の事であり、データのサイズは大きくなるが、受け取った側のデコード機能は不用である。

【００５３】

１０－２はＯＣＲ装置でビットマップ画像を認識してコード化する場合に用いるが、すでにコード状態であるドロー系の情報の場合にはこのＯＣＲ装置１０－２はバイパスする。

【００５４】

１０－３はキーワード抽出装置で、ＱｕｉｃｋＤｒａｗ－共通文書変換装置１０－１により変換して得た共通文書の中から、検索に使えそうな単語を拾いだし、その共通文書とセットでアクセスする為のものである。

【００５５】

１０－４はデータベースＡＰＩで、ローカルまたはネットワークを介して接続されているリモートのデータベース１３に接続する為のものである。

【００５６】
プリンタードライバーＡＰＩ６は、システムが提供するものに合わせる。このＡＰＩはプリンタが印刷だけを想定している通常の場合はデータの流れは一方通行であり、データベース１３から引き出した文書をＱｕｉｃｋＤｒａｗコマンドに逆変換してアプリケーション１、２、３側に送る事は出来ない。しかしながら、本実施例におけるプリンタードライバーＡＰＩは汎用ドライバーの形式とする。プリンタードライバーを介して双方向の文書交換をする為には、ＯＳやアプリケーションの変更が必要であるが、専用の文書検索ソフトウエアを用いた場合には双方向の文書交換が実現できる。
【００５７】
共通文書１２はＬＡＮ１１を介してデータベース１３に送られる。
【００５８】
この図１における装置を文書のファイリングシステムとして用いるには、通常スキャナー１５で原稿を読み取る。その制御はスキャナーアプリ１が行うが、そのアプリがスキャナー１５により読み取った原稿を印刷する場合にはやはりＱｕｉｃｋＤｒａｗ７に変換してプリンタードライバー６に伝達する。ＱｕｉｃｋＤｒａｗとは言っても、スキャン画像の場合には中身はビットマップであるから、１０－１でビットマップに戻し、必要に応じて圧縮回路（図示せず）によって画像圧縮を行う。
【００５９】
プリンタードライバー６は、パソコンのＯＳがサポートする全てのＱｕｉｃｋＤｒａｗコマンドをサポートしており、あらゆるアプリケーションからの印刷要求に応える。一方、プリンタードライバー６が一方通行であった時はアプリケーション１、２、３は自分の都合により一部のＱｕｉｃｋＤｒａｗコマンドしかサポートしなくても何等問題は無かった。ところが、プリンタードライバー６を双方向性にし、ＱｕｉｃｋＤｒａｗ－共通文書変換装置１０－１が出力するＱｕｉｃｋＤｒａｗコマンドを各アプリケーションで表示可能にする為には、やはり各アプリケーションが全てのＱｕｉｃｋＤｒａｗコマンドをサポートする必要がある。なぜなら、その文書は自分より高機能なアプリケーションが作成したものかも知れないからである。その為プリンタードライバー６は、アプリケーションに伝達しようとしたドロー系のコマンドがアプリケーションに理解できないと拒絶された場合には、それをペイント系のデータに変換して再度伝達をしなおす。
【００６０】
誤り制御機能の無いアプリケーションの為に、ＱｕｉｃｋＤｒａｗ運用に水準を設け、例えばＪＩＳの漢字コードの毎く第一、第二、第三水準に分類し、伝達しても理解されない事が分かっている場合には、あらかじめ低水準のＱｕｉｃｋＤｒａｗコマンドに変換して伝達しても良い。
【００６１】
図２は、検索アプリの位置づけを説明する図面である。同図に於て、図１と共通の機能要素には同一の番号を付与してある。図１では市販のアプリケーション１、２、３からデータベース１３への文書蓄積を説明したが、市販のアプリケーションはやはり文書蓄積／検索用に設計されている訳ではないから、スピーディーな検索には専用の検索アプリ１６が必要となる。このアプリはＱｕｉｃｋＤｒａｗ－共通文書変換装置１０－１と直接接続する。データフォーマットはＱｕｉｃｋＤｒａｗではなく共通文書フォーマット１７である。１０－１は汎用ドライバーとして構築されている。
【００６２】
図３は、ＯＣＲアクセラレーターを使う場合の実施例である。１０－５がそのアクセラレータである。アクセラレータは通常ハードウエアで実現する。ＯＣＲには変換時間がかかるので、専用のハードウエアに処理を代行させる事により高速化が図れる。
【００６３】
また、ＱｕｉｃｋＤｒａｗ－共通文書変換装置１０－１はプリンタードライバー／汎用ドライバーとしてソフトウエアで実現する。一方、ＯＣＲ１０－２、キーワード検索装置１０－３、データベースＡＰＩ１０－４はバックグラウンドで走る汎用ドライバーとしてソ

フトまたはハードで実現し、これらが処理をしている時でも、オペレーターの他の装置を中断しない。

【００６４】

図４はその関係を詳細に説明した図である。１０－６はプリントキュー、１０－７はバッファーメモリーである。ＱｕｉｃｋＤｒａｗ－共通文書変換装置１０－１はもっぱらプリンタードライバー６と一体で動作し、プリントキュー１０－６への出力はフォアグランドで行う。一方、ＯＣＲ１０－２以下はプリントキュー１０－６から文書データーをバックグランドで受取り、アプリケーション１、２、３、１６と同時並行的に処理する。バッファーメモリー１０－７は、プリントキュー１０－６が一時的に文書データーを蓄えておく為のものである。

【００６５】

図５は、対話式会議システムへの応用例である。同図に於て、左半分は図１のパソコンと同じ物であり、説明上必要な番号には「ａ」の枝番を付けてある。一方、右半分は第２のパソコンで、「ｂ」の枝番を付けたものは、「ａ」の枝番の付いた同じ番号の要素と同じ物である。第１のパソコン（左）にはスキャナー１５をそのまま残して説明してあるが、第２のパソコン（右）にはスキャナーが無い場合の状態を説明している。この場合、ＯＣＲ、キーワード検索、データーベースＡＰＩ機能は不用なので斜線で示した様に機能を止めて使う。ネットワーク上を流れるデーターが共通文書である限りは、アプリケーションとのインターフェースは共通文書であっても他のフォーマット、即ち、ＱｕｉｃｋＤｒａｗあっても構わない。スキャナー１５からのデータはプリンタードライバー６が共通文書に変換するので、何等問題無く、相互通信が可能である。

【００６６】

本実施例においてはプリンタードライバーの、専らファイリングの為に用いられている部分のみ説明するので、ここからは「ファイリングドライバー」と呼ぶ事にする。ファイリングドライバーの機能は以下の通りである。

【００６７】

（１）市販のアプリケーションからのファイリングを可能にする。

【００６８】

（２）形式はプリンタドライバーの形式で作成する。市販のアプリケーションから一般のプリンターに印刷する時と全く同じ操作により、ファイリングを実現する。すなわち、アプリケーションから発行されるＧＤＩ描画コマンドを所定のファイル形式に変換してハードディスクにファイルを保存する。ファイリングに特有のダイアログはファイリングドライバーが表示器（図示せず）に表示する。

【００６９】

（３）ネットワーク対応は、データベース自身がネットワーク対応しており、ファイリングドライバーとしては、特にネットワークを意識しなくてもネットワーク上のファイルサーバーにアクセスできる。

【００７０】

（４）ファイル形式

例えばマイクロソフト社のＷｉｎｄｏｗｓのＧＤＩの描画コマンドは、最終的に以下の情報を含むファイル形式でデータベースに保存される。

【００７１】

ｉ）ビットマップ画像

ｉｉ）文字情報

ｉｉｉ）レイアウト情報

ｉｖ）キーワード情報

ビットマップ画像はファイリングの基本となる情報で、「印刷ダイアログ」で選択した用紙サイズと同じサイズのビットマップ画像を、以下の圧縮方式のいずれかを用いて圧縮する。

【００７２】

ｉ）白黒２値画像　ＪＢＩＧ
ｉｉ）多値画像　ＪＰＥＧ
文字情報に関しては、本来のファイリングではスキャナーで読んだビットマップ画像にＯＣＲを掛けて文字情報を抽出する。ファイリングドライバーの中でもこの方法を踏襲するが、ＧＤＩコードでドライバーに渡されるので、ＧＤＩコマンドから直接文字情報を抽出する。すなわち、文字情報については、以下の２種類がある。
【００７３】
ｉ）ＯＣＲによって得られる文字情報
ｉｉ）ＧＤＩコマンドを解析して得られる文字情報
レイアウト情報については、上記ＯＣＲの前処理プログラムであるブロックセレクション（ＢＳ）情報から抽出する。すなわち、レイアウト情報については以下の２種類がある。
【００７４】
ｉ）ブロックセレクションで得られるレイアウト情報
ｉｉ）ＧＤＩコマンドを解析して得られるレイアウト情報
キーワード情報については、キーワード生成ルーチンを用いて文字情報の中から抽出する。
【００７５】
レイアウト情報抽出の方法は以下の通りである。
【００７６】
ＧＤＩコードを解析して、レイアウト情報を抽出する。レイアウト情報はページ情報とブロック情報とから成り、以下の手順でデーターの抽出を行う。
【００７７】
ｉ）ＧＤＩコード抽出（データ入力）
ｉｉ）ブロック分解
ｉｉｉ）ブロック情報抽出
ｉｖ）ページ情報抽出
ｖ）ＤＢアクセス（ファイル作成）
ブロック分解／情報抽出の方法は以下の通りである。
【００７８】
〈バンディングの結合〉
アプリから渡されるＧＤＩコードバンディングされている場合には、１ページ分全部のＧＤＩコードを取得しまとめる。
【００７９】
〈ビットマップ画像である事の判定方法〉
（１）アプリから渡されるＧＤＩコードの中から、ビットマップ画像（Ｉｍａｇｅ）として定義されている部分は、ＭＳ　Ｗｉｎｄｏｗｓ　ＳＤＫの記述により、容易にビットマップ画像である事が判定できる。目標１で全体画像をファイルに保存しているので、画像情報そのものは捨ててしまって構わない。ここで保存する情報は以下の通り。
【００８０】
ｉ）ビットマップ画像のＩＤ
ｉｉ）ビットマップ画像の種類のＩＤ（二値、ハーフトーン、カラー、フルカラー）
ｉｉｉ）ビットマップ画像のサイズ
ｉｖ）印刷画面上での位置情報
ｖ）ビットマップ画像の標準化色（８色）のヒストグラムの上位３種類
（標準化色とは、ＲＧＢ各色成分の上位３ビットをテーブルルックアップ方式で８色に変換する。）
などである。
【００８１】
（２）判定に成功したＧＤＩコードは、次の判定では無視する。
〈表である事の判定方法〉

Ｗｉｎｄｏｗｓ　ＯＳがファイリングドライバーに渡すＧＤＩコードを解析し、その中に表が含まれているかどうかを判別する方法は以下の通り。

【００８２】

（１）グループ化：アプリケーションから渡されるＧＤＩコードの中から、斜め線を無視し、縦線／横線のみを選別し、Ｘ座標またはＹ座標が「一致」する直線をグループにしていく。（一致する直線を加える毎にその直線の座標も「一致」の判定基準に加える）

（２）矩形領域算出：グループ化した直線のグループの内、Ｘ座標、Ｙ座標の最大値／最小値を「表」全体の矩形領域と推定する。

【００８３】

（３）セル数算出：発見したＸ座標及びＹ座標の種類の数から１を引いた数をＸ座標及びＹ座標のセル数と推定する。セル数がＸ、Ｙ方向共に２以上の時、それを「表」と推定する。

【００８４】

（４）セルサイズ算出：発見したＸ座標及びＹ座標の、数字が互いに接近する二点間の差分をセルサイズと推定する。

【００８５】

（５）表の判定：矩形領域の内部に文字があれば、そこを「表」と断定する。文字が無ければ図形と推定する。

【００８６】

（６）書式：「表」の内部の文字はまとめて、ＴＡＢ－Ｔｅｘｔ形式で表現する。

【００８７】

例えば、図６の様な表は図１７の様にＴＡＢ－Ｔｅｘｔ形式表記できる。

【００８８】

ただし図１７において、（Ｔ）はタブ、（ＣＲ）はキャリッジリターン。

【００８９】

あるいは図１８の様にＳＹＬＫ形式で表記しても良い。

〈図形の判定方法〉

（１）矩形領域の拡大：あるＧＤＩコードが矩形領域を持っている場合には、その矩形領域を「拡大された矩形領域」とし、直線の様に矩形領域を持たないＧＤＩコードの場合には、それが持つ２点が構成する矩形領域を「拡大された矩形領域」とする。

【００９０】

（２）コードのグループ化：「拡大された矩形領域」同士が重なるＧＤＩコード同士は、一つのグループに属すると判定する。

【００９１】

例えば、図７（Ａ）、（Ｂ）に示す様な図形があった場合、（Ａ）は直線の矩形領域が大きいので１個のグループと判定されるが、（Ｂ）は２個のグループと判定される。

【００９２】

（３）グループ化されたＧＤＩコーポネントのＸ座標およびＹ座標の最大値／最小値を「図形の矩形領域」とする。グループ化されたＧＤＩコンポーネント群を「ピクチャー」と呼ぶ。

【００９３】

（４）「ピクチャー」をコード化する。

【００９４】

コード化の基本ルールは図８に示す基本構文に従って図１９の様な項目の内容を記述する。

【００９５】

（５）図形を分類する

図形は図２０のルールに従って分類する。

【００９６】

〈テキスト部の判定〉

（１）テキストの連結：１文字単独で文字が現れた場合、フォントサイズから計算される「キャラクタ幅」に単独文字があれば、「単語」として連結する。「単語」はスペースを含まない。

【００９７】

（２）「単語」の連結：「単語」同士がスペースのサイズ程度に近接していれば、「行」として連結する。「行」はスペースを含み得る。

【００９８】

（３）「文」の連結：「行」同士が「フォント高さ」の３倍以内にあるときは、「文」として連結する。「文」は複数の改行を含み得る。行端の「単語」の連結はやらない。

（日本語の場合、１行の全てが「単語」として認識される場合がある。）

（４）「文」をコード化する：コード化の基本ルールは「ピクチャー」として類似しており、図９に示す基本構文に従って図２１の様な項目の内容を記述する。

【００９９】

〈ブロック情報抽出の方法〉

ブロック毎にＩＤ番号を振る。その順番は、

ｉ）自己ブロックＩＤ

ｉｉ）自己ブロック位置

ｉｉｉ）ブロックの大きさ

ｉｖ）ブロック属性（テキスト、図形、表、イメージ、グラフ、タイトル、文字コード）

ｖ）親ブロックＩＤ

ｖｉ）子ブロックＩＤ

ｖｉｉ）そのページの中での順序

【０１００】

〈ページ情報抽出の方法〉

タイトル部の判定方法は以下の通りである。

【０１０１】

ステップ１：

（１）最大フォントの判定：その文書の中で最もサイズの大きな「ストリング」を判定する。

【０１０２】

（２）場所の判定：最大サイズの「ストリング」が用紙の１／４より上（または下）にある時、それをタイトルと判定する。

【０１０３】

ステップ２：

（１）特徴フォントの判定：同一サイズであっても、アンダーライン、〈〉、「」、｛｝で特徴付けられた「ストリング」が用紙の１／４より上にある時、それをタイトルと判定する。

【０１０４】

ステップ３：

上記のいずれにも該当しない時、タイトルは無いものと判定する。

【０１０５】

〈紙の文書の電子ファイリングとの統合〉

図１０は、今まで述べてきたファイリングドライバーを用いた市販アプリケーションからのファイリングと、紙の文書の電子ファイリングとの統合化を説明する図である。説明の都合上、市販アプリケーションからのファイリングを「ＤＴＰファイリング」、紙の文書の電子ファイリングを「ＯＣＲファイリング」と呼ぶ事にする。

【０１０６】

図の中央にデータベース１３があり、それより左側には、今まで述べてきたファイリングドライバーを用いた市販のアプリケーションからのファイリング装置のブロック図が、右側には、紙の文書をスキャンしてファイリングする電子ファイリング装置のブロック図が

書いてある。
【０１０７】
図２と共通の構成要素には同一の番号が付与してある。
【０１０８】
同図に於て、２８はスキャナー、２９はそれをドライブする為のスキャナーモジュール、３１は電子ファイリングエンジン、３０は検索モジュールである。スキャナーモジュール２９と電子ファイリングエンジン３１は、それぞれスキャナーアプリケーション１と検索アプリケーション１６に符合する。点線３３で囲んで示す様に、スキャナーモジュール２９と電子ファイリングエンジン３１は電子ファイリングエンジン３１と一体になり、全体で「電子ファイリングアプリケーション」を形成している。
【０１０９】
電子ファイリングエンジン３１には、ブロックセレクション、ＯＣＲ、キーワード抽出という３つの大きな機能がある。スキャナーモジュール２９に指示してスキャナー２８から紙の原稿をスキャンして読み込みます。読み込んだビットマップ画像はスキャナーモジュール２９で二値化され、電子ファイリングエンジン内でブロックセレクションプログラムにより、レイアウト情報が抽出される。抽出される情報は、テキスト領域及び画像領域のいずれかのプロファイルを持った矩形領域情報である。
【０１１０】
画像領域を示す情報は、レイアウト検索用にデータベースに保存される。ブロックセレクションの結果に基づき、ＯＣＲソフトによってテキスト領域が文字コードに変換される。
【０１１１】
その後、キーワード抽出プログラムによりテキスト情報からキーワードとして使えそうな情報が抽出され、それぞれデーターベースＡＰＩ３２を介してデーターベース１３に保存される。保存された文書は、検索モジュール３０で簡単に見る事ができる。
【０１１２】
「ＯＣＲファイリング」はビットマップ画像からブロックセレクションプログラムによってレイアウト情報を抽出するので、「ＤＴＰファイリング」ほどには正確で詳細なレイアウト情報は得られないが、ビットマップ画像の有無、タイトルの有無、と言ったレベルの検索なら、両者共通に行える。ＯＣＲによって抽出したキャラクターコードは１００％正確ではない。しかし、抽出したキャラクターコードそのものを再利用するのではなく、全文検索用のインデックス構築の為に用いるので、そこそこに正確であれば良く、手で認識誤りを修正する必要はない。
【０１１３】
データベースＡＰＩ３２を介して保存される情報は、
ｉ）文書全体のビットマップ画像の圧縮結果
ｉｉ）ブロックセレクション結果
ｉｉｉ）ＯＣＲ結果
ｉｖ）候補文字情報
ｖ）ＯＣＲ結果から抽出したキーワード情報
等である。
【０１１４】
文書全体のビットマップ画像の圧縮方法はＪＢＩＧを用いる。ＪＢＩＧモノクロの文書の圧縮用であるが、階層符合化方式によりサムネール画像を簡単に作れる。
【０１１５】
図１１は、原稿の入力先としてＦａｘを取り込んでいるところを示す図である。Ｆａｘ３４で受診したＭＭＲ画像情報は、ＦａｘＩ／Ｆ３５でビットマップ画像に伸長され、電子ファイリングエンジン３１に引き渡される。それ以降は紙の原稿をスキャナーで読んだ場合と全く同じである。
【０１１６】
〈フローチャートの説明〉

以下、フローチャートを用いて、各部の動作を詳細に説明する。

【０１１７】

図１２は、アプリケーションプログラムと、ＯＳ（Ｗｉｎｄｏｗｓ）、ファイリングドライバーの動作を説明するフローチャートである。図中、左側がアプリケーション、中央がＯＳ、右側がファイリングドライバーの動作を示している。

【０１１８】

ＯＳがマルチタスクのものならもち論、シングルタスクの場合でも、アプリケーションとＯＳ、ドライバーの動作は見掛け上独立したタスクの様に振る舞う。図中点線の矢印で示してある部分は、実際には関数コールであるが、Ｃ++やクラス定義の中では、「メッセージ」という表現を用いる事が一般的であり、本実施例ではその例に習う事にする。

【０１１９】

アプリケーションがドキュメントを作成してそれを本発明の電子ファイルにファイリングする時、通常の印刷動作と全く同じ操作を行なう。印刷メニューを選択すると（ステップ２）、ＯＳに対して印刷開始メッセージが送られる。ＯＳはこの時種々のＯＳサービスを実行しながら（ステップ６）イベントループに入り（ステップ７）、アプリケーションからの印刷開始メッセージを受け付けると、ドライバーを起動させる（ステップ８）。

【０１２０】

ＯＳからのメッセージにより起動したファイリングドライバーは、レジスタセットやメモリ空間の確保等の初期化処理を実行（ステップ１４）した後、初期化完了メッセージをＯＳに返す。この間ＯＳはドライバーからの返事をループで待っている（ステップ９）。

【０１２１】

そしてアプリケーションに対し、ドライバーの準備が整った事を示すメッセージを送る（ステップ１０）。

【０１２２】

アプリケーションはステップ３のループから抜け出し、プリンターに描画すべきＧＤＩコードを出力する（ステップ４）。ＧＤＩコードを受け取ったＯＳは、プリンター（ドライバー）がＧＤＩプリンターならばＧＤＩコードのままで、またビットマッププリンターならばＯＳの中でＧＤＩコードをラスタライズしてプリンタードライバーに渡す（ステップ１１）。ファイリングドライバーは、プリンタードライバーをエミュレートしており、ＧＤＩプリンターかビットマッププリンターのどちらかである事を宣言することにより、そのどちらでも実現可能である。もしＧＤＩプリンターであると宣言すると、ＯＳからはＧＤＩコードがそのままの形で送られて来るので、ドライバーの中では容易にＧＤＩコードの解析ができるし、ビットマップが必要なら、ＯＳのシステムコールを呼んで、ラスタライズする事ができる。

【０１２３】

もしビットマッププリンターであると宣言すると、ＯＳの中でＧＤＩをビットマップに変換するＧＤＩサービス（ステップ１１）が実行され、ファイリングドライバーにはビットマップが渡される。Ｗｉｎｄｏｗｓの場合、たとえビットマップにしても、ビットマップ・コードの片すみに元のＧＤＩコードが記載されているから、本実施例ではビットマツプを宣言する事にする。

【０１２４】

ファイリングドライバーは受け取ったＧＤＩコードを解析し（ステップ１６）、原稿に書かれている内容の特徴を抽出し（ステップ１７）、それをデータベース１３に保存する（ステップ１８）。ドライバーからＯＳに対し、印刷完了のメッセージが届いたら、ドライバーの処理待ちになっているループ（ステップ１２）から抜け出し、印刷完了メッセージをアプリケーションに送る（ステップ１３）と共に、ファイリングドライバーを終了させ、メモリー空間からスワップアウトする。アプリケーションは、ＯＳからの印刷完了メッセージを受けて、印刷シーケンスを終了する（ステップ５）。

【０１２５】

図中、（＊１）で示した部分（ステップ１６及びステップ１７）の詳細な処理を図１３の

フローチャートを用いて説明する。

【０１２６】

図１３のフローチャートは、ファイリングドライバーでの動作である。ファイリングドライバーは、ＯＳ経由でアプリケーションから取得したＧＤＩコードを解析し、画像か（ステップ２１）、文字か（ステップ２２）、ドローか（ステップ２３）を区別する。画像を表すＧＤＩコードは、ＪＢＩＧ、ＪＰＥＧ、ＭＭＲ等の圧縮アルゴリズムを用いて圧縮され（ステップ２４）データベース１３の画像バッファに保存される。ドローとは、ベクトル情報の集合による線図形、面図形を用いた描画を言う。

【０１２７】

文字を表すＧＤＩコードはＲＡＭ（図示せず）内の文字バッファに蓄えられ、文字バッファの処理（ステップ２５）でタイトルであるかどうかの判定を受けた後、文字に関する各種の情報（文字コード、スタイル、文字色）を保存し、データベース１３のキーワードとして登録する。

【０１２８】

ドロー部分のＧＤＩコードは、ドロー情報の処理ルーチン（ステップ２７）に於て表、図形の判定を受け、また、統計処理を受ける。

【０１２９】

ＧＤＩコードが上記のいずれでも無い時は、制御コードであるから、ステップ２６に於て制御コードの処理を行う。制御コードの中身は、例えば紙サイズ指定であり、また、紙方向指定、解像度指定、余白指定、ページ番号指定、印刷枚数指定、改ページである。

【０１３０】

特徴対抽出処理（ステップ１７）であるステップ２４、または２５、２６、２７の後は、次のＧＤＩコードを取得する為の準備（ステップ２８）に入る。もし、制御コードが終了コードだった時は、ＧＤＩコードの解析は終了するが、そうでない場合には次のＧＤＩコードを取得する為に、ステップ２０にループする。

【０１３１】

図中、（＊２）で示すステップ２５の詳細な処理を、図１４のフローチャートを用いて説明する。

【０１３２】

図１４のフローチャートは、ファイリングドライバーでの動作である。ここでは文字バッファに蓄えられた文字情報を解析して、タイトル文字かどうかの判定を行い、キーワードとして登録している。

【０１３３】

ＧＤＩコードの場合、文字コードは一連のストリング（文章）になって表現されるとは限らない。ストリングのＧＤＩコードを吐き出すアプリケーションは、エディターと呼ばれるソフトウエア位で、高級なＤＴＰアプリケーションになればなるほど、むしろ一文字一文字描画する場合が多い。高級なＤＴＰアプリケーションは、ＧＤＩコードの表現能力を越えた高度の文章表現をするので、一文字一文字の描画になる事が多い。

【０１３４】

電子ファイルの場合それでは検索ができないので、文字を連結して単語や文章に変換する必要がある。それがステップ３０と３１であり、文字と文字との間隔がその文字の高さの１／２以上であれば文字に区切りと判断し（ステップ３２）、それ以上接近している場合は単語の一部と判断する（ステップ３７）。単語の一部と判断された場合には、単語の区切りを見つける為に、その時処理したＧＤＩコードの文字情報を文字バッファーに書き込み、次のＧＤＩコードを取得すべくこのルーチンを抜け、ステップ２８に進む。

【０１３５】

一方、文字と文字の間隔が、文字の高さの１／２より広い部分は、単語の区切りと判定し（ステップ３２）、データベース１３にキーワードとして登録する。その中でもタイトルと思われる文字はキーワードの中でも特にタイトルとして特別に保存し、タイトル文字として検索の用に供する。タイトルと判定するアルゴリズムは、対象とする文字が、原稿の

縦寸法の上から１／４以内にあると判断され（ステップ３３）、かつ、原稿中で最大サイズの文字とステップ３４で判断される場合にその文字をタイトルと判定し、タイトルであることを識別できる情報を付加する。それ以外の文字はタイトル以外の文字としてキーワード登録する（ステップ３６）。タイトル自身もキーワードである事は言うまでもない。

【０１３６】

図１３の中の（＊３）で示した部分（ステップ２７）の詳細な処理を、図１５のフローチャートを用いて説明する。

【０１３７】

図１５のフローチャートはファイリングドライバーでの動作であり、ドロー情報の処理を行なう。始めに、ＧＤＩコードが直線か否かを判別し（ステップ４０）、直線である場合にはさらに表であるかどうかを判別するルーチン（ステップ４１～ステップ４８）に入るが、直線以外の丸、だ円、四角、ポリゴン、斜目線、曲線等である場合には、単純にドロー画グループに加え（ステップ４９）データベース１３に保存する。

【０１３８】

ステップ４０で直線と判別された場合、すでにデータベース１３に登録済の直線グループの中の線と接するかどうかを判定する（ステップ４１）。判定方法は、その時処理対象となっている直線の持つ２点（始点、終点）のＸ座標とＹ座標のどれかが既に登録済の直線グループの中の線の持つ２点の座標のどれかと１つでも一致し、かつ、他の座標が所定範囲内にあれば接していると判別する。

【０１３９】

たとえば図１６の様に２本の線Ａ、Ｂが接している事を証明するには、

$X_1 = X_3$

をチェックするだけで良い。

【０１４０】

ＡもＢも斜目線でない事は分かっているから、

$X_1 = X_3 = X_4$

でもある。ＢのＹ座標は、Ａの始点と終点の間になければならないから、

$Y_4 \leqq Y_1 \leqq Y_3$

でなければならない。すなわち、２本の直線の有する各４種類のＸ、Ｙ座標の内、３つが一致する成分を調べ、もし一致する場合には、２つ共一致する成分を持つ直線の他方の成分の間に他の線分の同じ座標が入っていれば、接していると判別できる。これがＸ、Ｙ共に成り立つ時は、２本の線は直角を形成する。

【０１４１】

図１５に戻って説明すると、ステップ４１で、どの直線グループとも接しないと判別された直線は、それ自身が新たな直線グループを形成するものとして登録する（ステップ５０）。

【０１４２】

一方、接する事が分かった直線については、その直線グループの一員として加える（ステップ４２）。

【０１４３】

ある直線グループの有するＸ座標Ｙ座標の各々の最大、最小の４つにより定まる領域を、その直線グループの矩形領域と定める。この矩形領域は、新たな直線が加わる毎に再計算する（ステップ４３）。

【０１４４】

その直線グループの構成するＸ座標だけまたはＹ座標だけの種類の数を求め、それから１を引いた数をセル数として算出する（ステップ４４）。

【０１４５】

また同様に、Ｘ座標の成分と、Ｙ座標の成分を別々に大きさ順にならべ、互いに接近する座標同士の差分をセルサイズとして計算する（ステップ４５）。

【０１４６】

この直線グループの矩形領域の中に、文字が入る場合には（ステップ４６）、その文字を表中の文字として表中の文字であることを識別できる情報とともに特別に登録し（ステップ４７）、かつ、その直線グループを表と判定する（ステップ４８）。

【０１４７】

文字を全く含まない直線グループは表とは見なさず、通常の直線としてデータベース１３に登録する。

【０１４８】

【発明の効果】

以上説明した様に、本発明を用いれば紙の文書の電子ファイリングとＤＴＰアプリケーションで作成したコンピュータファイル文書とが統合的に電子ファイリングできるので、従来やや閉鎖的だった電子ファイリングがＤＴＰファイリングと共に効果的に、手軽に利用できる様になる。

【０１４９】

従来の電子ファイリングでは、特別に訓練されたオペレーターが必要であったが、本発明による電子ファイリングシステムによれば、自分の机の上から誰でも扱える様になり、紙文化は一新することができる。

【０１５０】

従来キーワード検索一辺倒だった電子ファイリングも、本発明を用いれば原稿の特徴量によるレイアウト検索が可能になり、例えば、タイトルが「野球」で、絵が２枚入っていて、表が１本入っており、キーワードが「長島茂雄」という検索が実行できるので、目的の文書が極めてスピーディーに探す事ができる。

【０１５１】

また、ビットマップの全体画像の圧縮方法は、ＪＢＩＧという階層符合化理論を用いているので、サムネール画像（小画像）表示による目視検索も使えるので、原稿に書かれた内容をチェックしながらの検索も可能になる。

【図面の簡単な説明】

【図１】本発明による電子ファイリング装置のブロックダイヤグラム。

【図２】検索アプリの位置づけを説明する図。

【図３】ＯＣＲアクセラレーターを使う場合の実施例を示す図。

【図４】デバイスドライバーの要部を説明した図。

【図５】対話式会議システムへの応用例を示す図。

【図６】Ｔｅｘｔ－Ｔａｂ形式による表の情報の表現方法を示す図。

【図７】図形のグループ化の為の２つの例を示す図。

【図８】ピクチャー部分をコード化する為の基本構文を示す図。

【図９】テキスト部分をコード化する為の基本構文を示す図。

【図１０】ファイリングドライバーと紙の文書の電子ファイリングとの統合化を示す図。

【図１１】ファクシミリ文書の電子ファイリングを示す図。

【図１２】実施例の全体処理のフローチャート。

【図１３】ＧＤＩコードの解析及び特徴抽出処理のフローチャート。

【図１４】文字バッファの処理のフローチャート。

【図１５】ドロー情報の処理のフローチャート。

【図１６】直線の解析を説明する図。

【図１７】ＴＡＢ－Ｔｅｘｔ形式表記例示図。

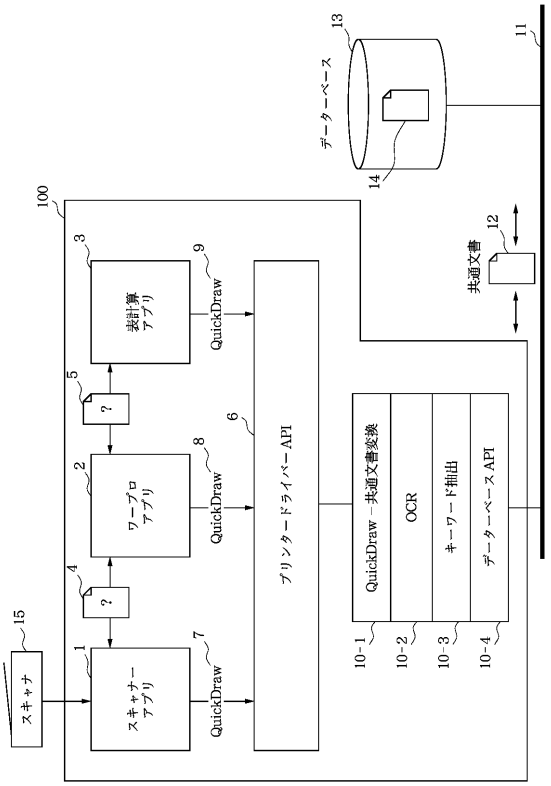
【図１８】表のＳＹＬＫ形式表記例示図。

【図１９】ピクチャーのコード化記述例示図。

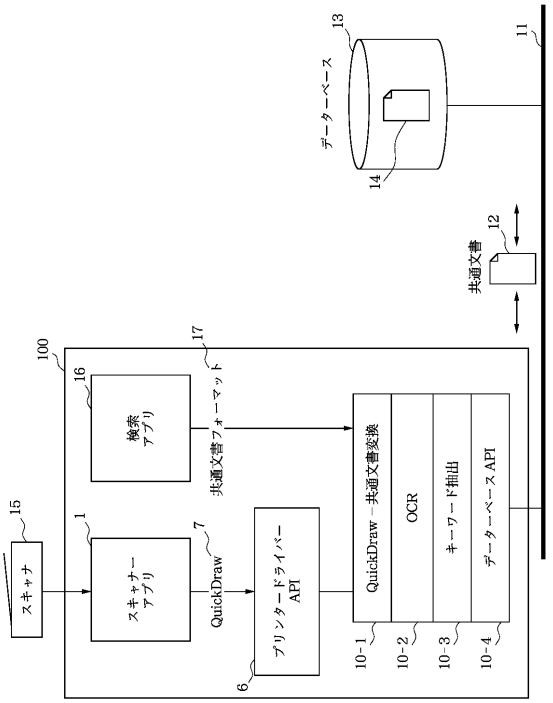
【図２０】図形の分類例を示す図。

【図２１】文のコード化記述例示図。

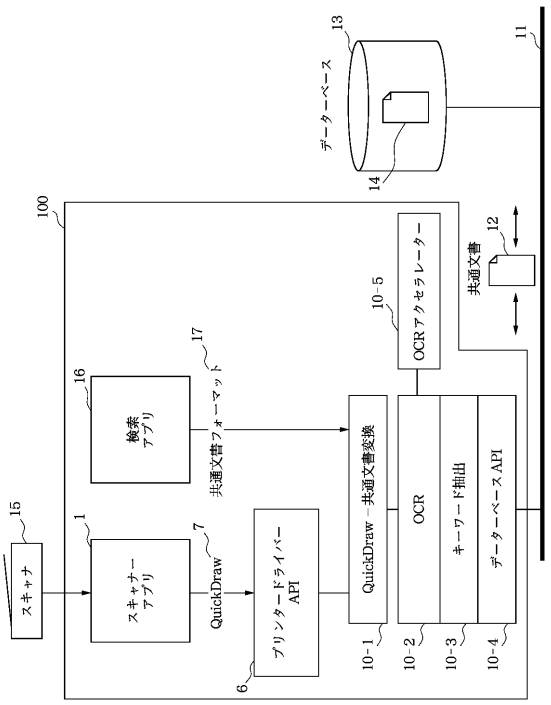
【図１】


スキャナ
スキャナーアプリ
ワープロアプリ
表計算アプリ
？
？
QuickDraw
プリンタードライバーAPI
QuickDraw－共通文書変換
OCR
キーワード抽出
データーベースAPI
データーベース
共通文書
100
1
2
3
4
5
6
7
8
9
10-1
10-2
10-3
10-4
11
12
13
14
15

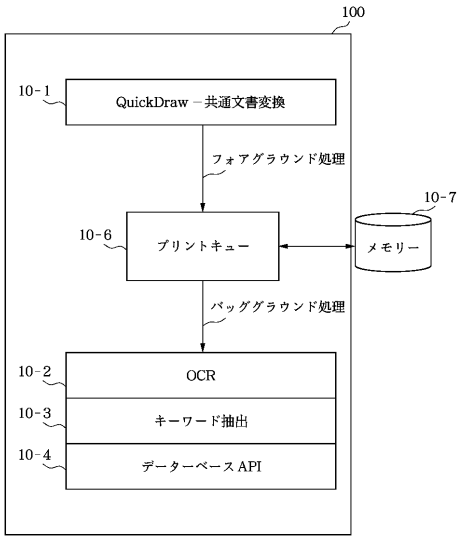
【図２】


スキャナ
スキャナーアプリ
検索アプリ
共通文書フォーマット
QuickDraw
プリンタードライバーAPI
QuickDraw－共通文書変換
OCR
キーワード抽出
データーベースAPI
データーベース
共通文書
100
1
6
7
10-1
10-2
10-3
10-4
11
12
13
14
15
16
17

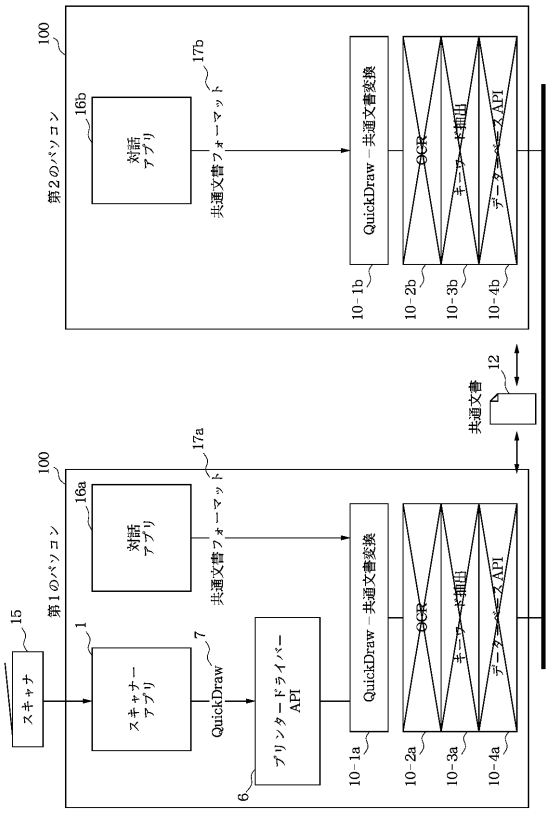
【図３】


スキャナ
スキャナーアプリ
検索アプリ
共通文書フォーマット
QuickDraw
プリンタードライバーAPI
QuickDraw－共通文書変換
OCR
キーワード抽出
データーベースAPI
OCRアクセラレーター
データーベース
共通文書
100
1
6
7
10-1
10-2
10-3
10-4
10-5
11
12
13
14
15
16
17

【図４】


100
10-1
QuickDraw－共通文書変換
フォアグラウンド処理
10-6
プリントキュー
10-7
メモリー
バックグラウンド処理
10-2
OCR
10-3
キーワード抽出
10-4
データーベースAPI

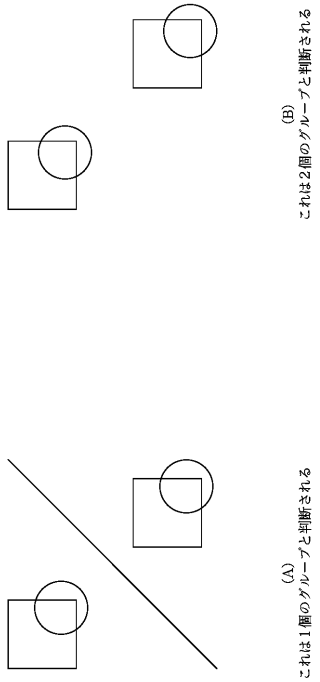
【図５】



【図６】

SelA1	SelB1	SelC1	SelD1
SelA2	SelB2	SelC2	SelD2
SelA3	SelB3	SelC3	SelD3

【図７】



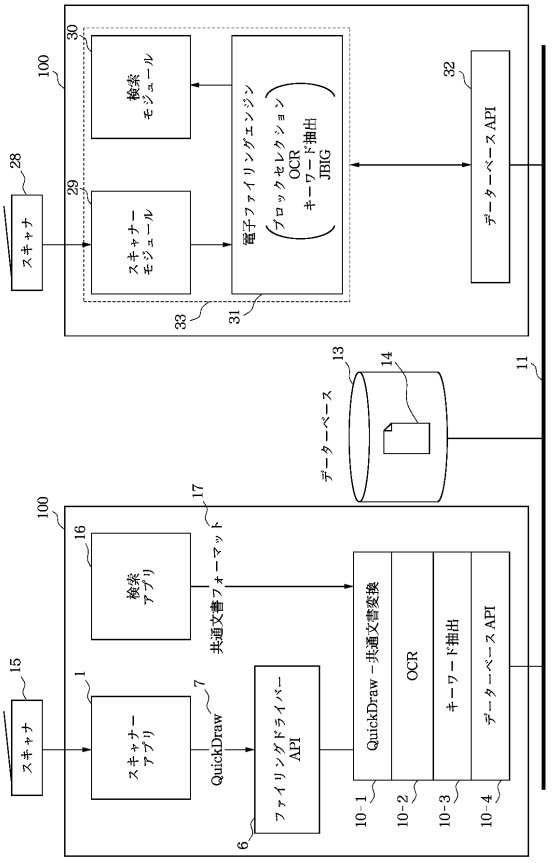
【図８】

	サイズ	意味
ID	2バイト	コンテンツの種類
レングス	2バイト	コンテンツに長さ（バイト）
コンテンツ	各種	(図形の記述)

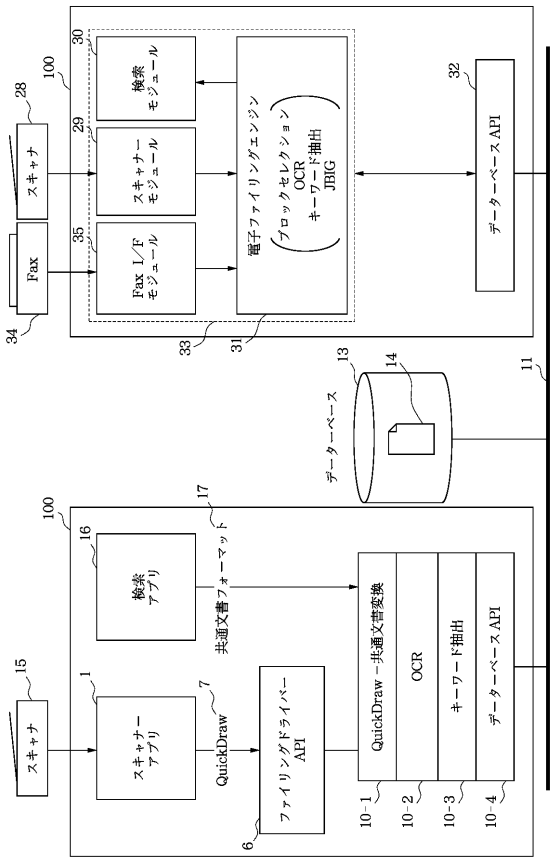
【図９】

ID	レングス	コンテンツ	
サイズ	2バイト	2バイト	各種
意味	コンテンツの種類	コンテンツに長さ（バイト）	（文の記述）

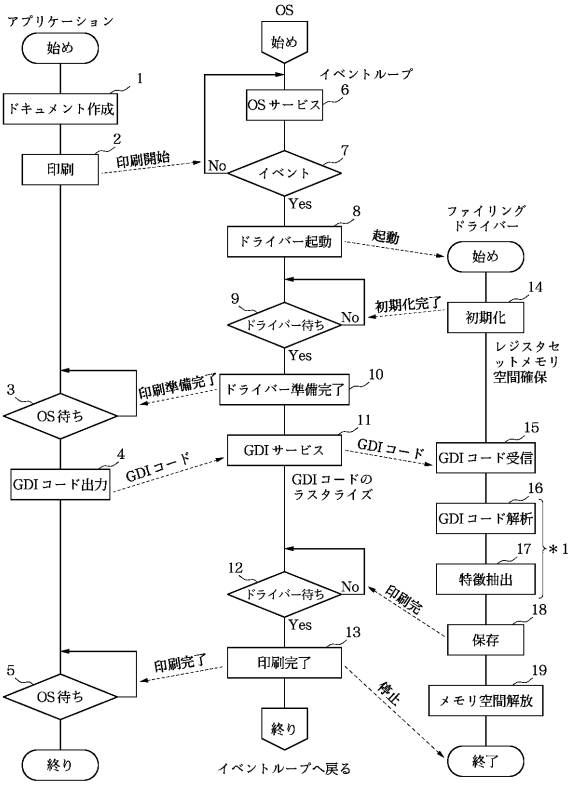
【図１０】



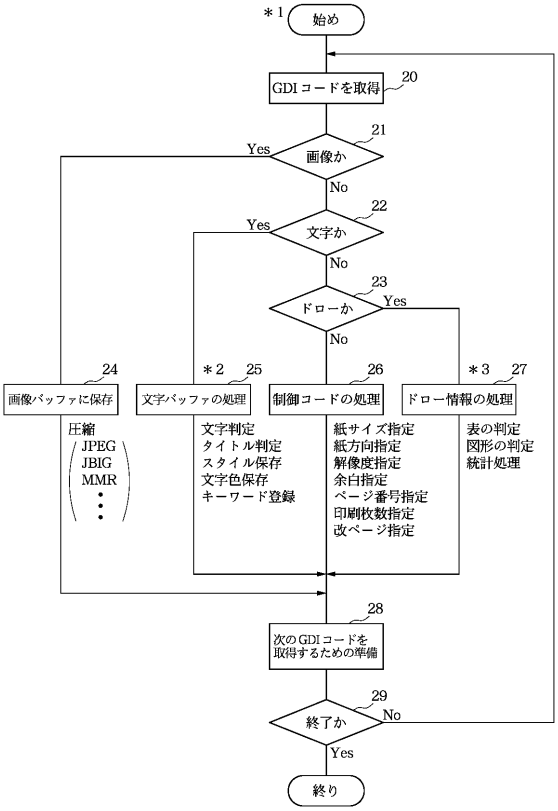
【図１１】



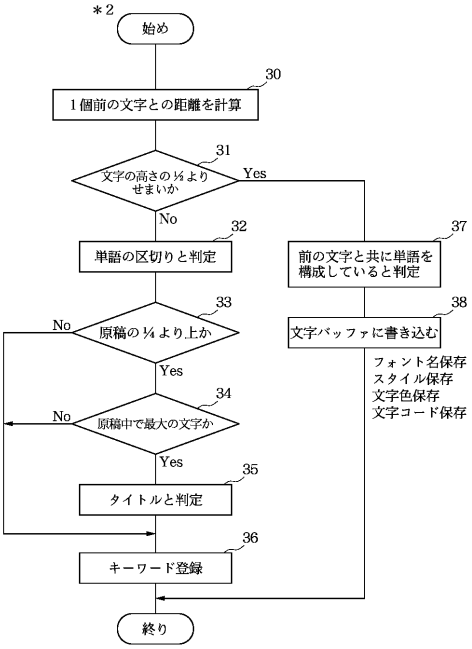
【図１２】



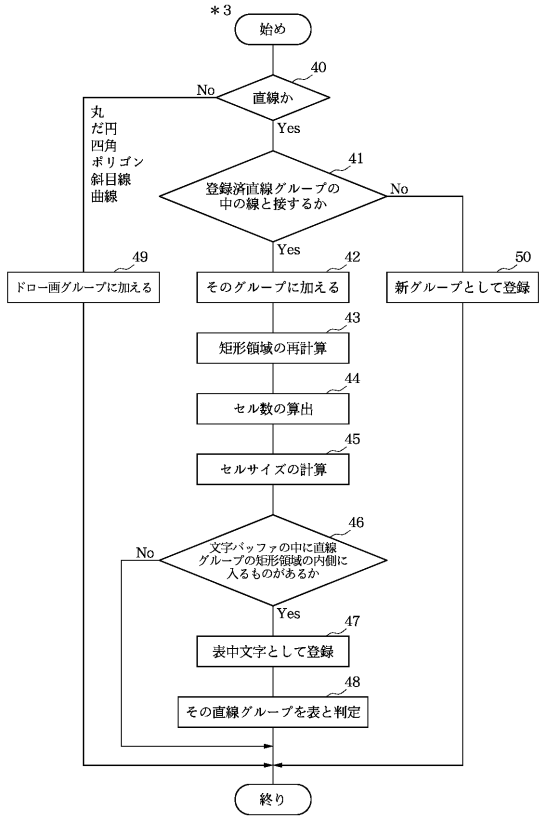
【図１３】



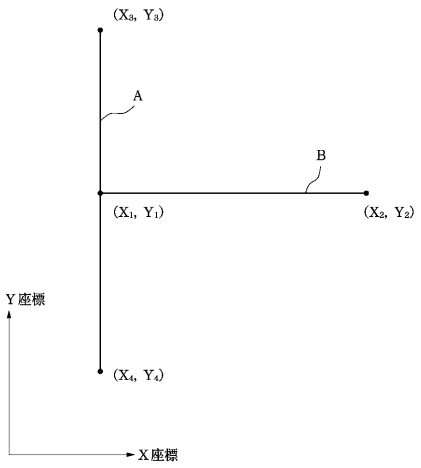
【図１４】



【図１５】



【図１６】



【図１７】

SelA1（T）	SelB1（T）	SelC1（T）	SelD1（CR）
SelA2（T）	SelB2（T）	SelC2（T）	SelD2（CR）
SelA3（T）	SelB3（T）	SelC3（T）	SelD3

【図１８】

```
TD;PWXL;N;E
P;PGeneral
P;P0
P;P0.00
P;P#, ##0
P;P#, ##0.00
P;P#, ##0;;\-#, ##0
P;P#, ##0;;[Red]\-#, ##0
P;P#, ##0.00;;\-#, ##0.00
P;P#, ##0.00;;[Red]\-#, ##0.00
P;P"\"#, ##0;;"\"\-#, ##0
P;P"\"#, ##0;;[Red]"\"\-#, ##0
P;P"\"#, ##0.00;;"\"\-#, ##0.00
P;P"\"#, ##0.00;;[Red]"\"\-#, ##0.00
P;P0%
P;P0.00%
P;P0.00E+00
P;P#\ ?/?
P;P#\ ??/??
P;Pyyyy/m/d
P;Pd\-mmm\-yy
P;Pd\-mmm
P;Pmmm\-yy
P;Ph:mm\ AM/PM
P;Ph:mm:ss\ AM/PM
P;Ph:mm
P;Ph:mm:ss
P;Pyyyy/m/d\ h:mm
P;P\_$#, ##0_);;\(\_$#, ##0\)
P;P\_$#, ##0_);;[Red]\(\_$#, ##0\)
P;P\_$#, ##0.00_);;\(\_$#, ##0.00\)
P;P\_$#, ##0.00_);;[Red]\(\_$#, ##0.00\)
P;Pr/m/d
P;Pr"年"m"月"d"日"
P;Prr"年"m"月"d"日"
P;Pm/d/yy
P;Pyyyy"年"m"月"d"日"
P;Ph"時"mm"分"
P;Ph"時"mm"分"ss"秒"
P;FOsaka;M240
P;FOsaka;M240;SB
P;FOsaka;M240;SI
P;FOsaka;M240;SBI
P;FOsaka;M240
F;P0;DG0G10;SM0;M320;N16384 12
B;Y3;X4;D0 0 2 3
O;L;D;V4;K47;G100 0.001
C;Y1;X1;K"SelA1"
C;X2;K"SelB1"
C;X3;K"SelC1"
C;X4;K"SelD1"
C;Y2;X1;K"SelA2"
C;X2;K"SelB2"
C;X3;K"SelC2"
C;X4;K"SelD2"
C;Y3;X1;K"SelA3"
C;X2;K"SelB3"
C;X3;K"SelC3"
C;X4;K"SelD3"
E
```

【図１９】

＜名称＞	＜コード＞	＜レングス＞	＜型＞
ピクチャーの始まり	TOP__OF__PICTURE	0	なし
ピクチャーの矩形領域	PICTURE__RECT	8	Rect
直線	LINE	8	Point x 2
四角	FRAME__RECT	8	Rect
四角（べた塗り）	FILL__FRAME__RECT	8	Rect
円、楕円	FRAME__OVAL	8	Rect
円、楕円（べた塗り）	FILL__FRAME__OVAL	8	Rect
ポリゴン	FRAME__POLY	不定形	Poligon
ポリゴン（べた塗り）	FILL__FRAME__POLY	不定形	Poligon
角の丸み	OVAL__SIZE	4	Point
ペンのサイズ	PEN__SIZE	4	Point
ペンのモード	PEN__MODE	2	Short
ペンのパターン	PEN__PAT	8	
ペンの色	FORE__COLOR	4	RGBColor
背景の色	BACK__COLOR	4	RGBColor
ピクチャーの終わり	END__OF__PICTURE	0	なし

【図２０】

図形の位置	全体、右上、左上、左下、右下
図形の特徴	線が多い、円が多い、四角が多い、楕円が多い、ポリゴンが多い、等
図形の色	標準化色（８色）の内上位３種類 （標準化色とは、RGB各色成分の上位３ビットをテーブルルックアップ方式で８色に変換する。）

【図２１】

＜名称＞	＜コード＞	＜レングス＞	＜型＞
ピクチャーの始まり	TOP__OF__PICTURE	0	なし
ピクチャーの矩形領域	PICTURE__RECT	8	Rect
フォントの名称	FONT	2	Short
フォントのサイズ	FONT__SIZE	2	Short
フォントのスタイル	FONT__FACE	2	Short
文	TEXT	不定形	
ピクチャーの終わり	END__OF__PICTURE	0	なし

フロントページの続き

(56)参考文献 特開平０６－０７６０４２（ＪＰ，Ａ）
特開平０６－２６６８１４（ＪＰ，Ａ）
特開平０６－２７４５５１（ＪＰ，Ａ）
特開平０５－１４５７３６（ＪＰ，Ａ）
特開平０４－１０５１７８（ＪＰ，Ａ）
特開平０５－２７４４７１（ＪＰ，Ａ）
特開平０３－２５０２７８（ＪＰ，Ａ）
特開平０６－２１５０８８（ＪＰ，Ａ）
特開平０５－１８１８６４（ＪＰ，Ａ）
特開平６－２１４９１８（ＪＰ，Ａ）
特開昭６１－１００８６７（ＪＰ，Ａ）
特開平６－２８２５７１（ＪＰ，Ａ）
渋谷 壮一，”プリンター・ドライバーの謎Ｉ”，日経MAC，日経ＢＰ社，１９９３年１０月１５日，第7号，ｐ．１８６－１９３

(58)調査した分野(Int.Cl.[7]，ＤＢ名)
G06T 1/00
G06F 17/30
G06F 12/00 - 12/00,549
G06F 3/12