(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号

特許第4250470号
(P4250470)

(45)発行日　平成21年4月8日(2009.4.8)　(24)登録日　平成21年1月23日(2009.1.23)

(51)Int.Cl.		F I		
G06F 3/12	(2006.01)	G06F	3/12	C
B41J 21/00	(2006.01)	B41J	21/00	Z
G06T 3/40	(2006.01)	G06T	3/40	A
H04N 1/387	(2006.01)	H04N	1/387	101

請求項の数 18　(全 24 頁)

(21)出願番号　特願2003-197271 (P2003-197271)
(22)出願日　平成15年7月15日 (2003.7.15)
(65)公開番号　特開2004-152255 (P2004-152255A)
(43)公開日　平成16年5月27日 (2004.5.27)
審査請求日　平成17年12月13日 (2005.12.13)
(31)優先権主張番号　特願2002-261979 (P2002-261979)
(32)優先日　平成14年9月6日 (2002.9.6)
(33)優先権主張国　日本国(JP)

(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　100076428
弁理士　大塚　康徳
(74)代理人　100112508
弁理士　高柳　司郎
(74)代理人　100115071
弁理士　大塚　康弘
(74)代理人　100116894
弁理士　木村　秀二
(72)発明者　友松　美明
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

最終頁に続く

(54)【発明の名称】情報処理装置及び情報処理方法及び印刷制御プログラム

(57)【特許請求の範囲】
【請求項１】
アプリケーションプログラムにより作成された描画データをオペレーションシステムによりスプールさせ、当該スプールされた描画データに基づいて画像形成ジョブを生成するプリンタドライバを備える情報処理装置であって、
前記オペレーションシステムによるスプール処理のスプールサイズを減らす軽減モードを実施するか否かを判定する判定手段と、
前記判定手段により前記軽減モードを実施すると判定されると、出力サイズが低下された描画データを作成するように当該アプリケーションプログラムに対して指示する第一指示手段と、
前記第一指示手段による指示に従って前記アプリケーションプログラムにより生成された前記出力サイズが低下された描画データを画像形成デバイスでの画像形成処理に適合した出力サイズで展開することにより得られるラスタイメージを含む画像形成ジョブを生成する画像形成ジョブ生成手段と、
を有することを特徴とする情報処理装置。
【請求項２】
前記画像形成用デバイスは、ホストコンピュータから送られてくるラスタイメージデータに基づいて印刷処理を実行するホストベースプリンタであることを特徴とする請求項１に記載の情報処理装置。
【請求項３】

前記軽減モードを設定する設定手段を更に有し、
前記第一指示手段は、前記設定手段により前記軽減モードが設定された場合に前記アプリケーションプログラムによって出力サイズが低下された描画データが作成されるように指示することを特徴とする請求項１又は２に記載の情報処理装置。
【請求項４】
前記設定手段は、更に前記軽減モードを自動的に設定するかどうかを設定でき、前記判定手段は、前記アプリケーションプログラムからのスプール指示に応じて、前記描画データのデータ量が所定量以上の場合に前記軽減モードを実施すると判定することを特徴とする請求項３に記載の情報処理装置。
【請求項５】
スプールされた前記描画データをラスタイメージに展開する前記オペレーションシステムの展開モジュールに対して、前記画像形成デバイスでの画像形成処理に適合した出力サイズで展開するように指示する第二指示手段を更に有し、
前記画像形成ジョブ生成手段は、前記第二指示手段で指示された出力サイズで展開されたラスタイメージを含む画像形成ジョブを生成することを特徴とする請求項１乃至３のいずれか１項に記載の情報処理装置。
【請求項６】
１ページ内に複数ページ分の描画データを縮小して配置するように設定されている場合、前記第一指示手段は、前記アプリケーションプログラムにより作成される描画データの出力サイズを低下させて前記オペレーションシステムによりスプールするように指示することを特徴とする請求項１乃至５のいずれか１項に記載の情報処理装置。
【請求項７】
アプリケーションプログラムにより作成された描画データをオペレーションシステムによりスプールさせ、当該スプールされた描画データに基づいて画像形成ジョブを生成するプリンタドライバを備える情報処理装置における情報処理方法であって、
前記オペレーションシステムによるスプール処理のスプールサイズを減らす軽減モードを実施するか否かを判定する判定工程と、
前記判定工程で前記軽減モードを実施すると判定されると、出力サイズが低下された描画データを作成するように当該アプリケーションプログラムに対して指示する第一指示工程と、
前記第一指示工程による指示に従って前記アプリケーションプログラムにより生成された前記出力サイズが低下された描画データを画像形成デバイスでの画像形成処理に適合した出力サイズで展開することにより得られるラスタイメージを含む画像形成ジョブを生成する画像形成ジョブ生成工程と、
を有することを特徴とする情報処理方法。
【請求項８】
前記画像形成用デバイスは、ホストコンピュータから送られてくるラスタイメージデータに基づいて印刷処理を実行するホストベースプリンタであることを特徴とする請求項７に記載の情報処理方法。
【請求項９】
前記軽減モードを設定する設定工程を更に有し、
前記第一指示工程は、前記設定工程で前記軽減モードが設定された場合に前記アプリケーションプログラムによって出力サイズが低下された描画データが作成されるように指示することを特徴とする請求項７又は８に記載の情報処理方法。
【請求項１０】
前記設定工程では、更に前記軽減モードを自動的に設定するかどうかを設定でき、前記判定工程では、前記アプリケーションプログラムからのスプール指示に応じて、前記描画データのデータ量が所定量以上の場合に前記軽減モードを実施すると判定することを特徴とする請求項９に記載の情報処理方法。
【請求項１１】

スプールされた前記描画データをラスタイメージに展開する前記オペレーションシステムの展開モジュールに対して、前記画像形成デバイスでの画像形成処理に適合した出力サイズで展開するように指示する第二指示工程を更に有し、

前記画像形成ジョブ生成工程では、前記第二指示工程で指示された出力サイズで展開されたラスタイメージを含む画像形成ジョブを生成することを特徴とする請求項７乃至１０のいずれか１項に記載の情報処理方法。

【請求項１２】

１ページ内に複数ページ分の描画データを縮小して配置するように設定されている場合、前記第一指示工程は、前記アプリケーションプログラムにより作成される描画データの出力サイズを低下させて前記オペレーションシステムによりスプールするように指示することを特徴とする請求項７乃至１１のいずれか１項に記載の情報処理方法。

【請求項１３】

アプリケーションプログラムにより作成された描画データをオペレーションシステムによりスプールさせ、当該スプールされた描画データに基づいて画像形成ジョブを生成する処理をコンピュータに実行させるために、該コンピュータを、

前記オペレーションシステムによるスプール処理のスプールサイズを減らす軽減モードを実施するか否かを判定する判定手段と、

前記判定手段により前記軽減モードを実施すると判定されると、出力サイズが低下された描画データを作成するように当該アプリケーションプログラムに対して指示する第一指示手段と、

前記第一指示手段による指示に従って前記アプリケーションプログラムにより生成された前記出力サイズが低下された描画データを画像形成デバイスでの画像形成処理に適合した出力サイズで展開することにより得られるラスタイメージを含む印刷ジョブを生成する印刷ジョブ生成手段とを有する印刷制御装置として機能させることを特徴とする印刷制御プログラム。

【請求項１４】

前記印刷装置は、ホストコンピュータから送られてくるラスタイメージデータに基づいて印刷処理を実行するホストベースプリンタであることを特徴とする請求項１３に記載の印刷制御プログラム。

【請求項１５】

前記軽減モードを設定する設定手段を更に有し、

前記第一指示手段は、前記設定手段により前記軽減モードが設定された場合に前記アプリケーションプログラムによって出力サイズが低下された描画データが作成されるように指示することを特徴とする請求項１３又は１４に記載の印刷制御プログラム。

【請求項１６】

前記設定手段は、更に前記軽減モードを自動的に設定するかどうかを設定でき、前記判定手段は、前記アプリケーションプログラムからのスプール指示に応じて、前記描画データのデータ量が所定量以上の場合に前記軽減モードを実施すると判定することを特徴とする請求項１５に記載の印刷制御プログラム。

【請求項１７】

スプールされた前記描画データをラスタイメージに展開する前記オペレーションシステムの展開モジュールに対して、前記印刷装置での印刷処理に適合した印刷サイズで展開するように指示する第二指示手段を更に有し、

前記印刷ジョブ生成手段では、前記第二指示手段で指示された印刷サイズで展開されたラスタイメージを含む印刷ジョブを生成することを特徴とする請求項１４乃至１６のいずれか１項に記載の印刷制御プログラム。

【請求項１８】

１ページ内に複数ページ分の描画データを縮小して配置するように設定されている場合、前記第一指示手段は、前記アプリケーションプログラムにより作成される描画データの印刷サイズを低下させて前記オペレーションシステムによりスプールするように指示する

ことを特徴とする請求項１４乃至１７のいずれか１項に記載の印刷制御プログラム。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】
本発明は、文字、図形等を含む描画データに基づいて、画像形成デバイスで形成すべきデータを生成する情報処理装置及び情報処理方法及び印刷制御プログラムに関するものである。
【０００２】
【従来の技術】
ホストコンピュータから送られてくるラスタイメージに基づいてプリンタ装置によりプリントを行うプリントシステムが知られている。このようなプリンタ装置は、インクジェットプリンタや電子写真プリンタの中でホストベースプリンタと呼ばれている。ホストベースプリンタに送信する印刷データにおけるラスタイメージの展開に際しては、ホストコンピュータのアプリケーション・ソフトウェア（以下、単にアプリケーション）により作成された文字、グラフィックス、写真などの描画データは、一旦、ホストコンピュータの仮想的なページメモリにオペレーションシステム（ＯＳ）の描画モジュール（米国マイクロソフト社のWindows（登録商標）OSでは、ＧＤＩ（Graphic Device Interface）と呼ばれる）を介してスプールファイル（ＥＭＦデータ）としてスプールされ、その後、スプールファイルを読み出して、プリンタドライバもしくはＯＳの描画モジュールにより展開される。
【０００３】
このようなアプリケーションからの描画データをオペレーションシステムのスプールファイルにスプールし、プリントプロセッサにより読み出してプリンタドライバまたはＯＳの描画モジュールにより展開（ラスタライズ）処理されるシステムがある。（特許文献１）
【特許文献１】
特開２００３－１６７７０１号公報
【０００４】
【発明が解決しようとする課題】
また、このようなラスタイメージデータへの展開時には、アプリケーションは、テキストデータの場合は文字種、キャラクタコードなどに加えて、その出力位置を示す座標情報を出力する。またグラフィックデータの場合は、形状などの種別、色、座標情報を出力し、また写真などのイメージ画像についてはオリジナル画像と出力先のサイズを出力する。
【０００５】
しかし、近年のアプリケーションの高機能化などに伴って、アプリケーションにより生成されるデータが複雑になってきている。従って、プリンタ装置やプリンタドライバの機能によっては、アプリケーションより受取ったデータを正しく解釈してイメージデータに展開できない場合が発生する。このため、一部のアプリケーションは、その作成したテキストデータ、グラフィックデータ等をプリンタ装置に出力する際、そのアプリケーションによりプリンタ装置のレンダリング解像度に応じたイメージデータに展開してから印刷を行うことがある。このようなアプリケーションでは、そのイメージ展開したデータをスプールする場合、そのスプールデータのサイズが非常に大きくなり、ハードディスクやメモリを圧迫するという問題が発生している。
【０００６】
また、イメージ展開するアプリケーションでなくとも同様の問題が考えられる。近年、高解像度のデジタルカメラの普及により、ホストコンピュータ上で高解像度のイメージデータを扱うようになってきている。そして、高解像度のイメージデータを文書のページに貼り付けて印刷させる場合、アプリケーションによっては、貼り付けられた領域の画素数に適合させずに、そのままオリジナルのイメージデータを出力する場合がある。このようなアプリケーションでは、出力したデータをスプールする場合、そのスプールデータのサイズが非常に大きくなり、ハードディスクやメモリを圧迫するという問題が発生している。

具体的には、オペレーションシステムによっては、スプールデータのサイズが４ＧＢ（ギガバイト）を超えると印刷エラーとなる場合や、また、アプリケーション側でもサポートしている出力サイズを超えて印刷エラーとなる場合や、また、ハードディスクの空き容量が足りずに印刷エラーとなる場合が発生している。

【０００７】

特に近年、プリンタの高画質化に伴うレンダリング解像度が益々高解像度になっているため、スプールデータのサイズがより大きくなり、アプリケーションやスプーラ内部でエラーが発生してしまい、正常にイメージに展開できなくなるという問題が発生している。その結果、レンダリング解像度を上げることもできずプリンタ装置の高画質化にとって障害となっている。

【０００８】

本発明は上記従来例に鑑みてなされたもので、画像形成デバイスにラスタイメージデータを出力するためのスプールファイルのサイズを小さくし、かつ高品位の画像を形成できる情報処理装置及び情報処理方法及び印刷制御プログラムを提供することを目的とする。

【０００９】

【課題を解決するための手段】

上記目的を達成するために本発明の情報処理装置は以下のような構成を備える。即ち、

アプリケーションプログラムにより作成された描画データをオペレーションシステムによりスプールさせ、当該スプールされた描画データに基づいて画像形成ジョブを生成するプリンタドライバを備える情報処理装置であって、

前記オペレーションシステムによるスプール処理のスプールサイズを減らす軽減モードを実施するか否かを判定する判定手段と、

前記判定手段により前記軽減モードを実施すると判定されると、出力サイズが低下された描画データを作成するように当該アプリケーションプログラムに対して指示する第一指示手段と、

前記第一指示手段による指示に従って前記アプリケーションプログラムにより生成された前記出力サイズが低下された描画データを画像形成デバイスでの画像形成処理に適合した出力サイズで展開することにより得られるラスタイメージを含む画像形成ジョブを生成する画像形成ジョブ生成手段と、

を有することを特徴とする。

【００１０】

上記目的を達成するために本発明の情報処理方法は以下のような工程を備える。即ち、

アプリケーションプログラムにより作成された描画データをオペレーションシステムによりスプールさせ、当該スプールされた描画データに基づいて画像形成ジョブを生成するプリンタドライバを備える情報処理装置における情報処理方法であって、

前記オペレーションシステムによるスプール処理のスプールサイズを減らす軽減モードを実施するか否かを判定する判定工程と、

前記判定工程で前記軽減モードを実施すると判定されると、出力サイズが低下された描画データを作成するように当該アプリケーションプログラムに対して指示する第一指示工程と、

前記第一指示工程による指示に従って前記アプリケーションプログラムにより生成された前記出力サイズが低下された描画データを画像形成デバイスでの画像形成処理に適合した出力サイズで展開することにより得られるラスタイメージを含む画像形成ジョブを生成する画像形成ジョブ生成工程と、

を有することを特徴とする。

【００１１】

上記目的を達成するために本発明の印刷処理プログラムは以下のような構成を備える。即ち、

アプリケーションプログラムにより作成された描画データをオペレーションシステムによりスプールさせ、当該スプールされた描画データに基づいて画像形成ジョブを生成する

処理をコンピュータに実行させるために、該コンピュータを、
前記オペレーションシステムによるスプール処理のスプールサイズを減らす軽減モードを実施するか否かを判定する判定手段と、
前記判定手段により前記軽減モードを実施すると判定されると、出力サイズが低下された描画データを作成するように当該アプリケーションプログラムに対して指示する第一指示手段と、
前記第一指示手段による指示に従って前記アプリケーションプログラムにより生成された前記出力サイズが低下された描画データを画像形成デバイスでの画像形成処理に適合した出力サイズで展開することにより得られるラスタイメージを含む印刷ジョブを生成する印刷ジョブ生成手段とを有する印刷制御装置として機能させることを特徴とする。

【００１２】
【発明の実施の形態】
以下、添付図面を参照して本発明の好適な実施の形態を詳細に説明する。
【００１３】
図１は、本発明の実施の形態に係る印刷システムの概略構成を示す図である。尚、この図１で示される実施の形態として、例えば一般的に普及しているパーソナルコンピュータ（本発明の情報処理装置に相当する）にMicrosoft社のWindows（登録商標）をオペレーティングシステム（以下、OSと呼ぶ）として使用し、印刷処理機能を有する任意のアプリケーション１０１をインストールし、更に、このコンピュータにプリンタ１０８を接続した形態が考えられる。
【００１４】
アプリケーション１０１が印刷要求した描画データ（文書データ、画像データ等の印刷データ）は、ＯＳの描画モジュールであるグラフィックデバイス・インターフェース（Graphics Device Interface：以下、「ＧＤＩ」）１０２経由でプリンタドライバ１０３に渡される。これによりプリンタドライバ１０３により印刷ジョブが生成され、この生成された印刷ジョブはプリンタ１０８へ印刷要求として出力される。ＧＤＩ１０２は、通常、一旦、そのデータをＥＭＦ（Enhanced Meta File）データとしてスプールファイル１０４にスプール（ＯＳのスプール処理）し、そのスプールが終了すると、プリントプロセッサ１０５に印刷要求を発行する。これによりプリントプロセッサ１０５は、スプールファイル１０４から印刷出力情報と描画データをページ単位で読み込み、１ページずつＧＤＩ１０２を介してプリンタドライバ１０３に出力して印刷するように要求する。このプリントプロセッサ１０５は、スプールファイル１０４にスプールされている描画データをプリンタドライバ１０３に出力する際、プリンタドライバ１０３から出力されるラスタイメージデータの解像度に合わせて座標情報を変換する機能がある。尚、このプリントプロセッサ１０５は、ＯＳのモジュールとして用意されているが、プリンタのメーカ等がカスタマイズしたプリントプロセッサ１０５をプリンタドライバ１０３とともに供給し、それらをハードディスクにインストールすることも許容されている。こうしてプリントプロセッサ１０５から印刷が要求されたプリンタドライバ１０３は、ＯＳの描画機能であるＧＤＩレンダリングエンジン１０７を用いて、その描画データをプリンタの印刷解像度に合わせてラスタイメージデータにレンダリングした後、色処理やプリンタコマンドの付加などを行い、印刷ジョブとして生成し、ポートモニタ１０６経由でプリンタ１０８に送信する。
【００１５】
このＯＳのスプール処理で作成されるスプールファイル１０４は、アプリケーション１０１が印刷を要求しているコマンド、或いはビットマップイメージ等をファイルとして保存している。この際、通常、文字については、文字種、文字列のキャラクタコードと座標情報などが保存され、写真などのイメージは、オリジナル画像情報と座標情報とが保存されている。これらキャラクタ、グラフィック及びイメージデータは、プリンタドライバ１０３で読み込まれた後、ＧＤＩレンダリングエンジン１０７でレンダリングされてプリンタ１０８の解像度に応じたラスタイメージデータに変換される。そのため、最近のプリンタ１０８における高画質化に伴う高解像度化においても、スプールデータ１０４のサイズは

大きくならず問題とならなかった。なお、ＧＤＩ１０２とＧＤＩレンダリングエンジン１０７は、説明を分かりやすくするために個別に記載したが、米国マクロソフト社のWindows（登録商標）OSでは、同一の描画出力モジュールで構成されている。また、プリンタドライバ１０３自体が、レンダリング機能を備えていてもよいが、ＧＤＩ１０２（１０７）がサポートする描画関数の拡張に備えて、その都度、プリンタドライバ１０３のレンダリング機能がサポートするように開発することはプリンタ企業にとって負担となるため、ＯＳの描画機能（ＧＤＩレンダリングエンジン１０７）を用いてレンダリングさせることが望ましい。

【００１６】

しかし、その描画データが複雑なデータ等の場合には、プリンタ１０８やプリンタドライバ１０３の機能によっては、正常に印刷できないことがある。このため、アプリケーション１０１によっては、レンダリングをアプリケーション１０１で行い、その結果であるラスタイメージをプリンタドライバ１０３に出力するものがある。このような場合、通常、アプリケーション１０１からはプリンタドライバ１０３におけるレンダリング処理を行う場合と同じ解像度で、ページの画像全体をラスタイメージに変換して出力するため、ラスタイメージデータの解像度（プリンタ１０８の解像度）の増加とともにスプールデータ１０４自体も巨大化してしまうという問題が発生している。また、レンダリング処理を行わないアプリケーションでも、前述したようにスプールデータサイズの増大によるシステムエラーが発生するという問題がある。

【００１７】

図２は、印刷対象の各用紙サイズ及びプリンタ１０８の各レンダリング解像度に対する、１ページ分のスプールファイル１０４のサイズの一例を示した図である。

【００１８】

図２は１ページ分のスプールファイル１０４の容量を示しており、ここではＡ４，Ｂ４，Ａ３判のそれぞれに対する各解像度（６００，７２０，１２００ｄｐｉ）でのスプールデータのデータ容量がメガバイトの単位で表わされている。当然、複数ページがスプールされた場合には、スプールファイル１０４の必要な容量は更に大きくなる。このようにしてスプールファイル１０４のサイズが大きくなると、印刷に要する時間が長くなるだけでなく、ハードディスクの容量不足でスプールエラーが発生したり、扱うラスタイメージデータが巨大化することから、アプリケーション１０１やスプーラ内部でメモリエラー等が発生したりして、正常に印刷されないという問題が発生する。

【００１９】

このような事態に対する対処法としては、レンダリングの解像度を下げる、即ち、ラスタイメージの解像度を下げることも考えられるが、文字やグラフィックを描画コマンドとして出力するアプリケーションの場合、これでは通常のキャラクタコードで出力されている文字列まで低い解像度でレンダリングされることになり、プリンタ１０８本来の印刷品位での印刷が不可能になる。

【００２０】

そこで本実施の形態では、レンダリング解像度、即ち、プリンタ１０８の解像度に応じた解像度でレンダリングしたラスタイメージデータを出力するアプリケーション１０１から出力されるスプールファイル１０４のサイズを小さくしながら、印刷キャラクタコードで出力される文字列などのように、解像度に依存せずに出力される描画データについては高品位でレンダリングすることを目的とする。

【００２１】

次に本実施の形態１に係る処理について図３乃至図８を参照して詳しく説明する。

【００２２】

図５は、本実施の形態１に係る印刷システムにおける処理の流れを説明するフローチャートである。ここではアプリケーション１０１が作成した描画データをスプールする処理について説明する。

【００２３】

まずアプリケーション１０１は、印刷するための描画データ（文書データや画像データ等の印刷データ）を、通常、対象とするプリンタ１０８の解像度や印刷領域（解像度に応じたピクセル数）に合わせて作成する。そしてステップＳ１０１で、アプリケーション１０１は、ＯＳのＧＤＩ１０２に対してプリンタドライバ１０３の初期化を要求し、プリンタ１０８の印刷領域や解像度などのプリンタ情報を予め定められたＡＰＩ（Application Programming Interface）を用いて要求する。次にステップＳ１０２に進み、ＧＤＩ１０２は、この要求に応じてプリンタドライバ１０３を初期化し、プリンタドライバ１０３に対して「ＧＤＩＩＮＦＯ」と呼ばれるプリンタ情報を要求する。

【００２４】

この「ＧＤＩＩＮＦＯ」は、図３に示すようなデータ構造をしている。

【００２５】

またプリンタドライバ１０３は、「ＤＥＶＭＯＤＥ」と呼ばれるユーザ設定値を基に用紙サイズや用紙内の印刷領域などを設定する。例えば、「ulLogPixelsX」３０１，「ulLogPixelsY」３０２はレンダリング解像度であり（画像の水平及び垂直方向の論理インチ当りの画素数）、このレンダリング解像度に応じて、「ulHorzRes」（画素単位での印刷幅）３０３／「ulVertRes」（画素単位での印刷高さ）３０４、「ptlPhysOffset.x」（印刷領域の左側オフセット）３０５／「ptlPhysOffset.y」（印刷領域の上側オフセット）３０６などの値を設定する。

【００２６】

図４は、ユーザ設定値「ＤＥＶＭＯＤＥ」の一例を示す図である。

【００２７】

この「ＤＥＶＭＯＤＥ」のデータ構造において、「dmDriverData」（デバイス固有の情報）４０１以外は、ＯＳで共通の仕様となっている。この「dmDriverData」４０１は、各プリンタドライバ毎に自由に使用することが可能である。

【００２８】

前述したレンダリング解像度でレンダリングしたラスタイメージデータを、アプリケーション１０１から出力するスプールファイル１０４のサイズを小さくするには、このレンダリング解像度の値を小さくすれば良い。しかし、このレンダリング解像度を小さくすると、キャラクタコードで表わされている文字列などのように、解像度に依存しない描画データについても、その印刷品位が下がってしまう。

【００２９】

ここでプリンタドライバ１０３が初期化されるケースは２種類あり、一つは、前述したアプリケーション１０１がＧＤＩ１０２に描画データを出力する際に行う初期化である。もう一つは、プリントプロセッサ１０５がスプールファイル１０４のデータをプリンタ１０８に出力する際の初期化である。

【００３０】

ＯＳの仕様に基づけば、「ulLogPixelsX」３０１，「ulLogPixelsY」３０２はレンダリング解像度を設定することになっている。ここで、従来のプリンタドライバは、両方の初期化に対して同じ解像度を返している。しかし、本実施の形態に係るプリンタドライバ１０３は、アプリケーション１０１からの初期化の場合だけスプール軽減用の低解像度の値を返し、プリントプロセッサ１０５からの初期化の時には高解像度のレンダリング解像度（プリンタ１０８の解像度に対応）を返している。これにより、アプリケーション１０１において、レンダリング解像度でレンダリングしたラスタイメージデータを出力するスプールファイル１０４のサイズを小さくしながら、キャラクタコードで出力されている文字列など、解像度に依存せずに出力される描画データについては高品位のままで印刷することができる。

【００３１】

そこでまず、プリンタドライバ１０３は、スプールサイズ軽減用の解像度を返すか、即ち、スプールする描画データの解像度を低下させる必要があるかを判定する必要がある（ステップ１０３）。

【００３２】
この判定処理について図６のフローチャートを参照して説明する。尚、図６において、図５と共通する部分は同じ記号で示している。
【００３３】
図６は、図５のステップＳ１０３における、プリンタドライバ１０３によりスプールサイズの軽減機能をオンするかどうかを判定し、それに応じた解像度を返す処理を示すフローチャートである。
【００３４】
まずステップＳ２０１で、プリンタドライバ１０３は、その初期化がスプール目的のＧＤＩ１０２を経由したアプリケーション１０１からの初期化か、実際に印刷するためのＧＤＩ１０２を経由したプリントプロセッサ１０５からの初期化かであるかを判定する。これを判定する一つの方法として、呼び元のモジュールを調べることが可能なので、呼び元のモジュールの中にプリントプロセッサ１０５を示すコードが含まれるかどうかで判定することができる。
【００３５】
また他の方法としては、前述したようにプリントプロセッサ１０５としてＯＳ標準のものを使う以外に、カスタマイズしたプリントプロセッサ１０５をプリンタドライバ１０３とともに供給することもあり得る。その場合、プリントプロセッサ１０５が、プリンタドライバ１０３を初期化する際に、「ＤＥＶＭＯＤＥ」にプリントプロセッサ１０５からの初期化であるというフラグを設定しておき、プリンタドライバ１０３がその値を参照して、その初期化がどこから指示されたかを判定することができる。
【００３６】
このステップＳ２０１で、アプリケーション１０１からの初期化であると判断した場合はステップＳ２０２に進み、プリンタドライバ１０３は、スプールサイズの軽減機能が必要かどうかを判定する。これを判定する一つの方法としては、ユーザが選択する方法がある。これは印刷中にエラーが発生して印刷されなかったり、印刷に要する時間が異常に長かったり、印刷してもページの一部又は全体の描画データが抜けて印刷された場合には、ユーザの判定により、プリンタドライバ１０３のユーザ設定機能を用いて、このスプールサイズの軽減機能をオンにする。
【００３７】
図１２は、このスプールサイズの軽減機能を設定するためのプリンタドライバ１０３によるユーザインターフェース画面例を示す図である。
【００３８】
このユーザインタフェース画面は、プリンタドライバ１０３の印刷詳細設定のプロパティを開き、カスタム設定ボタン（図示省略）を指示することによりプリンタドライバ１０３により表示される。ユーザは印刷の開始を命令する際に、アプリケーション１０１の印刷設定画面から、プリンタドライバ１０３の印刷詳細画面を呼び出して、このカスタム設定用の画面を表示させ、このスプールサイズ軽減機能の設定を行うことができる。即ち、図１２の例では「印刷データのサイズを小さくする」のチェックボックスをチェックすることにより、スプールサイズ軽減機能の設定を行う。プリンタドライバ１０３は、「ＤＥＶＭＯＤＥ」の「dmDriverData」４０１に、このスプールサイズの軽減機能を実行するか否かの情報を記憶しておき、印刷時にその情報を参照することで、スプールサイズを軽減させるかどうかを判定することができる。
【００３９】
また他の方法としては、プリンタドライバ１０３で、ページ全体をイメージで出力した場合を想定した１ページ当りのスプールファイル１０４のサイズが所定のサイズ以上かどうかに基づいて判定することも可能である。この方法では、「ＤＥＶＭＯＤＥ」の用紙のサイズを示す「dmSize」４０２や、印刷品位を示す「dmPrintQuality」（プリンタの解像度）４０３に基づいて判定される。またその他には、プリンタドライバ１０３で、ユーザがスプールサイズに基づいてスプールサイズを軽減させるかどうか判定するか、スプールサ

イズの軽減処理を常にオン又はオフにする設定機能を設け、その設定値とスプールサイズとからスプールサイズの軽減処理を実行するかどうかを判定することも可能である。

【００４０】

図１３は、図１２のユーザインタフェース画面を拡張した表示画面例を示す図であり、スプールサイズの軽減機能の設定に際して、スプールサイズに基づいて自動判定する機能を追加した例を示している。この例では、「印刷データのサイズを小さくする」のチェックボックスがチェックされている状態で「自動的に判定する」が選択できるようになる。「印刷データのサイズを小さくする」のチェックボックスがチェックされていて（スプールサイズの軽減処理が選択）、かつ「自動的に判定する」のチェックボックスがチェックされている場合に、スプールサイズに基づいて、自動的にスプール軽減処理を実行するかどうかが決定される。

【００４１】

再び図６に戻り、ステップＳ２０１又はＳ２０２で「ＮＯ」と判定した場合（スプールサイズの軽減を行わない）にはステップＳ２０４に進み、プリンタドライバ１０３は、印刷用のレンダリング解像度（プリンタ１０８の解像度）を基にＧＤＩ１０２にプリンタ情報を返す。

【００４２】

一方、上記ステップＳ２０１及びＳ２０２で「ＹＥＳ」と判定した場合（スプールサイズの軽減を行う）はステップＳ２０３に進み、プリンタドライバ１０３は、ＧＤＩ１０２経由で、スプールサイズを軽減するために、低解像度のプリンタ情報をアプリケーション１０１に返す。

【００４３】

本実施の形態では、プリンタの印刷解像度が６００ｄｐｉであり、スプールサイズ軽減機能を実行する際の解像度を３００ｄｐｉとしている。この３００ｄｐｉの場合、アプリケーションが描画データをラスタイメージで出力するものであっても、ＯＳのスプール処理に伴うスプールサイズは、描画データが多値データであっても１ＧＢ（ギガバイト）を超えることがなくなる。これにより、情報処理装置のＲＡＭ領域やＨＤＤの空き容量を加味しても、印刷処理でスプール処理エラーになることは一般にはありえないと予想できる。このようにして、前述したレンダリング解像度でレンダリングしたイメージを出力するアプリケーション１０１からのスプールファイル１０４の出力データサイズを小さくできる。しかし、レンダリング解像度でレンダリングしたイメージを出力するアプリケーション１０１の場合には、この低下させた解像度でレンダリングした結果を出力するため、解像度が低下するほどスプールサイズが小さくなるが、それに応じて、その印刷結果も劣化してしまう。そこで、用紙サイズや印刷品位などに応じて、この解像度を決定する必要がある。

【００４４】

こうしてステップＳ２０３或いはＳ２０４を実行した後ステップＳ１０６（図５）に進み、ＧＤＩ１０２に返されたプリンタ情報は、そのＧＤＩ１０２を経由してアプリケーション１０１に返される。これによりアプリケーション１０１は、この返されたプリンタ情報に基づいて、その得られた解像度や印刷領域などに基づいて描画データを作成し、ＧＤＩ１０２に出力する。

【００４５】

次にステップＳ１０７に進み、ＧＤＩ１０２は、アプリケーション１０１から要求された描画データを、Windows（登録商標）の標準スプールファイル１０４のフォーマットであるＥＭＦフォーマットで、スプールファイル１０４としてスプールする。このスプールが終了するとステップＳ１０８に進み、ＧＤＩ１０２は、プリントプロセッサ１０５に印刷の開始を要求する。

【００４６】

次にステップＳ１０９に進み、ステップＳ１０８で印刷開始が指示されたプリントプロセッサ１０５は、プリンタドライバ１０３を初期化するため、ＧＤＩ１０２に初期化を要求

する。ここで前述の図６のステップＳ２０１の判定で、「ＤＥＶＭＯＤＥ」にプリントプロセッサ１０５からの初期化であることを設定して判定する場合には、プリントプロセッサ１０５は、初期化する際に、この「ＤＥＶＭＯＤＥ」にプリントプロセッサ１０５からの初期化であることを示す情報を設定し、ＧＤＩ１０２に対し、プリンタドライバ１０３の初期化を要求するとともに、プリンタ１０８の印刷領域や解像度などのプリンタ情報を要求する。次にステップＳ１１０に進み、こうしてプリントプロセッサ１０５からプリンタドライバ１０３の初期化を要求されたＧＤＩ１０２は、プリンタドライバ１０３を初期化し、プリンタドライバ１０３に対してプリンタ情報である「ＧＤＩＩＮＦＯ」を要求する。

【００４７】

次にステップＳ１１１に進み、これに応じてプリンタドライバ１０３は、「ＧＤＩＩＮＦＯ」を設定して初期化する。この際、前述のステップＳ１０３と同じ初期化関数が呼ばれるため、ステップＳ１０３と同様に、ステップＳ２０１の判定を行う。ここでは、アプリケーション１０１からの初期化でないと判定されるので、プリンタ１０８の解像度である印刷用の高解像度でプリンタ情報を「ＧＤＩＩＮＦＯ」に設定し、このプリンタ情報をＧＤＩ１０２に返す。

【００４８】

次にステップＳ１１２に進み、プリントプロセッサ１０５は、このプリンタ情報を基に描画データをＧＤＩ１０２に出力する。この際、まずＧＤＩ１０２に対し、ＧＤＩ１０２経由で得たプリンタ情報に従った解像度への変換（座標空間の変更）を要求する。次にステップＳ１１３に進み、ＧＤＩ１０２は、プリントプロセッサ１０５が要求した解像度の変換指示に従って、スプールファイル１０４の描画データを１ページ毎に読み出してプリンタドライバ１０３に出力する。その結果、スプールされた時と異なる解像度になっても、ＧＤＩ１０２内で解像度変換されてプリンタドライバ１０３に対して印刷要求されるので、同じレイアウトで印刷することができる。

【００４９】

アプリケーション１０１からＧＤＩ１０２への出力コマンドと、ＧＤＩ１０２からプリンタドライバ１０３への出力コマンドの簡易的に記述した例を図７と図８に示す。

【００５０】

図７は、一般的な印刷による出力コマンドの一例を示す図、図８は、本実施の形態に係る印刷コマンドの一例を示す図である。

【００５１】

図７において、グラフィックイメージ７０１は、アプリケーション１０１からの出力が、プリンタドライバ１０３のレンダリング解像度に合わせて印刷されている。即ち、イメージ７０１は６００×６００がそのイメージで、（２００，２００）を始点として６００×６００画素で出力されている。７０２はキャラクタコードで出力されている文字列の出力領域を示している。ここで文字列「abcdefghijk」は、サイズ２００で位置（１０００，４００）に出力され、文字列「123456789」はサイズ２００で、位置（１０００，６００）に出力される。７０３はプリンタドライバ１０３のレンダリング解像度に依存しないオリジナルイメージの拡大出力領域である。このイメージ７０３は、１００×１００のイメージを始点（２００，１０００）から１６００×１６００のサイズで拡大して出力されている。実際には、レンダリング解像度のラスタイメージデータで出力するアプリケーションの場合、このページ全体をレンダリング解像度のラスタイメージデータに変換して印刷するが、この図７では、ラスタイメージデータではなく、描画データによって、この印刷イメージを設定している。この図７に示す通常の印刷例では、アプリケーション１０１からの出力が、そのままＧＤＩ１０２からの出力になっている。従って図７の例では、ＧＤ１１０２は、上述の７０１～７０３のサイズ及び表示位置、表示サイズをそのまま出力して描画している。即ち、図７の例では、「アプリケーションからの出力コマンド」と「ＧＤＩからの出力コマンド」とは完全に一致している。

【００５２】

図８は図７と同じ描画データによる印刷結果を示す図であるが、この例では、アプリケーション１０１からの出力時にはスプール軽減用の低解像度として、印刷用のレンダリング解像度に対して縦横それぞれ１／２の値を返しており、これは図８の「アプリケーションから出力コマンド」（図８）に示すようなコマンドで指定される。

【００５３】

図７と比較すると明らかなように、図７の７０１では、「６００×６００ピクセルの画像」がスプールされているのに対し、この図８の８０１では、「３００×３００ピクセルの画像」がスプールされているので、スプールデータのサイズが減少している。しかし、レンダリング解像度に依存しない文字列及びイメージ８０２，８０３については、その座標情報やサイズが異なるだけで、同じデータサイズでスプールされる。即ち、文字列のサイズは、図７の７０２で示すサイズ「２００」の半分「１００」となり、座標情報も（１０００，４００），（１０００，６００）のそれぞれの半分である（５００，２００）、（５００，３００）となっている。また８０３で示すイメージに関しても、始点は図７の７０３の始点（２００，１０００）の半分の（１００，５００）、サイズも図７の７０３の（１６００×１６００）の半分である（８００×８００）となっている。

【００５４】

上述の「アプリケーションから出力コマンド」（図８）では、イメージ８０１は、３００×３００画素のイメージで、（１００，１００）を始点として、サイズ３００×３００画素で出力されている。また文字列「abcdefghijk」は、サイズ１００で、位置（５００，２００）に出力され、文字列「123456789」は、サイズ１００で、位置（５００，３００）に出力される。更にイメージ８０３は、１００×１００のイメージを、始点（１００，５００）から８００×８００のサイズで拡大して出力されている。

【００５５】

また後述の「ＧＤＩ１０２からの出力コマンド」（図８）においては、グラフィック８０１に関しては、図７のグラフィック７０１と比べて、印刷されるイメージのサイズ（解像度）が１／２×１／２で（３００×３００）となっていて小さくなっているため、印刷された画像品位が低下している。しかし、プリントプロセッサ１０５の要請によるＧＤＩ１０２の解像度変換の結果、６００×６００のサイズで描画され、かつ文字列８０２，イメージ８０３に関しては、図７の７０２，７０３と全く同じコマンドで出力されるため、これら文字列８０２及びイメージ８０３は高いレンダリング解像度でレンダリングされ、図７の文字列７０２，イメージ７０３と同じ画像品位で印刷できることになる（ステップＳ１１３）。

【００５６】

即ち、上述の「ＧＤＩ１０２からの出力コマンド」（図８）では、イメージ８０１は、３００×３００画素のイメージで、（２００，２００）を始点として、サイズ６００×６００画素で描画されている。また文字列「abcdefghijk」は、サイズ２００で、位置（１０００，４００）に描画され、文字列「123456789」は、サイズ２００で、位置（１０００，６００）に描画される。更にイメージ８０３は、１００×１００のイメージを、始点（２００，１０００）から１６００×１６００のサイズで拡大して描画されている。

【００５７】

次いでステップＳ１１４に進み、プリンタドライバ１０３は、上記スプールファイル１０４を、ＧＤＩレンダリングエンジン１０７を用いてラスタイメージデータにレンダリングする。ここで、前述したように、ＧＤＩレンダリングエンジン１０７はＧＤＩ１０２と同じモジュールである。そしてステップＳ１１５に進み、プリンタドライバ１０３は、そのレンダリングされたラスタイメージデータをプリンタコマンドの印刷データに変換してプリンタ１０８に出力する。そして、この処理を終了する。

【００５８】

以上説明したように本実施の形態１によれば、印刷された画像の劣化を抑えて、アプリケーションにより印刷が指示されたデータをスプールするデータ量を削減できるという効果がある。

【００５９】
［実施の形態２］
上記実施の形態１では、低解像度のレンダリング解像度用にスプールされたＥＭＦフォーマットのスプールファイル１０４を、プリンタドライバ１０３で高解像度にレンダリングする方法について述べた。しかし、アプリケーション１０１の判断で、ＥＭＦフォーマットでのスプール自体を止めるように指示して印刷することがある。例えば、プリンタドライバ１０３がプリンタ１０８に送信する印刷コマンドデータそのものをスプールするＲＡＷスプールなどに変更する場合である。その場合、前述の実施の形態１では、「アプリケーションからの出力コマンド」（図８）がそのまま、ＧＤＩ１０２経由でプリンタドライバ１０３に出力されるために、縮小されて印刷されてしまう。
【００６０】
そこで、以下ではこれを防止する方法について説明する。
【００６１】
図９は、本発明の実施の形態２に係る印刷システムのＲＡＷスプールの構成を示すブロック図で、前述の図１と共通する部分は同じ記号で示し、その説明を省略する。
【００６２】
前述の実施の形態１では、アプリケーション１０１が印刷要求したデータはＧＤＩ１０２で一旦ＥＭＦデータとしてスプールファイル１０４にスプールされていた。これに対して本実施の形態２に係るＲＡＷスプールの場合には、ＧＤＩ１０２は、直接プリンタドライバ１０３に印刷を要求する。こうして印刷が要求されたプリンタドライバ１０３は、ＧＤＩレンダリングエンジン１０７を用いて、その描画データをラスタイメージデータにレンダリングした後、色処理やプリンタコマンドの付加などを行って出力する。そして、そのプリンタコマンドがスプールファイル１０４としてスプールされる。このスプールされたプリンタコマンドは、プリントプロセッサ１０５及びポートモニタ１０６を経由してプリンタ１０８に送信される。
【００６３】
本実施の形態２における処理について、前述の実施の形態１との違いを中心に図１０Ａ，１０Ｂのフローチャートを参照して説明する。尚、図１０Ａ，１０Ｂにおいて図５のフローチャートと共通する部分は同じ記号で示しており、それらの説明を省略する。
【００６４】
ステップＳ１０６（図１０Ａ）で、アプリケーション１０１が描画データを作成してＧＤＩ１０２に出力する際に、アプリケーション１０１はＥＭＦスプールを禁止することが可能である。
【００６５】
図１１は、アプリケーション１０１が初期化するときに指定する「ＤＯＣＩＮＦＯ」のデータ構造の一部を示した図である。
【００６６】
この「ＤＯＣＩＮＦＯ」の「lpszDatatype」１１０１で、スプールファイル１０４のデータタイプを指定することができる。通常は、０（ＮＵＬＬと呼ばれる）を指定し、システムの設定をそのままで印刷するが、文字列が「ｅｍｆ」であればＥＭＦスプールを示し、「ｒａｗ」であればＲＡＷスプールへの切り替えを要求している。
【００６７】
こうしてステップＳ３０１で、ＧＤＩ１０２は、「ＤＯＣＩＮＦＯ」の「lpszDatatype」１１０１に基づいて、アプリケーション１０１からＥＭＦスプール以外が指定されたかを判定する。ここで「ＮＯ」、即ち、ＥＭＦスプールが指定されると前述のステップＳ１０７乃至ステップＳ１１３（図１０Ａ，１０Ｂ）までの処理を行う。
【００６８】
一方ステップＳ３０１（図１０Ａ）で「ＹＥＳ」、即ち、ＥＭＦスプール以外が指定されるとステップＳ３０２（図１０Ｂ）に進み、ＧＤＩ１０２はアプリケーション１０１からの描画データをプリンタドライバ１０３に直接出力する。

【００６９】
こうしてステップＳ１０７乃至Ｓ１１３（図１０Ａ，１０Ｂ）を実行するか、或いはステップＳ３０２の処理を実行するとステップＳ１１４（図１０Ｂ）に進み、ＧＤＩ１０２経由でプリンタドライバ１０３に印刷を要求する。しかし、ステップＳ３０１で「ＹＥＳ」と判定した場合には、スプール軽減用の低解像度でレンダリングされてしまう。
【００７０】
そこでステップＳ３０３以降の処理では、低解像度でレンダリングされた場合だけ、プリンタ１０８に出力する前にスプール軽減用の低解像度から、印刷用の高解像度に拡大する。そのためステップＳ３０３で、レンダリングされたイメージのサイズなどからスプールサイズ軽減用の低解像度でレンダリングしたかどうかを判定する。このステップＳ３０３で「ＮＯ」と判定された場合にはステップＳ１１５（図１０Ｂ）に進み、印刷用の高解像度でレンダリングされているので、プリンタドライバ１０３は、そのラスタイメージデータをそのままプリンタ１０８に出力する。
【００７１】
一方、ステップＳ３０３で「ＹＥＳ」と判定した場合はステップＳ３０４に進み、この場合にはスプールサイズ軽減用の低解像度でレンダリングされているので、低解像度でラスタライズされているラスタイメージデータを、ＧＤＩレンダリングエンジン１０７が持つ拡大機能用いて印刷用の解像度に拡大する。そして最後にステップＳ３０５で、その拡大したラスタイメージデータをプリンタコマンドに変換してプリンタ１０８に出力する。そして、本処理を終了する。
【００７２】
以上説明したように本実施の形態２によれば、プリンタ１０８の解像度に応じたレンダリング解像度でレンダリングしたラスタイメージデータを出力するアプリケーション１０１からの出力スプールファイルのサイズを小さくしながら、キャラクタコードで記述されている文字列などのように、解像度に依存せずに出力される描画データについては高品位でレンダリングできる。
【００７３】
［実施の形態３］
プリンタの高解像度化に伴い縮小しても視認性が損なわれにくいため、複数のページを縮小して一枚の用紙に印刷するというＮ－ｕｐ印刷（縮小配置印刷）がよく使われるようになってきた。Ｎ－ｕｐのＮは、１ページ当たりの印刷ページ数を示し、例えば１枚の用紙に４ページ分の描画データを印刷する場合は４－ｕｐと呼ばれている。
【００７４】
図１４は、４ページ分の描画データを１枚の用紙に２×２で印刷した４－ｕｐ印刷の一例を示す図である。
【００７５】
このようにＮ－ｕｐ印刷においては、各ページの描画データを縮小して印刷するため、図表等のアプリケーション１０１でよく用いられる細線などは、縮小された場合に線幅が「０」になってしまい、線として印刷されない場合がある。
【００７６】
図１５は、図１４の印刷データを印刷した場合に、ページ１（Page1）の表の一部の細線が印刷されなかった例を示す図である。
【００７７】
図１５の例では、１ページ（Page1）目の描画データにおいて細線の一部が印刷されていない。そこで本実施の形態３では、Ｎ－ｕｐ印刷時における細線の欠落を防止する技術について説明する。尚、この実施の形態３における印刷システムのハードウェア構成は前述の実施の形態１の場合と同様であるため、その説明を省略する。
【００７８】
前述の実施の形態１で説明したように、アプリケーション１０１が印刷要求した描画データ（文書データ、画像データ等の印刷データ）は、通常、ＧＤＩ１０２が一旦、ＥＭＦデ

ータとしてスプールファイル１０４にスプールし、プリントプロセッサ１０５が、スプールファイル１０４から印刷出力情報と描画データをページ単位で読み込み、１ページずつＧＤＩ１０２を介してプリンタドライバ１０３に出力して印刷している。

【００７９】

このＧＤＩ１０２には、このスプールされたページ毎に用紙の一部の領域を指定し、その領域に１ページ分のデータを縮小して出力する機能があり、ＧＤＩ１０２に対し用紙への出力終了命令を指定するまでは同じ用紙内に何ページでも出力できる。プリントプロセッサ１０５はこの機能を用いてＮ－ｕｐ印刷を行っている。

【００８０】

つまり、プリントプロセッサ１０５は、プリンタドライバ１０３に出力する際に、プリンタドライバ１０３が返した印刷領域をＮ等分し、スプールファイル１０４から印刷出力情報と描画データをページ単位で読み込み、１ページずつＧＤＩ１０２を介してＮ等分した領域に順番に出力している。

【００８１】

本実施の形態３では、プリンタドライバ１０３は、アプリケーション１０１に対しては、縮小されることを考慮して解像度に縮小率を掛けた解像度に基づくプリンタ情報を返す。一方、印刷時には、プリンタドライバ１０３は、印刷用の高解像のプリンタ情報を返すことで、印刷時に縮小された線幅は「１」のままで印刷できる。

【００８２】

以下、本実施の形態３に係る処理について説明する。この実施の形態３に係るハードウェア構成は前述の実施の形態１の印刷システムと同じハードウェア構成であるため、その説明を省略する。

【００８３】

図１６は、本発明の実施の形態３に係る印刷システムにおける処理の流れを説明するフローチャートで、前述の実施の形態１のフローチャート（図５）と共通する処理ステップは同じ記号で示し、それらステップの説明を省略する。ここではアプリケーション１０１が作成した描画データをスプールする処理について説明する。

【００８４】

ステップＳ１０２で、ＧＤＩ１０２はプリンタドライバ１０３を初期化し、「ＧＤＩＩＮＦＯ」と呼ばれるプリンタ情報を要求する。次にステップＳ４０２に進み、プリンタ情報に基づいて、Ｎ－ｕｐ用の細線欠け対策モードがオンであると判定されるとステップＳ４０３に進み、アプリケーション１０１からの初期化の場合だけ、プリンタドライバ１０３は、Ｎ－ｕｐ用の細線欠けの対策用に低解像度の値を返す。Ｎ－ｕｐ用の細線欠け対策モードがオフのとき、或いはプリントプロセッサ１０５からの初期化の時は、ステップＳ２０４で、高解像度のレンダリング解像度（プリンタ１０８の解像度に対応）を返す。

【００８５】

図１７は、図１６のステップＳ３０３における、プリンタドライバ１０３によりＮ－ｕｐ用の細線欠け対策するかどうかを判定し、それに応じた解像度を返す処理を示すフローチャートである。

【００８６】

図１７において、ステップＳ２０１で、アプリケーション１０１からの初期化であると判断した場合はステップＳ４０２に進み、Ｎ－ｕｐ用の細線欠けの対策機能がオンかどうかを判定する。これはユーザが、図１８のユーザインターフェースを用いて、この機能をオンに設定したかどうかにより判定する。Ｎ－ｕｐで印刷した結果、細線の一部が抜けて印刷された場合には、ユーザの判定により、プリンタドライバ１０３のユーザ設定機能を用いて、このＮ－ｕｐ用の細線欠け対策機能をオンにする。

【００８７】

図１８は、Ｎ－ｕｐ用の細線欠け対策機能を設定するためのプリンタドライバ１０３のユーザインターフェースの画面例を示す図である。ここで「Ｎ－ｕｐ時の細線欠けを防ぐ」のチェックボックスをチェックすることにより、Ｎ－ｕｐ用の細線欠けの対策機能をオン

に設定できる。
【００８８】
ユーザは印刷開始する際に、アプリケーション１０１の印刷設定画面から、プリンタドライバ１０３の設定を行うことができる。プリンタドライバ１０３は、「ＤＥＶＭＯＤＥ」の「dmDriverData」４０１（図４）に、このＮ－ｕｐ用の細線欠け対策機能を実行するか否かの情報を記憶し、その情報を、図１７のステップＳ４０２で参照することで判定することができる。
【００８９】
図１７のステップＳ２０１又はＳ４０２で「ＮＯ」（Ｎ－ｕｐ用の細線欠け対策を行わない）と判定した場合はステップＳ２０４に進み、プリンタドライバ１０３は、印刷用のレンダリング解像度（プリンタ１０８の解像度）を基にＧＤＩ１０２にプリンタ情報を返す。
【００９０】
一方、上記ステップＳ２０１及びＳ４０２で「ＹＥＳ」（Ｎ－ｕｐ用の細線欠け対策を行う）と判定した場合はステップＳ４０３に進み、プリンタドライバ１０３は、ＧＤＩ１０２経由で、Ｎ－ｕｐ用の細線欠け対策機能用の低解像度のプリンタ情報をアプリケーション１０１に返す。
【００９１】
この場合の解像度の決定は、Ｎ－ｕｐの縮小率から決定できる。縦方向にNvページ分、横方向にNhページ分の描画データを一枚の用紙に印刷（Nv×Nh－ｕｐ）する場合、縮小率の逆数Ｓは、Nh≧Nvの場合はＳ＝Nh、Nh＜Nvの場合はＳ＝Nvとなる。
【００９２】
しかし、通常、Ｎ－ｕｐでは見やすさを向上するため、各ページ間や、各ページの周りに空白を入れることが多い。
【００９３】
図１９は、Ｎ－ｕｐ印刷において、各ページの印刷画像間に空白部分を挿入した例を説明する図である。図１９の斜線分が空白部分を示している。この空白部をどのようにとるかは色々な方法があるが、水平方向、垂直方向に等間隔であることが望ましいので、等間隔である場合を例にして説明する。
【００９４】
いま用紙の幅をW、用紙高さをHとする。水平方向の各空白長さＩh、垂直方向の各空白高さＩvは、拡大／縮小率が決定した後でないと決まらないので、まず各空白長さの最低値Ｉを決め、これに基づいて縮小率を求めて決定する。
【００９５】
縮小後の各ページ分の画像幅は、［｛W－Ｉ×（Nh＋１）｝÷Nh］で求めることができるので、水平方向の縮小率の逆数Ｓhは、
Ｓh＝W÷［｛W－Ｉ×（Nh＋１）｝÷Nh］
で求まる。同様に、垂直方向の縮小率の逆数Ｓvは、
Ｓv＝H÷［｛H－Ｉ×（Nv＋１）｝÷Nv］
で求まり、Ｓは次の様にして決定される。即ち、Ｓh≧Ｓvの場合はＳ＝Ｓh、Ｓh＜Ｓvの場合はＳ＝Ｓvとなる。
【００９６】
なお、この計算の結果、Ｓは整数にならない場合がある。
【００９７】
本実施の形態３においても、前述の実施の形態２と同様の処理を行うことが可能である。その場合にはレンダリング後に拡大することになるが、その拡大時に非整数倍で拡大すると処理時間がかかる等の問題があるため、少数以下を切り上げて整数化することも可能である。
【００９８】
印刷用のレンダリング解像度をＲとすると、Ｎ－ｕｐ印刷用の細線欠け対策用の解像度は

Ｒ／Ｓ（Ｓ＜１）となる。これにより、１ピクセル幅等で印刷するように要求された細線であっても、縮小後でも線幅が「０」にならない線としてスプールされる。

【００９９】

次に図１６のステップＳ１１０以降の処理について説明する。

【０１００】

ステップＳ１１０で、プリントプロセッサ１０５からプリンタドライバ１０３の初期化を要求されたＧＤＩ１０２は、プリンタドライバ１０３を初期化し、プリンタドライバ１０３に対してプリンタ情報である「ＧＤＩＩＮＦＯ」を要求する。これに応じてプリンタドライバ１０３は、「ＧＤＩＩＮＦＯ」を設定して初期化する。この際、前述のステップＳ３０３の場合と同じ初期化関数が呼ばれるため、ステップＳ３０３と同様に、ステップＳ２０１（図１７）の判定を行う。ここでは、アプリケーション１０１からの初期化でないと判定されるので、プリンタ１０８の解像度である印刷用の高解像度を用いてプリンタ情報を「ＧＤＩＩＮＦＯ」に設定し、このプリンタ情報をＧＤＩ１０２に返す。

【０１０１】

次にステップＳ３１２に進み、プリントプロセッサ１０５は、このプリンタ情報を基に、描画データをＧＤＩ１０２に出力する。Ｎ－ｕｐ印刷の際には、スプールファイル１０４の描画データから１ページ毎に読み出してプリンタドライバ１０３に出力するという処理をＮ回行う。この際、プリントプロセッサ１０５は、ページ毎にＧＤＩ１０２経由で得たプリンタ情報に従い、ＧＤＩ１０２に対し、印刷位置の指定及び、解像度変換（座標空間の変更）を要求する。そしてＮページ分の印刷が終了すると、用紙への書き込みが終了したことをＧＤＩ１０２に通知する。

【０１０２】

次にステップＳ１１３に進み、ＧＤＩ１０２は、プリントプロセッサ１０５が要求した各ページの印刷データの印刷位置、解像度の変換指示に従って、Ｎページ分のスプールファイル１０４の描画データを１枚のページデータとしてプリンタドライバ１０３に出力する。次にステップＳ１１４に進み、プリンタドライバ１０３は、上記スプールファイル１０４を、ＧＤＩレンダリングエンジン１０７を用いてラスタイメージデータにレンダリングする。

【０１０３】

このとき、Ｎ－ｕｐ印刷用の細線欠け対策がオフの場合には、アプリケーション１０１がレンダリング解像度６００ｄｐｉのプリンタ１０８に対し、１ピクセル幅の細線を１インチ描画しようとした場合、ステップＳ２０４で６００ｄｐｉのプリンタ情報が返されるために、（０，０）～（１，６００）の領域を塗りつぶすという描画命令を行う。しかし、例えば、４－ｕｐ印刷の場合には、ステップＳ３１２の座標変換で縦横１／２の領域に縮小して印刷される。このため通常、１画素未満の値は切り捨てるため、（０，０）～（０，３００）、つまり幅「０」の領域を塗りつぶすと変換されてしまう。これによりＧＤＩ１０２は、プリンタドライバ１０３に描画命令を出力することなく、次の描画命令の処理を行い、細線が印刷されなくなってしまう。但し、（１，０）～（２，６００）の領域を塗りつぶすという描画命令は、１／２の領域に縮小しても（０，０）～（１，３００）の領域を描画すると変換されるため、１ピクセルの幅で描画されることになる。つまり、位置によって印刷されたりされなかったりする。

【０１０４】

一方、Ｎ－ｕｐ印刷用の細線欠け対策がオンの場合には、１ピクセル幅の細線を１インチ描画しようとした場合、ステップＳ４０３で、４－ｕｐではプリンタドライバ１０３が解像度３００ｄｐｉのプリンタ情報を返すので、アプリケーション１０１は、（０，０）～（１，３００）の領域を塗りつぶすという描画命令を実行する。しかし、実際に印刷するときには、ステップＳ２０４で、プリンタドライバ１０３が解像度６００ｄｐｉのプリンタ情報を返す。これにより、前述の実施の形態１で説明したように、プリントプロセッサ１０５がスプールされている描画データをプリンタドライバ１０３の解像度に合わせて座標情報を変換して出力する機能が働き、（０，０）～（２，６００）を塗りつぶすという

描画命令に変換される。その結果、４－ｕｐ印刷で１／２に縮小して印刷しても、（０，０）～（１，３００）の描画命令として、１ピクセル幅の細線が０．５インチ描画されることになる。

【０１０５】

そしてステップＳ１１５に進み、そのレンダリングしたラスタイメージデータをプリンタコマンドに変換してプリンタ１０８に出力する。そして、この処理を終了する。

【０１０６】

以上説明したように本発明の実施の形態３によれば、Ｎ－ｕｐ印刷で各ページの印刷データを縮小して印刷しても、図表などのアプリケーション１０１で使われる細い細線などは縮小されて幅「０」になってしまう、即ち、細線が印刷されないという問題を回避できる。

【０１０７】

（その他の実施の形態）

本発明の目的は前述したように、実施形態の機能を実現するソフトウェアのプログラムコードを記録した記憶媒体をシステムあるいは装置に提供し、そのシステムあるいは装置のコンピュータ（またはＣＰＵやＭＰＵ）が記憶媒体に格納されたプログラムコードを読み出し実行することによっても達成される。この場合、記憶媒体から読み出されたプログラムコード自体が前述した実施形態の機能を実現することになり、そのプログラムコードを記憶した記憶媒体は本発明を構成することになる。このようなプログラムコードを供給するための記憶媒体としては、例えば、フロッピィディスク、ハードディスク、光ディスク、光磁気ディスク、ＣＤ－ＲＯＭ，ＣＤ－Ｒ、磁気テープ、不揮発性のメモリカード、ＲＯＭなどを用いることができる。

【０１０８】

また、コンピュータが読み出したプログラムコードを実行することにより、前述した実施の形態の機能が実現されるだけでなく、そのプログラムコードの指示に基づき、コンピュータ上で稼動しているＯＳ（オペレーティングシステム）などが実際の処理の一部または全部を行い、その処理によって前述した実施の形態の機能が実現される場合も含まれている。

【０１０９】

更に、記憶媒体から読み出されたプログラムコードが、コンピュータに挿入された機能拡張ボードやコンピュータに接続された機能拡張ユニットに備わるメモリに書きこまれた後、そのプログラムコードの指示に基づき、その機能拡張ボードや機能拡張ユニットに備わるＣＰＵなどが実際の処理の一部または全部を行い、その処理によって前述した実施の形態の機能が実現される場合も含む。

【０１１０】

【発明の効果】

以上説明したように本発明によれば、画像形成デバイスにラスタイメージデータを出力するためのスプールファイルのサイズを小さくし、かつ高品位の画像を形成できるという効果がある。

【図面の簡単な説明】

【図１】　本発明の実施の形態１に係る印刷システムの構成を示すブロック図である。

【図２】　各用紙サイズ及び各レンダリング解像度に対する１ページ当りのスプールファイルのサイズの一例を示す図である。

【図３】　ＧＤＩＩＮＦＯのデータ構造の一部を示す図である。

【図４】　ＤＥＶＭＯＤＥのデータ構造の一部を示す図である。

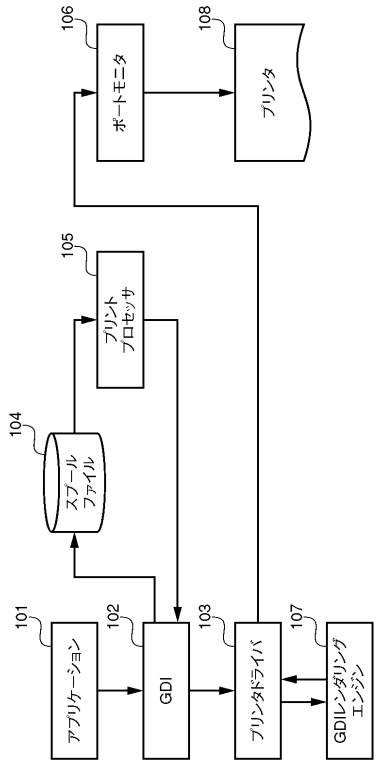
【図５】　本発明の実施の形態１に係る印刷システムにおける処理の流れの一例を示すフローチャートである。

【図６】　図７のステップＳ１０３における判定及びその判定に基づく処理を説明するフローチャートである。

【図７】　通常の印刷におけるアプリケーションからの出力コマンドとＧＤＩからの出力

コマンドの例を簡易的に記述した図である。
【図８】　本実施の形態に係る印刷におけるアプリケーションからの出力コマンドとＧＤＩからの出力コマンドの例を簡易的に記述した図である。
【図９】　本発明の実施の形態２に係る印刷システムの構成を示すブロック図である。
【図１０Ａ】　本発明の実施の形態２に係る印刷システムにおける処理の流れの一例を示すフローチャートである。
【図１０Ｂ】　本発明の実施の形態２に係る印刷システムにおける処理の流れの一例を示すフローチャートである。
【図１１】　ＤＯＣＩＮＦＯデータ構造の一部を示す図である。
【図１２】　スプールサイズの軽減機能を設定するためのプリンタドライバによるユーザインターフェース画面例を示す図である。
【図１３】　スプールサイズから判定する機能を追加したスプールサイズの軽減機能を設定するためのプリンタドライバによるユーザインターフェース画面例を示す図である。
【図１４】　４ページ分のデータを１枚の用紙に２×２で印刷した４－ｕｐ印刷例を示す図である。
【図１５】　Ｎ－ｕｐ印刷で細線の一部が印刷されなかった例を示す図である。
【図１６】　本発明の実施の形態３に係る印刷システムにおける処理の流れを説明するフローチャートである。
【図１７】　図１６のステップＳ３０３における判定及びその判定に基づく処理を説明するフローチャートである。
【図１８】　Ｎ－ｕｐ用の細線欠け対策機能を設定するためのプリンタドライバによるユーザインターフェース画面例を示す図である。
【図１９】　Ｎ－ｕｐ印刷で、各ページの印刷データ間に空白を挿入した様子を説明する図である。

【図１】



【図２】

	600dpi	720dpi	1200dpi
A4	93.8	135.0	187.6
B4	142.2	204.8	284.4
A3	190.9	274.2	381.7

(Mega Bytes / Page)

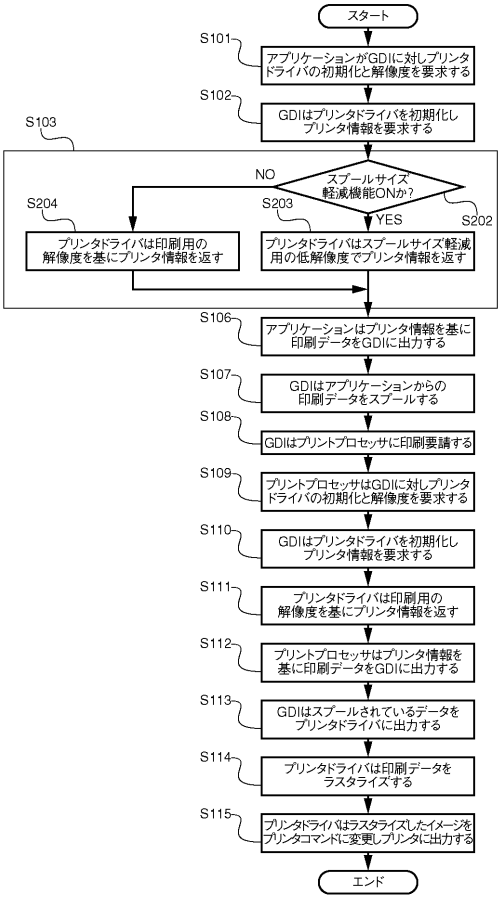
【図３】

ulVersion	デバイスドライバのバージョン。
：	：
ulHorzSize	ミリメートル（mm）単位の画面の物理的な幅。
ulVertSize	ミリメートル（mm）単位の画面の物理的な高さ。
ulHorzRes	ピクセル単位の印刷領域の幅。 303
ulVertRes	ピクセル単位の印刷領域の高さ。 304
：	：
ulLogPixelsX	画面の水平方向での、論理インチ当たりのピクセル数。 301
ulLogPixelsY	画面の垂直方向での、論理インチ当たりのピクセル数。 302
⋮	⋮
ptlPhysOffset.x	物理的なページの左端から印刷可能領域の左端までの距離をデバイス単位で表す。 305
ptlPhysOffset.y	物理的なページの上端から印刷可能領域の上端までの距離をデバイス単位で表す。 306
szlPhysSize.x	ページの物理的な幅をデバイス単位で表す。
szlPhysSize.y	ページの物理的な高さをデバイス単位で表す。
⋮	⋮

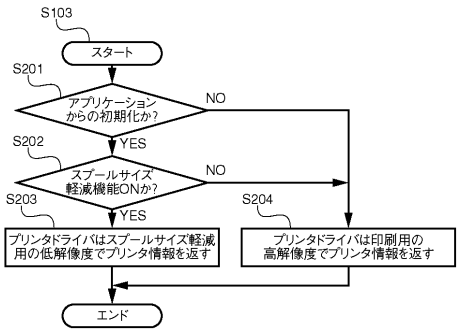
【図４】

dmDeviceName	デバイス名
dmSpecVersion	初期化データ仕様のバージョン番号
dmDriverVersion	ドライバのバージョン番号
dmSize	dmDriverDataメンバを除いたDEVMODEデータのバイト長 402
dmDriverExtra	このデータに続くプライベートなドライバデータのバイト長
dmFields	メンバで初期化されているメンバを指定
dmOrientation	用紙の方向
dmPaperSize	印刷する用紙のサイズ
dmPaperLength	用紙の長さ（10分の1mm単位）
dmPapeWidth	用紙幅（10分の1mm単位）
：	：
dmPrintQuality	プリンタの解像度 403
dmColor	カラープリンタでカラーとモノクロを設定
⋮	⋮
dmDriverData	デバイス固有の情報 401

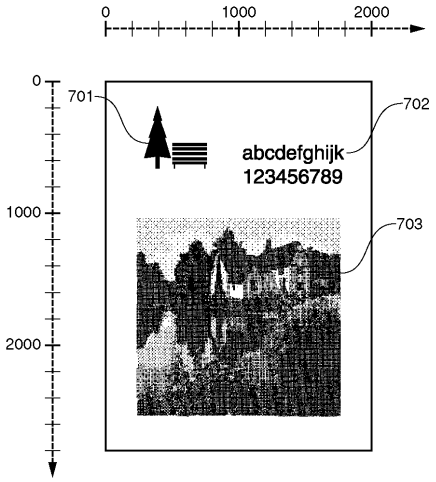
【図５】



【図６】



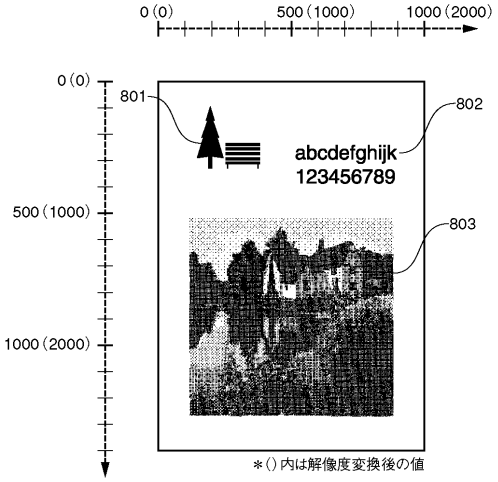
【図７】



アプリケーションからの出力コマンド
701: 600×600のイメージを(200,200)を始点として600×600のサイズで描画
702: abcdefghijk(サイズ200)の文字列を(1000,400)の位置に描画
123456789(サイズ200)の文字列を(1000,600)の位置に描画
703: 100×100のイメージを(200,1000)を始点として1600×1600のサイズで描画

GDIからの出力コマンド
701: 600×600のイメージを(200,200)を始点として600×600のサイズで描画
702: abcdefghijk(サイズ200)の文字列を(1000,400)の位置に描画
123456789(サイズ200)の文字列を(1000,600)の位置に描画
703: 100×100のイメージを(200,1000)を始点として1600×1600のサイズで描画

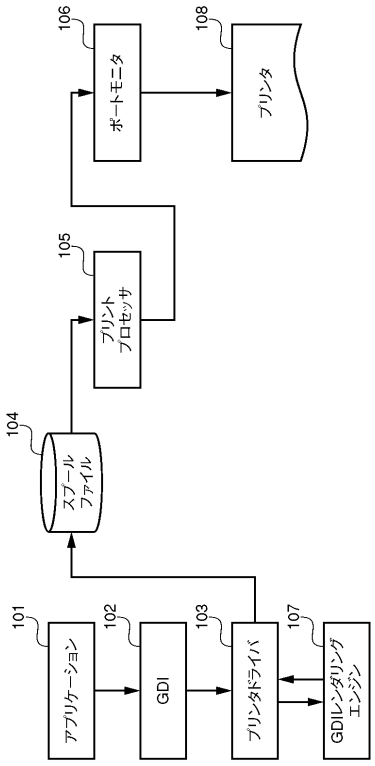
【図８】



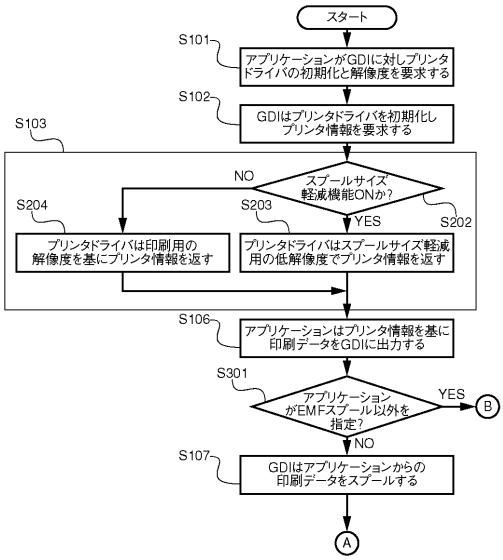
アプリケーションから出力コマンド
801: 300×300のイメージを(100,100)を始点として300×300のサイズで描画
802: abcdefghijk(サイズ100)の文字列を(500,200)の位置に描画
123456789(サイズ100)の文字列を(500,300)の位置に描画
803: 100×100のイメージを(100,500)を始点として800×800のサイズで描画

GDIからの出力コマンド
801: 300×300のイメージを(200,200)を始点として600×600のサイズで描画
802: abcdefghijk(サイズ200)の文字列を(1000,400)の位置に描画
123456789(サイズ200)の文字列を(1000,600)の位置に描画
803: 100×100のイメージを(200,1000)を始点として1600×1600のサイズで描画

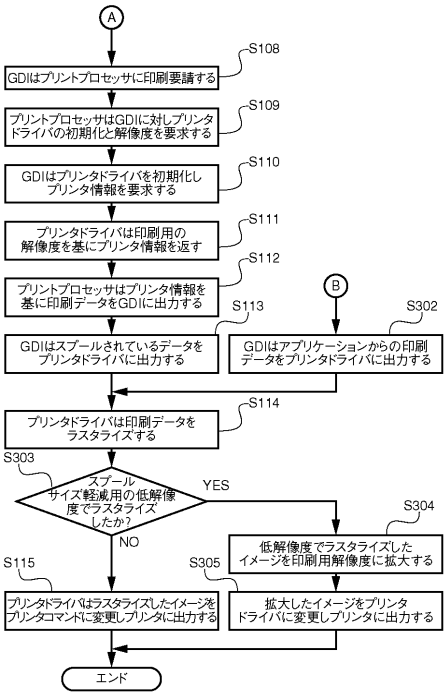
【図９】



【図１０Ａ】



【図１０Ｂ】

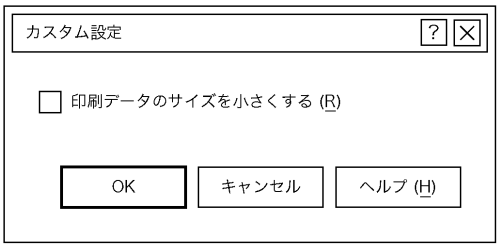


【図１１】

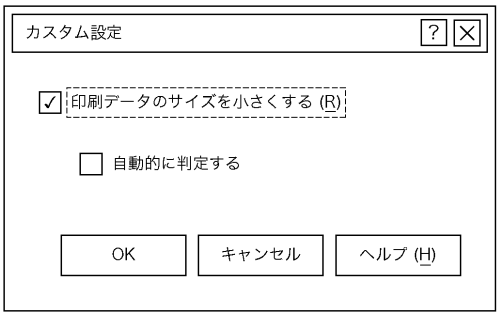
cbSize	DOCINFOのバイト長
lpszDocName	ドキュメントファイルの名前
lpszOutput	出力メディア（ファイルや出力ポート）のファイル名またはデバイス名
lpszDatatype	スプールファイルのデータタイプ
:	:

1101

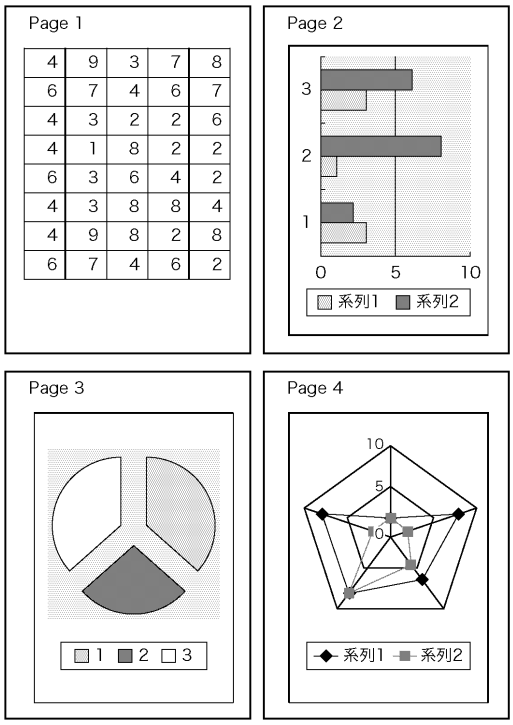
【図１２】



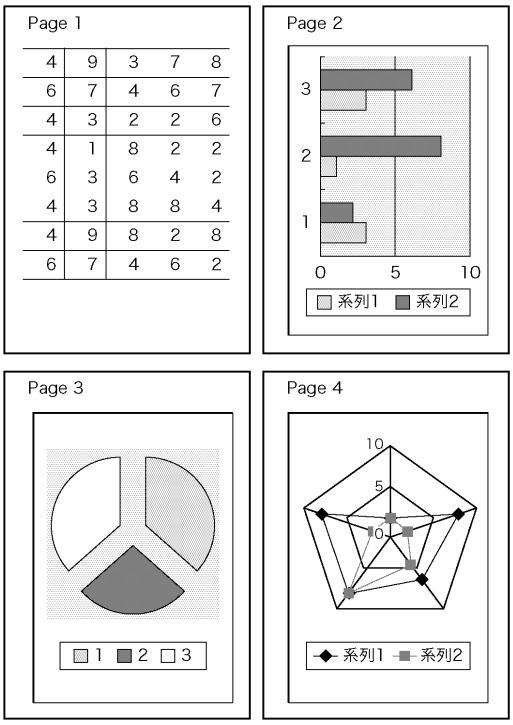
【図１３】



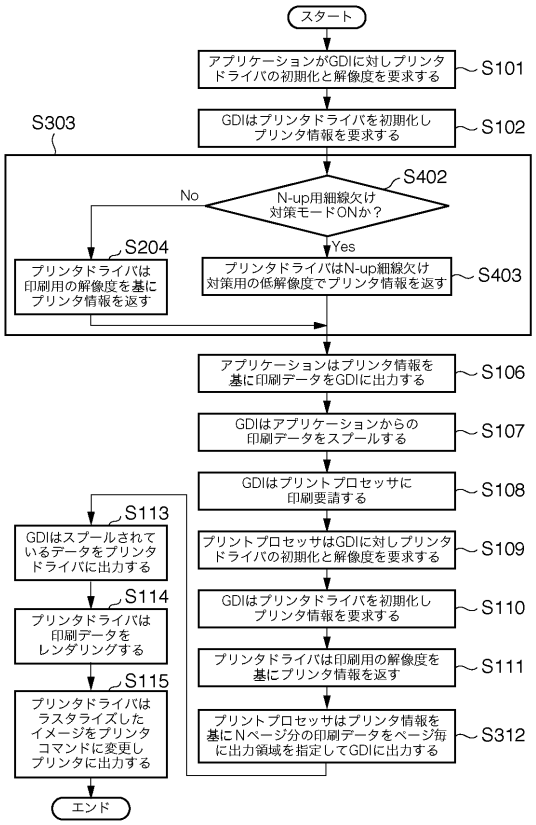
【図１４】



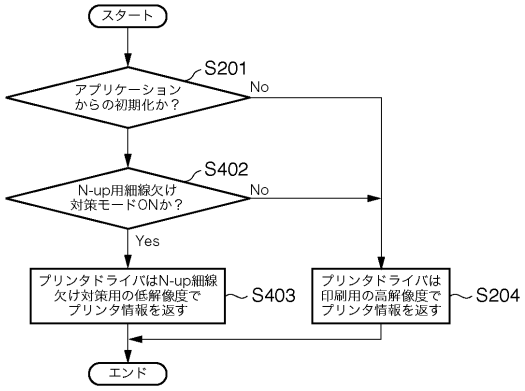
【図１５】



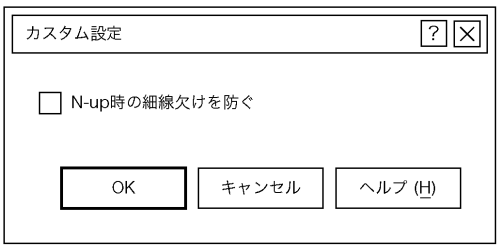
【図１６】



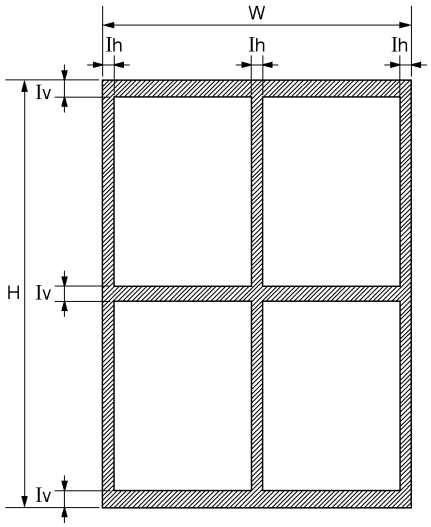
【図１７】



【図１８】



【図１９】



フロントページの続き

審査官　緑川　隆

(56)参考文献　特開２００１－１９５２１３（ＪＰ，Ａ）
特開２００１－１９７３００（ＪＰ，Ａ）
特開２０００－２４２４４５（ＪＰ，Ａ）
特開２００１－１６９０８７（ＪＰ，Ａ）
特開平１１－３２７８４９（ＪＰ，Ａ）
特開２００１－３４４０７９（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
G06F　3/12
B41J　21/00
G06T　3/40
H04N　1/387