(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第6437076号
(P6437076)
(45)発行日　平成30年12月12日(2018.12.12)　(24)登録日　平成30年11月22日(2018.11.22)

(51)Int.Cl.　F I
G06F 3/12 (2006.01)　G06F 3/12 345
G06F 3/12 308
G06F 3/12 324
G06F 3/12 344
G06F 3/12 346

請求項の数 11　(全 16 頁)

(21)出願番号　特願2017-207215(P2017-207215)
(22)出願日　平成29年10月26日(2017.10.26)
(62)分割の表示　特願2015-209846(P2015-209846)の分割
原出願日　平成27年10月26日(2015.10.26)
(65)公開番号　特開2018-22527(P2018-22527A)
(43)公開日　平成30年2月8日(2018.2.8)
審査請求日　平成29年10月26日(2017.10.26)

(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　110001243
特許業務法人　谷・阿部特許事務所
(72)発明者　母里　南陽
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　征矢　崇

最終頁に続く

(54)【発明の名称】情報処理装置及びその制御方法とプログラム

(57)【特許請求の範囲】
【請求項１】
入力されたＸＰＳ形式の描画データから、印刷装置に対応した印刷データを生成するプリンタドライバを備えた情報処理装置であって、
前記プリンタドライバは、
前記入力されたＸＰＳ形式の描画データが、ＧＤＩアプリケーションの印刷処理に基づき生成されたＧＤＩ形式の描画データから変換モジュールで変換されたＸＰＳ形式の描画データであるかどうかを判定し、
当該入力されたＸＰＳ形式の描画データが、所定のグラフィック描画命令を含むかどうかを判定し、
当該判定の結果、前記入力されたＸＰＳ形式の描画データが、前記ＧＤＩアプリケーションの印刷処理に基づき生成されたＧＤＩ形式の描画データから前記変換モジュールで変換されたＸＰＳ形式の描画データであり、且つ、前記所定のグラフィック描画命令を含む、と判定した場合、当該所定のグラフィック描画命令を前記印刷装置で表現可能な特定の描画を指定するグラフィック描画命令に変更し、
当該変更後のグラフィック描画命令を含む描画データに基づいて前記印刷データを生成する、
ことを特徴とする情報処理装置。
【請求項２】
前記プリンタドライバは、予め設定された探索条件の情報に従って、前記入力されたＸ

ＰＳ形式の描画データから特定の文字列を探索することで、前記ＸＰＳ形式の描画データが前記ＧＤＩアプリケーションからの印刷処理に基づき生成されたＧＤＩ形式の描画データから前記変換モジュールで変換されたＸＰＳ形式の描画データであるかどうかを判定す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】

前記プリンタドライバは、前記プリンタドライバのユーザインタフェース画面でユーザが行なった出力方式に関する設定内容に基づいて、前記入力されたＸＰＳ形式の描画データが、前記ＧＤＩアプリケーションからの印刷処理に基づき生成されたＧＤＩ形式の描画データから前記変換モジュールで変換されたＸＰＳ形式の描画データであるかどうかを判定す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４】

前記所定のグラフィック描画命令とは、オブジェクト幅が０のグラフィック描画命令であ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５】

前記プリンタドライバは、

前記入力されたＸＰＳ形式の描画データ内のプリントチケットから印刷解像度と印刷用紙サイズの情報を取得して前記印刷装置で表現可能な最小のオブジェクト幅を導出し、

前記所定のグラフィック描画命令を、当該導出した前記印刷装置で表現可能な最小のオブジェクト幅での描画を指定するグラフィック描画命令に変更する、

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６】

前記プリンタドライバは、前記所定のグラフィック描画命令を、前記印刷装置で表現可能な最小のオブジェクト幅での描画を指定する予め定められた専用のグラフィック描画命令に変更す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７】

前記オブジェクト幅が０のグラフィック描画命令は、線オブジェクトのグラフィック描画命令であることを特徴とする請求項４に記載の情報処理装置。

【請求項８】

前記プリンタドライバは、前記入力されたＸＰＳ形式の描画データが、ＸＰＳアプリケーションの印刷処理に基づき生成されるＸＰＳ形式の描画データである場合、当該ＸＰＳ形式の描画命令に含まれる前記所定のグラフィック描画命令を変更しない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９】

入力されたＸＰＳ形式の描画データから、印刷装置に対応した印刷データを生成するプリンタドライバによって実行される、印刷データ生成方法であって、

前記印刷データ生成方法は、

前記入力されたＸＰＳ形式の描画データが、ＧＤＩアプリケーションの印刷処理に基づき生成されたＧＤＩ形式の描画データから変換モジュールで変換されたＸＰＳ形式の描画データであるか判定する第１の判定ステップと、

前記入力されたＸＰＳ形式の描画データが、所定のグラフィック描画命令を含むか判定する第２の判定ステップと、

前記第１の判定ステップにおいて、前記入力されたＸＰＳ形式の描画データが、前記ＧＤＩアプリケーションの印刷処理に基づき生成されたＧＤＩ形式の描画データから前記変換モジュールで変換されたＸＰＳ形式の描画データであると判定され、且つ、前記第２の判定ステップにおいて、前記入力されたＸＰＳ形式の描画データが、前記所定のグラフィック描画命令を含むと判定された場合、当該所定のグラフィック描画命令を前記印刷装置で表現可能な特定の描画を指定するグラフィック描画命令に変更する変更ステップと、

前記変更ステップで変更した後のグラフィック描画命令を含む描画データに基づいて前記印刷データを生成する生成ステップと、

を含むことを特徴とする印刷データ生成方法。

【請求項１０】
入力されたＸＰＳ形式の描画データから、印刷装置に対応した印刷データを生成するプリンタドライバであって、
前記プリンタドライバは、コンピュータを、
前記入力されたＸＰＳ形式の描画データが、ＧＤＩアプリケーションの印刷処理に基づき生成されたＧＤＩ形式の描画データから変換モジュールで変換されたＸＰＳ形式の描画データであるかどうかを判定する第１の判定手段と、
当該入力されたＸＰＳ形式の描画データが、所定のグラフィック描画命令を含むかどうかを判定する第２の判定手段と、
前記第１の判定手段において、前記入力されたＸＰＳ形式の描画データが、前記ＧＤＩアプリケーションの印刷処理に基づき生成されたＧＤＩ形式の描画データから前記変換モジュールで変換されたＸＰＳ形式の描画データである、と判定し、且つ、前記第２の判定手段において、前記入力されたＸＰＳ形式の描画データが前記所定のグラフィック描画命令を含む、と判定した場合に、当該所定のグラフィック描画命令を前記印刷装置で表現可能な特定の描画を指定するグラフィック描画命令に変更する変更手段と、
前記変更手段で変更した後のグラフィック描画命令を含む描画データに基づいて前記印刷データを生成する生成手段と、
して機能させるためのプログラムであることを特徴とするプリンタドライバ。
【請求項１１】
前記所定のグラフィック描画命令とは、オブジェクト幅が０のグラフィック描画命令であり、
前記印刷装置で表現可能な特定の描画を指定するグラフィック描画命令とは、前記印刷装置で表現可能な最小のオブジェクト幅での描画を指定するグラフィック描画命令である、ことを特徴とする請求項１０に記載のプリンタドライバ。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、プリンタドライバの描画制御技術に関するものである。
【背景技術】
【０００２】
マイクロソフト社のプリンティングシステムでは、従来よりＧＤＩ（Graphic Device Interface）と呼ばれるグラフィックスエンジンが使用されてきた。ＧＤＩを用いた従来の印刷処理フローは、以下のとおりである。
１）ユーザからの印刷指示に応じて、ＧＤＩアプリケーション（以下「ＧＤＩアプリ」と呼ぶ。）がＧＤＩのサービス関数群を呼び出して描画命令を発行する。
２）発行されたＧＤＩ描画命令は、描画データとしてスプールされ、ＧＤＩ用のプリンタドライバ（以下「ＧＤＩドライバ」と呼ぶ）に渡される。
３）ＧＤＩドライバでは、ＧＤＩ描画命令で構成される描画データ（以下、「ＧＤＩ形式の描画データ」と呼ぶ。）を印刷装置が解釈可能な印刷データ（いわゆるＰＤＬデータ）に変換し、印刷装置に出力する。
【０００３】
このような、ＧＤＩアプリからＧＤＩドライバを介して印刷を行う処理フローを「ＧＤＩプリントパス」と呼ぶこととする。近年、このＧＤＩプリントパスに、ＸＰＳプリントパスと呼ばれる新しい印刷処理フローが追加された。ＸＰＳは、ＸＭＬベースの電子文書フォーマットで、XML Paper Specificationの略である。ＸＰＳプリントパスでは、ＸＰＳを扱うアプリケーション（以下「ＸＰＳアプリ」と呼ぶ。）でＸＰＳ形式の描画データを生成し、ＸＰＳ用のプリンタドライバ（以下、「ＸＰＳドライバ」と呼ぶ。）にて印刷データに変換され、印刷装置に出力される。
【０００４】
ＧＤＩプリントパスとＸＰＳプリントパスとは排他的な関係にあるわけではなく、例え

ばＧＤＩアプリがＧＤＩを使って生成したＧＤＩ形式の描画データをＸＰＳ形式の描画データに変換してＸＰＳドライバで印刷データにすることが可能である。また、逆にＸＰＳアプリによるＸＰＳ形式の描画データをＧＤＩ形式の描画データに変換してＧＤＩドライバで印刷データにすることも可能である。すなわち、ＧＤＩアプリからＸＰＳドライバを介して印刷を行う印刷処理フローや、ＸＰＳアプリからＧＤＩドライバを介して印刷を行う印刷処理フローも存在する。例えば、ＧＤＩアプリからＸＰＳドライバを用いて印刷を行う場合、ＧＤＩ形式の描画データが変換モジュールによってＸＰＳ形式の描画データに変換される。この変換モジュールはＭＸＤＷ（Microsoft XPS Document Writer）と呼ばれる。ＸＰＳ形式に変換された描画データは、ＸＰＳスプールファイルに格納された後、ＸＰＳドライバにて印刷データへ変換される。

【０００５】

このようにＷｉｎｄｏｗｓ（登録商標）ＶＩＳＴＡ以降のＯＳにおいては４つの印刷処理フローが利用可能になっている。これにより、ＧＤＩドライバあるいはＸＰＳドライバのどちらか一方を用意すれば、ＧＤＩアプリおよびＸＰＳアプリの両方で生成された描画データの印刷処理に対応することが出来る。

【０００６】

しかしながら、ＸＰＳ形式とＧＤＩ形式とでは仕様が異なることから、描画データの形式変換には様々な問題が存在する。その一例として、印刷可能領域からはみ出した部分に印刷画像を割り当てた印刷要求が、ＧＤＩアプリからＸＰＳドライバに対して発行されてしまうという問題が挙げられる。この問題に対しては、ＸＰＳドライバに入力されるＸＰＳ形式の描画データが、ＧＤＩアプリ由来のものか、ＸＰＳアプリ由来のものかを判断して印刷可能領域内に印刷されるように制御する技術が提案されている（特許文献１）。

【先行技術文献】

【特許文献】

【０００７】

【特許文献１】特開２００８－２７６７４５号公報

【発明の概要】

【発明が解決しようとする課題】

【０００８】

そして、描画データの形式変換における問題として、上述の問題の他に、グラフィック属性のオブジェクトの描画に関する問題の存在が明らかとなった。例えば、グラフィック属性のオブジェクトの描画命令でオブジェクト幅＝０を指定した場合、プリンタドライバによってオブジェクト（例えば、線）が描画されたりされなかったりするという問題である。この問題は、ＧＤＩ形式におけるオブジェクト幅＝０はデバイスで表示できる最小の幅を意味するのに対し、ＸＰＳ形式におけるオブジェクト幅＝０は当該オブジェクトをまったく描画しないことを意味することに起因している。つまり、ＧＤＩアプリがＧＤＩを介して生成した描画データにオブジェクト幅＝０の線の描画を指定する描画命令が含まれる場合に、ＧＤＩドライバで印刷を行えば線が描画されるのに対し、ＸＰＳドライバでの印刷では線が描画されないことになる。すなわち、同じＧＤＩアプリからなされた同じ描画命令であるにも関わらず、所定のグラフィック描画命令に対して、利用するプリンタドライバの種類によって印刷結果に違いが出るのでは、印刷品質に大きな影響を及ぼすことになる。

【課題を解決するための手段】

【０００９】

本発明に係る情報処理装置は、入力されたＸＰＳ形式の描画データから、印刷装置に対応した印刷データを生成するプリンタドライバを備えた情報処理装置であって、前記プリンタドライバは、前記入力されたＸＰＳ形式の描画データが、ＧＤＩアプリケーションの印刷処理に基づき生成されたＧＤＩ形式の描画データから変換モジュールで変換されたＸＰＳ形式の描画データであるかどうかを判定し、当該入力されたＸＰＳ形式の描画データが、所定のグラフィック描画命令を含むかどうかを判定し、当該判定の結果、前記入力さ

れたＸＰＳ形式の描画データが、前記ＧＤＩアプリケーションの印刷処理に基づき生成されたＧＤＩ形式の描画データから前記変換モジュールで変換されたＸＰＳ形式の描画データであり、且つ、前記所定のグラフィック描画命令を含む、と判定した場合、当該所定のグラフィック描画命令を前記印刷装置で表現可能な描画を指定するグラフィック描画命令に変更し、当該変更後のグラフィック描画命令を含む描画データに基づいて前記印刷データを生成する、ことを特徴とする。

【発明の効果】

【００１０】

本発明によれば、入力されたＸＰＳ形式の描画データから印刷装置に対応する印刷データを生成するプリンタドライバ（ＸＰＳドライバ）において、当該入力されたＸＰＳ形式の描画データが、ＧＤＩアプリケーションの印刷処理で生成されたＧＤＩ形式の描画データから所定の変換モジュールによってＸＰＳ形式の描画データに変換されたものであり、且つ、所定のグラフィック描画命令を含んでいる場合、当該所定のグラフィック描画命令を印刷装置で表現可能な特定の描画を指定するグラフィック描画命令に変更し、当該変更後のグラフィック描画命令を含む描画データに基づいて印刷データを生成することにより、ＧＤＩアプリケーションからの同じ描画命令でありながら、使用するプリンタドライバの種類がＸＰＳドライバであるかＧＤＩドライバであるかによって印刷結果に差が生じてしまうのを防ぐことができる。

【図面の簡単な説明】

【００１１】

【図１】印刷データを生成する情報処理装置の構成の一例を示す機能ブロック図である。

【図２】情報処理装置のハードウェア構成の一例を示すブロック図である。

【図３】ＵＩドライバの内部構成の一例を示す図である。

【図４】グラフィックスドライバの内部構成の一例を示す図である。

【図５】印刷データ生成処理の流れを示すフローチャートである。

【図６】ＸＰＳデータのデータ構造を示す図である。

【図７】実施例１に係るアプリ判定処理の詳細を示すフローチャートである。

【図８】（ａ）は探索文字列設定情報の一例を示す図であり、（ｂ）は探索の対象ファイルとしてのプリントチケットの一例を示す図である。

【図９】描画命令変更処理の詳細を示すフローチャートである。

【図１０】ＸＰＳドライバのＵＩ画面の一例を示す図である。

【図１１】実施例２に係るアプリ判定処理の詳細を示すフローチャートである。

【図１２】実施例２に係るプリントチケットの一例である。

【発明を実施するための形態】

【００１２】

以下、図面を参照して本発明の実施の形態を詳しく説明する。なお、以下の実施の形態は特許請求の範囲に係る発明を限定するものでなく、また実施の形態で説明されている特徴の組み合わせの全てが発明の解決手段に必須のものとは限らない。

【００１３】

［実施例１］

図１は、本実施例に係る、印刷データを生成する情報処理装置の構成の一例を示す機能ブロック図である。情報処理装置１００には、不図示のオペレーティングシステム（以下、ＯＳ）がインストールされている。本実施例では、ＯＳとしてＷｉｎｄｏｗｓ（登録商標）を前提としている。そして、情報処理装置１００には描画データを生成するソフトウェアとして、ＧＤＩアプリケーション１１０とＸＰＳアプリケーション１２０の２種類の描画用ソフトウェアがＯＳ上で動作する。ＧＤＩアプリ１１０は、ユーザからの印刷指示を受けてＧＤＩ関数を呼び出して、オブジェクトの属性（文字、グラフィック、イメージなど）に応じたＧＤＩ形式の描画データを、ＧＤＩ１１１を用いて生成する。ＧＤＩ１１１は、一般にグラフィックスエンジンと呼ばれる。ＧＤＩ形式の描画データは、そのままではＸＰＳドライバ１３０で入力が受け付けられないので、ＭＸＤＷ１１２に送られる。

ＭＸＤＷ１１２は、ＧＤＩ－ＸＰＳ変換手段であって、ＧＤＩ形式の描画データをＸＰＳ形式の描画データに変換するモジュールでありＯＳによって提供される。ＭＸＤＷ１１２によってＸＰＳ形式に変換された描画データが、ＸＰＳドライバ１３０に入力される。ＸＰＳアプリ１２０は、ＸＰＳ形式の描画データを直接生成する。なお、ＸＰＳ形式の描画データは、ＷＰＦ（Windows Presentation Foundation）と呼ばれるグラフィックエンジンを介して生成される場合もあるが、本発明の特徴とは無関係であるので説明を省く。ＸＰＳアプリ１２０で生成されたＸＰＳ形式の描画データは、そのままＸＰＳドライバ１３０に入力される。

【００１４】

ＸＰＳドライバ１３０は、入力描画データとしてＸＰＳ形式の描画データのみを受け付けるプリンタドライバである。ＸＰＳドライバ１３０は、印刷部数や用紙サイズといった印刷設定を管理し、入力された描画データから印刷装置が解釈可能な印刷データ（ＰＤＬデータ）を生成する。ＸＰＳドライバ１３０で生成された印刷データは、スプーラ１４０を経て印刷装置に送信される。ＸＰＳドライバ１３０は、ＵＩドライバ１３１、グラフィックスドライバ１３２、設定データ保持部１３３から構成される。ＵＩドライバ１３１は、ＸＰＳドライバ１３０のユーザインタフェース画面（以下、ＵＩ画面）の表示や、ユーザから入力された印刷設定に関する情報の保存を行う。グラフィックスドライバ１３２は、ＵＩドライバ１３１でユーザが設定した印刷設定情報に基づいて、ＸＰＳ形式の描画データから印刷データを生成する。ＵＩドライバ１３１及びグラフィックスドライバ１３２の詳細については後述する。設定データ保持部１３３は、ＸＰＳドライバ１３０が印刷データ生成時に参照する、印刷に関連する各種設定データを保持する。この設定データはＸＭＬ形式で記述され、ＵＩドライバ１３１及びグラフィックスドライバ１３２の動作設定に関する情報が含まれる。本実施例では、アプリケーションの種類を判定するための探索文字列に関する情報が設定データの一部として保持されており、印刷データ生成時に参照される。

【００１５】

図２は、情報処理装置１００のハードウェア構成の一例を示すブロック図である。なお図２に示す構成は一例であって、情報処理装置１００は単体の機器であっても、ＬＡＮやＷＡＮ等のネットワークを介して接続がされた複数の機器からなるシステムであってもよい。情報処理装置１００は、ＲＯＭ２０６やハードディスク２０３に記憶された処理プログラムをＣＰＵ２０１が実行することで様々な処理を実現し、システムバス２１０に接続された各部を総括的に制御する。また、ＣＰＵ２０１は、マウスカーソル等で指示されたコマンドに基づいて、予め登録されたウインドウを開くといった種々の処理を実行する。例えばユーザが印刷の実行を指示する際は、印刷設定のウインドウを開き、ＸＰＳドライバ１３０に対する各種設定を行う。ＲＡＭ２０２は、ＣＰＵ２０１の主メモリ、ワークエリア等として機能する。ハードディスク２０３には、ＯＳ、ブートプログラム、各種のアプリケーション、フォントデータ、ユーザファイル、編集ファイル等が記憶される。ＸＰＳドライバ１３０もハードディスク２０３に保存される。ディスプレイコントローラ２０４は、不図示のディスプレイの表示を制御する。プリンタコントローラ２０５は、所定の双方向性インタフェース（不図示）を介して印刷装置に接続され、印刷装置との通信制御を担う。なお、ＣＰＵ２０１は、例えばＲＡＭ２０２上に設定された表示情報ＲＡＭへのアウトラインフォントの展開（ラスタライズ）処理を実行し、ディスプレイ上でのWYSIWYG（ウィジウィグ）を可能としている。外部記憶ドライブ２０７は、ＣＤやＤＶＤといった記憶媒体とのアクセスを制御する。キーボードコントローラ２０９は、キーボードやポインティングデバイスからのキー入力を制御する。

【００１６】

続いて、ＵＩドライバ１３１の詳細について説明する。図３は、ＵＩドライバ１３１の内部構成の一例を示す図である。ＵＩドライバ１３１は、ＵＩ表示部３０１、ＵＩ設定部３０２、ドライバ初期化部３０３、ドライバハンドル返却部３０４から構成される。

【００１７】

ＵＩ表示部３０１は、ユーザが印刷設定を行うためのＵＩ画面をディスプレイ上に表示する。ＵＩ設定部３０２は、ＵＩ表示部３０１を介してユーザが設定または変更した印刷設定を保存する。ドライバ初期化部３０３は、ＵＩドライバ１３１の初期化を行う。アプリケーションやＧＤＩ１１１からドライバ初期化指示を受けると、初期化処理で生成したドライバハンドルをアプリケーションまたはＧＤＩ１１１に渡す。ドライバハンドルとは、ドライバを固有に識別するための識別名である。初期化処理で生成したドライバハンドルはドライバハンドル返却部３０４にて初期化指示を出したアプリケーションに返却する。ドライバハンドルを受け取ったアプリケーションは、ユーザに対して印刷設定入力の許可や、ＸＰＳドライバ１３０に対して描画データの出力を行う。なお、ユーザが印刷設定を行うために、ドライバの初期化指示と共にＵＩ画面の表示指示がＵＩドライバ１３１に渡される場合がある。その場合には、ＵＩ表示部３０１にてＵＩダイアログを表示する。

【００１８】

続いて、グラフィックスドライバ１３２の詳細について説明する。図４は、グラフィックスドライバ１３２の内部構成の一例を示す図である。前述したように、グラフィックスドライバ１３２はＸＰＳ形式の描画データ（以下、「ＸＰＳデータ」と呼ぶ。）を受け取って、印刷設定情報に応じた印刷データを生成し、スプーラ１４０を介して印刷装置に送信する機能を有する。グラフィックスドライバ１３２は、ＸＰＳ解析部４０１と印刷データ生成部４０２とで構成される。

【００１９】

ＭＸＤＷ１１２又はＸＰＳアプリ１２０から入力されるＸＰＳデータは、ドキュメント情報の全てがXMLで記述されており、記述内容ごとの複数ファイルをZIP圧縮により1つにまとめている。入力されたＸＰＳデータは、ＸＰＳ解析部４０１において解凍された後に解析される。解析の内容としては、例えば、入力されたＸＰＳデータがＧＤＩアプリ１１０に由来するものかどうかの判定処理がある。解析処理の詳細については後述する。また、ＸＰＳ解析部４０１は、印刷データの前段階のデータである中間データの生成も行なう。印刷データ生成部４０２は、中間データと印刷設定情報を基に印刷データ（ＰＤＬデータ）を生成する。生成された印刷データはスプーラ１４０に出力される。

【００２０】

次に、本実施例に係る、ＸＰＳドライバ１３０における印刷データ生成処理について詳しく説明する。図５は、印刷データ生成処理の流れを示すフローチャートである。この印刷データ生成処理は、ＸＰＳドライバ１３０に対する印刷指示に応答して開始する。具体的には、ユーザの印刷指示に応じてＲＯＭ２０６に格納された所定のプログラムがＲＡＭ２０２に展開され、これをＣＰＵ２０１が実行することで実現される。

【００２１】

ステップ５０１において、ＸＰＳドライバ１３０に、ＭＸＤＷ１１２或いはＸＰＳアプリ１２０からＸＰＳデータが入力される。入力されたＸＰＳデータは、グラフィックスドライバ１３２内のＸＰＳ解析部４０１に送られる。この時点でＸＰＳデータはZIP形式の状態である。ＸＰＳドライバ１３０がＸＰＳデータの入力を受け付けた時点では、それがＸＰＳアプリ１２０から直接に入力されたものか、当初はＥＭＦ形式であった描画データがＭＸＤＷ１１２でＸＰＳ形式に変換された後に入力されたものかは分からない。そこで、入力されたＸＰＳデータが、ＧＤＩアプリ１１０に由来するものか、ＸＰＳアプリ１２０に由来するものかを識別するために後述のステップ５０３にてアプリ判定処理を行なう。

【００２２】

ステップ５０２において、ＸＰＳ解析部４０２は、ZIP形式のＸＰＳデータを解凍する。図６は、ＸＰＳデータのデータ構造を示す図である。ＸＰＳデータはツリー構造になっており、rootフォルダの下の第一階層に「Metadata」「_rels」、「Documents」といった複数のフォルダが存在する。

【００２３】

ステップ５０３において、ＸＰＳ解析部４０２は、アプリ判定処理を行う。このアプリ

判定処理は、入力されたＸＰＳデータの生成元のアプリケーションが、ＧＤＩアプリ１１０であるのかＸＰＳアプリ１２０であるのかを判定する処理である。アプリ判定処理の詳細については後述する。判定の結果、生成元のアプリがＧＤＩアプリ１１０であれば、ステップ５０５に進む。一方、生成元のアプリがＸＰＳアプリ１２０であれば、描画命令に手を加えることなくそのまま中間データを生成し、ステップ５０７に進む。

【００２４】

ステップ５０５において、ＸＰＳ解析部４０１は、解凍して得られた「Documents」フォルダ内のページ単位のファイル内に、グラフィック属性のオブジェクトの幅として０が指定された描画命令が存在するかどうかを判定する。図６に示す“1.fpage”や“2.fpage”がページ単位のファイルであり、その中には、例えば、以下に示す「Path」の描画命令が含まれている。

<Path Data="F1 M 19.2,19.2 L 96,19.2" Stroke="#ffff0000" StrokeThickness="0" StrokeLineJoin="Round" StrokeStartLineCap="Round" StrokeEndLineCap="Round" Clip="M 0,0 L 0,1083.84 755.2,1083.84 755.2,0 z" />

【００２５】

上記「Path」は線の描画を指定する命令であり、各要素は線描画の設定内容を表している。例えばData要素は、(19.2, 19.2)の座標に移動(M)し、線(L)を(96, 19.2)に対して描画することを表している。そして、オブジェクト幅としての線幅に関する要素は“StrokeThickness”であり、その要素値が“０”であること、すなわちＸＰＳの仕様では線を描画しないことを意味している。

【００２６】

判定の結果、オブジェクト幅が０のグラフィック描画命令が存在する場合は、ステップ５０６に進む。一方、オブジェクト幅が０のグラフィック描画命令が存在しない場合は、ファイル内の描画命令に手を加えることなく中間データを生成してステップ５０７に進む。

【００２７】

ステップ５０６において、ＸＰＳ解析部４０１は、オブジェクト幅が０のグラフィック描画命令を、最小幅でのオブジェクト（例えば、線オブジェクト）の描画を指定するグラフィック描画命令に変更する処理を行なう。例えば、上述の「Path」の描画命令の場合、以下のように変更される。

<Path Data="F1 M 19.2,19.2 L 96,19.2" Stroke="#ffff0000" StrokeThickness="0.16" StrokeLineJoin="Round" StrokeStartLineCap="Round" StrokeEndLineCap="Round" Clip="M 0,0 L 0,1083.84 755.2,1083.84 755.2,0 z" />

【００２８】

上記具体例の場合、“StrokeThickness”の要素値“０”が、“0.16”に変更されており、これは0.16の幅(単位はpoint)で線を描画することを意味している。

このように、オブジェクト幅が０のグラフィック描画命令をＸＰＳの仕様に従ってそのまま描画しないと解釈するのではなく、ＧＤＩの仕様に従ってデバイスの最小単位で描画するという内容に解釈し直し、描画命令の内容が変更される。この描画命令変更処理の具体的な流れについては後述する。そして、この描画命令変更処理の結果を踏まえて或いは描画命令変更処理の過程において中間データが生成され、印刷データ生成部４０２に送られる。

【００２９】

ステップ５０７において、印刷データ生成部４０２は、中間データと印刷設定情報に基づいて印刷データを生成する。生成された印刷データはスプーラに送られる。

【００３０】

以上が、ＸＰＳドライバ１３０における印刷データ生成処理の内容である。上述のとおり、ＥＭＦ（ＧＤＩ）とＸＰＳとでは仕様が異なる、オブジェクト幅＝０のグラフィック描画に関して生成元アプリケーションの種類に応じた印刷データの生成を行うことで、各アプリケーションで想定している仕様で印刷を行うことが可能となる。

【００３１】
続いて、上述した図５のフローチャートにおけるアプリ判定処理（ステップ５０３）の詳細について説明する。本実施例のアプリ判定処理では、予め設定された探索条件の情報に従って、ＸＰＳ形式の描画データ内の所定のファイルから特定の文字列を探索することで、ＸＰＳドライバに入力されたＸＰＳデータの生成元のアプリを特定する。図７は、本実施例に係るアプリ判定処理の詳細を示すフローチャートである。
【００３２】
ステップ７０１では、生成元アプリを特定するための探索文字列に関する設定情報が設定データ保持部１３３から取得される。図８（ａ）は、本ステップで取得する探索文字列設定情報の一例を示しており、ＸＰＳデータ内のどの部分（ファイル）を対象に如何なる文字列を探索するかが指定されている。図８（ａ）に示す探索文字列設定情報において、探索条件の単位であるSetting要素８０１は、「SearchString」、「SearchPath」、「Mode」の３つの要素をさらに有している。「SearchString」は探索する文字列を指定してする要素であり、ここではＭＸＤＷ１１２を表す文字列である“microsoftxpsdocumentwriter”がその値として設定されている。ＭＸＤＷ１１２を指すこの文字列は、ＥＭＦ形式からＸＰＳ形式に変換する際にＭＸＤＷ１１２において、ジョブレベルのプリントチケットに記述されるものである。そのため、この文字列を含むプリントチケットを持つＸＰＳデータはＧＤＩアプリ１１０が生成元であると判断することができる。なお、ＭＸＤＷ１１２を表す文字列であればよいので“microsoftxpsdocumentwriter”に限定されるわけではない。
【００３３】
「SearchPath」は、ＸＰＳデータの中のどのファイルから文字列を探索するかを特定するためのパスを指定する要素である。ここではその値として“/MeaData/*_PT.xml”が設定されている。前述のとおりＸＰＳデータは複数のファイルがZIP圧縮されたデータ形式であるため、「SearchPath」で探索先のパスを指定することによりどのファイルを探索するかを特定する。本実施例のように、正規表現を用いることで複数のファイル（例えば“A_PT.xml”や“B_PT.xml”といったファイル名のファイル）を探索対象にすることができる。ジョブレベルのプリントチケットのファイル名は“JOB_PT.xml”であるので、正規表現を用いることなく“JOB_PT.xml”の文字列を設定値としてもよい。「Mode」は、「SearchPath」で指定されたパスに対応するファイル内に「SearchString」で指定された探索文字列が存在した場合に、どの種類のアプリケーションに由来するＸＰＳデータであるとするかを特定する要素である。ここではその値として“cpk:GDI”が設定されている。つまり、この場合、メタデータフォルダ内の“_PT.xml”をファイル名に含むファイル内に“microsoftxpsdocumentwriter”の文字列があれば、ＧＤＩアプリ１１０由来のＸＰＳデータと判定されることになる。もし「Mode」要素の設定値として“cpk:XPS”が入っていれば、ＸＰＳアプリ１２０に由来すると判定されることになる。なお、図８（ａ）の例では探索条件は１つ（Setting要素が１つのみ）しか設定されていないが、複数の探索条件を設定し、いずれかの探索条件（Setting要素）を満たした場合に、アプリケーションの種類が特定されるようにしてもよい。このように設定情報において、ＸＰＳデータのどのファイルからどの文字列を探索し、見つかった場合にどの種類のアプリケーションと判定するか、という内容を定義している。これにより、プログラム自体はそのままで設定情報における定義変更によって探索条件を変更することができる。
【００３４】
ステップ７０２では、探索文字列の設定情報に従って、「SearchPath」で指定されたパスが示すファイル内に、「SearchString」で指定された探索文字列が存在するかどうかが判定される。図８（ｂ）は、探索の対象ファイルとしてのプリントチケットの一例を示す図である。図８（ｂ）に示す例では、“psf:PrintTicket”要素の“xmlns:ns0000”要素の中に上記探索文字列“microsoftxpsdocumentwriter”が含まれている。判定の結果、対象ファイル内に探索文字列が存在する場合はステップ７０３に進む。一方、対象ファイル内に探索文字列が存在しない場合はステップ７０４に進む。

【００３５】
ステップ７０３では、入力ＸＰＳデータについて、ＭＸＤＷ１１２を経由したＧＤＩアプリ１１０が生成元であると決定され、本処理を終える。
【００３６】
ステップ７０４では、入力ＸＰＳデータについて、ＸＰＳアプリ１２０が生成元であると決定され、本処理を終える。
以上が、本実施例に係るアプリ判定処理の内容である。
【００３７】
続いて、上述した図５のフローチャートにおける描画命令変更処理（ステップ５０６）の詳細について説明する。図９は、描画命令変更処理の詳細を示すフローチャートである。
【００３８】
ステップ９０１では、ユーザが設定した印刷設定情報から印刷解像度と印刷用紙サイズの情報が取得される。この印刷設定情報はＸＰＳデータ内の上述のプリントチケットと呼ばれる部分に記述されている。ＸＰＳ解析部４０１は、プリントチケットを解析して、印刷解像度と印刷用紙サイズの情報を取得する。図８（ｂ）の例では、印刷解像度は“psk:PageResolution”要素にて100(dpi)が、印刷用紙サイズは“psk:PageMediaSize”要素にてA４に相当するサイズ（縦210㎜×横297㎜）がそれぞれ設定されている。
【００３９】
ステップ９０２では、取得した印刷解像度及び印刷用紙サイズの情報から、デバイスで表現可能な最小の幅である１ピクセル(pixel)の幅が導出される。例えば、上述のように印刷解像度が100(dpi)、印刷用紙サイズがA4に設定されていた場合、画像の画素数は縦827pixel×横1169pixelとなる。この場合、１ピクセルの幅として、約0.25㎜が導出されることとなる。
【００４０】
ステップ９０３では、オブジェクト幅＝０（ＸＰＳ形式では「描画しない」を意味）を指定するグラフィック描画命令を、１ピクセル幅に相当するオブジェクト幅（ここでは0.25㎜）でオブジェクトの描画を指定するグラフィック描画命令に変更する。
【００４１】
以上が、描画命令変更処理の内容である。この変更処理により、入力されたＸＰＳデータの生成元アプリがＧＤＩアプリであった場合、そのままでは何も描画されない内容の描画命令が、ＧＤＩ（ＥＭＦ）の仕様と同様、最小のオブジェクト幅で描画する描画命令に変更される。
【００４２】
なお、上述の例では、１ピクセル幅を計算によって求めているが、印刷装置や生成する印刷データの仕様によっては、線などのグラフィック属性のオブジェクトを最小幅で描画する専用の描画命令が予め用意されている場合もある。このような場合は、上述のステップ９０２における導出処理を省略し、図５のフローにおける印刷データの生成（ステップ５０７）にて、オブジェクト幅が０のグラフィック描画命令については、上記専用の描画命令に変更した上で印刷データを生成すればよい。この場合、印刷装置側でオブジェクト幅の計算が行われることになる。
【００４３】
また、本実施例では、情報処理装置にインストールされているプリンタドライバがＸＰＳドライバのみの構成であったが、さらにＧＤＩドライバがインストールされていてもよい。本実施例を適用することで、ＧＤＩアプリで生成した描画データについていずれのプリンタドライバを用いても同じ印刷結果を得ることができる。
【００４４】
本実施例によれば、ＸＰＳドライバに入力されたＸＰＳデータがＧＤＩアプリに由来する場合は、オブジェクト幅＝０のグラフィック描画命令が、ＧＤＩアプリの仕様に則ってオブジェクトの最小幅での描画を指定するグラフィック描画命令に変更される。これによ

り、ＧＤＩアプリでユーザが意図した内容の印刷結果を、ＸＰＳドライバを介した印刷によって得ることができる。

【００４５】

［実施例２］

実施例１では、予め設定された探索条件の情報に従って、ＸＰＳデータ内の所定のファイルから特定の文字列を探索することで、ＸＰＳドライバに入力されたＸＰＳデータの生成元のアプリを特定し、必要に応じて描画命令を変更する態様であった。次に、印刷時に使用するＸＰＳドライバのユーザインタフェース画面でユーザが行なった設定内容に基づいて、ＸＰＳデータの生成元のアプリを特定する態様について、実施例２として説明する。なお、実施例１と基本的な構成は共通であり、以下では実施例１との差異点であるアプリ判定処理を中心に説明するものとする。

【００４６】

最初に、ユーザが印刷時に使用する、ＸＰＳドライバ１３０のユーザインタフェース画面（以下、ＵＩ画面）について説明する。図１０は、ＸＰＳドライバ１３０のＵＩ画面の一例を示す図である。このＵＩ画面１０００は、ＵＩドライバ１３１のＵＩ表示部３０１によって情報処理装置１００のディスプレイ上に表示される。ユーザは印刷時、文書等の作成アプリケーションから図１０に示すようなＵＩ画面１０００を使用して、出力用紙サイズや印刷の向きといった各種印刷設定を行う。このＵＩ画面１０００では、右上部にある「出力方式」の項目１００１において、出力モードとして「印刷（ＧＤＩ－ＸＰＳ互換）」を選択できるようになっている。この「印刷（ＧＤＩ－ＸＰＳ互換）」は、ＧＤＩアプリによって生成した描画データを印刷出力するユーザが、ＧＤＩドライバで印刷する時と同様の印刷結果を、ＸＰＳドライバを介した印刷で得たい場合に選択するモードである。なお、ユーザが、「印刷（ＧＤＩ－ＸＰＳ互換）」を選択しなかった場合（例えば、通常の印刷を示す「印刷（通常）」を選択した場合）は、すべてＸＰＳの仕様に従うことになる。すなわち、ＧＤＩアプリによって生成した描画データであったとしても、ＸＰＳの仕様に従って、オブジェクト幅＝０のグラフィック描画命令はオブジェクトの描画を行なわない描画命令として扱われることになる。

【００４７】

図１１は、本実施例に係るアプリ判定処理の詳細を示すフローチャートである。

ステップ１２０１では、受信したＸＰＳデータの中から、印刷設定情報が記述されたファイルであるプリントチケットが取得され、内容の解析がなされる。この際、ユーザが印刷設定時に上述のＵＩ画面１０００の項目１００１で選択した、出力モードの設定値を取得する。図１２は、本実施例に係るＸＰＳデータのプリントチケットの一例である。ここで、ＵＩ画面１０００における出力方式の項目１００１は、ＸＰＳデータのプリントチケット上では、「psk:OutputMode」要素で記載される。図１２の例では、「psk:OutputMode」の要素値として“GDI”が設定されており、これは出力方式として「印刷（ＧＤＩ－ＸＰＳ互換）」が選択されたことを意味している。ＸＰＳ解析部４０１はプリントチケット上の「psk:OutputMode」要素の設定値を読み取ることで、生成元のアプリケーションの種類を特定することが可能となる。

【００４８】

ステップ１１０２では、取得した出力モードの設定値（「psk:OutputMode」の要素値）が“GDI”であるかどうかが判定される。判定の結果、“GDI”である場合は、ステップ１１０３に進む。一方、“GDI”でない場合は、ステップ１１０４に進む。

【００４９】

ステップ１１０３では、入力されたＸＰＳデータはＭＸＤＷ１１２を経由したＧＤＩアプリ１１０が生成元のＸＰＳデータであると決定され、本処理を終える。

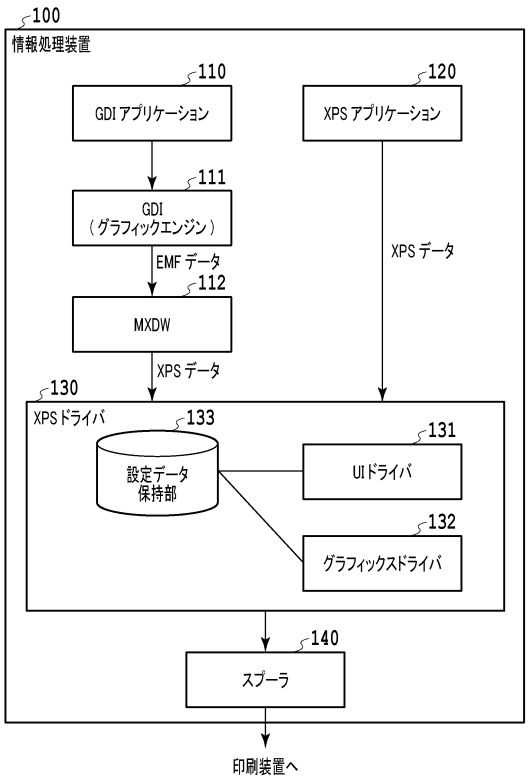
【００５０】

ステップ１１０４では、入力されたＸＰＳデータは、ＸＰＳアプリ４が生成元のＸＰＳデータであると決定され、本処理を終える。

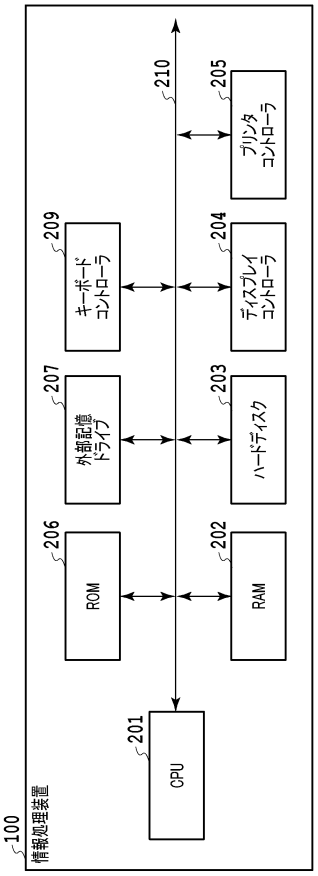
以上が、本実施例に係るアプリ判定処理の内容である。

【００５１】
本実施例によっても、実施例１と同様、ＧＤＩアプリでユーザが意図した内容の印刷結果を、ＸＰＳドライバを介した印刷によって得ることができる。
【００５２】
（その他の実施例）
本発明は、上述の実施形態の１以上の機能を実現するプログラムを、ネットワーク又は記憶媒体を介してシステム又は装置に供給し、そのシステム又は装置のコンピュータにおける１つ以上のプロセッサーがプログラムを読出し実行する処理でも実現可能である。また、１以上の機能を実現する回路（例えば、ＡＳＩＣ）によっても実現可能である。

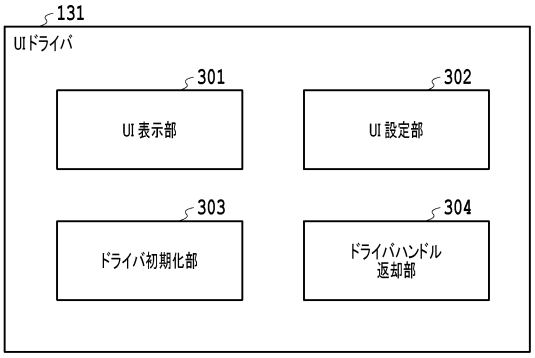
【図１】



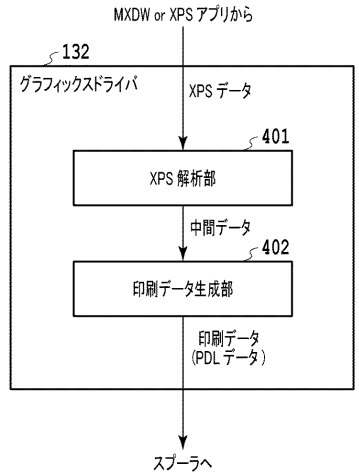
【図２】



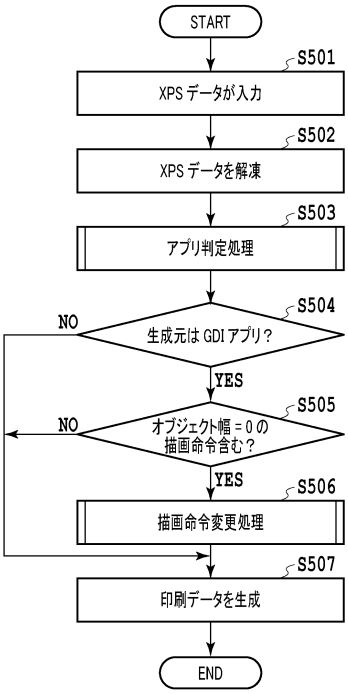
【図３】


131
UIドライバ
301
UI 表示部
302
UI 設定部
303
ドライバ初期化部
304
ドライバハンドル
返却部

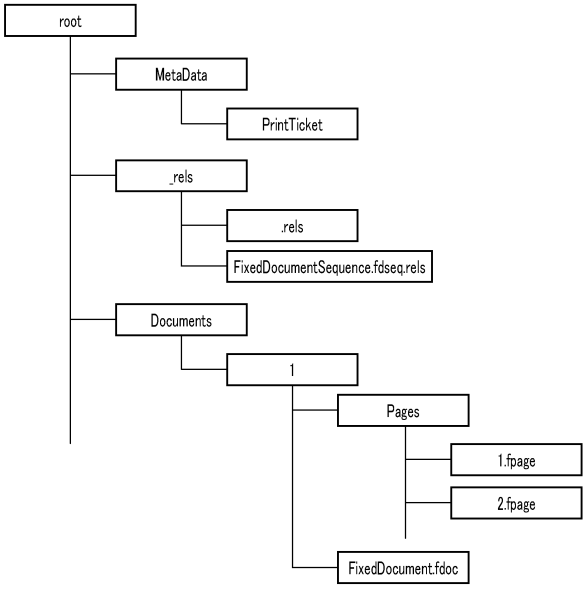
【図４】


MXDW or XPS アプリから
132
グラフィックスドライバ
XPS データ
401
XPS 解析部
中間データ
402
印刷データ生成部
印刷データ
(PDL データ)
スプーラへ

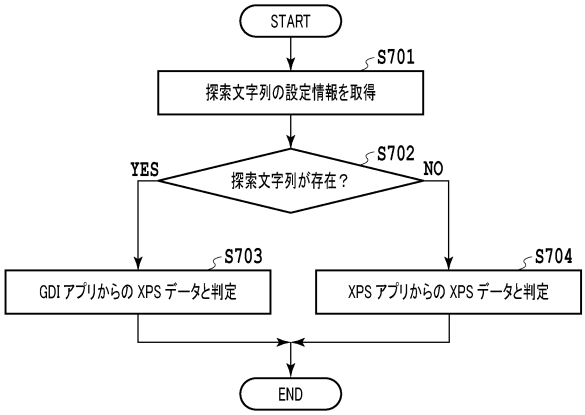
【図５】


START
S501
XPS データが入力
S502
XPS データを解凍
S503
アプリ判定処理
S504
生成元は GDI アプリ？
NO
YES
S505
オブジェクト幅 = 0 の
描画命令含む？
NO
YES
S506
描画命令変更処理
S507
印刷データを生成
END

【図６】


root
MetaData
PrintTicket
_rels
.rels
FixedDocumentSequence.fdseq.rels
Documents
1
Pages
1.fpage
2.fpage
FixedDocument.fdoc

【図７】


START
S701
探索文字列の設定情報を取得
S702
探索文字列が存在？
YES
NO
S703
GDI アプリからの XPS データと判定
S704
XPS アプリからの XPS データと判定
END

【図８】

(a)

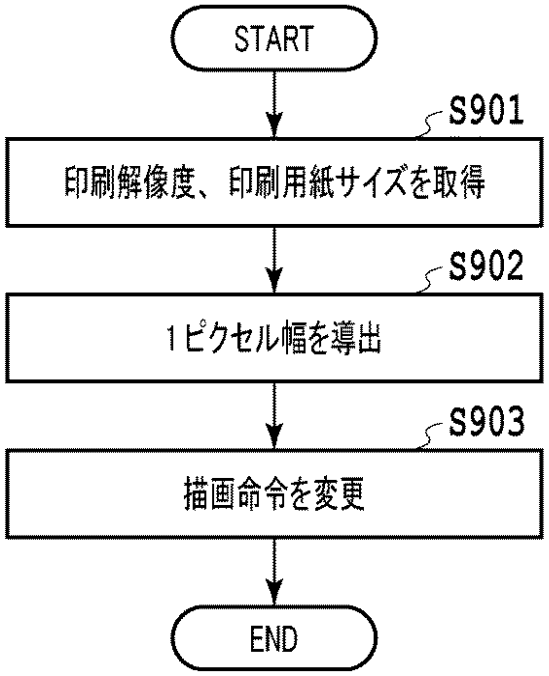
801

```
<?xml version="1.0" encoding=" utf-8"?>
<Settings>
  <Setting SearchString="microsoftxpsdocumentwriter" SearchPath=" /MetaData/*_PT.xml" Mode=" GDI" />
</Settings>
```

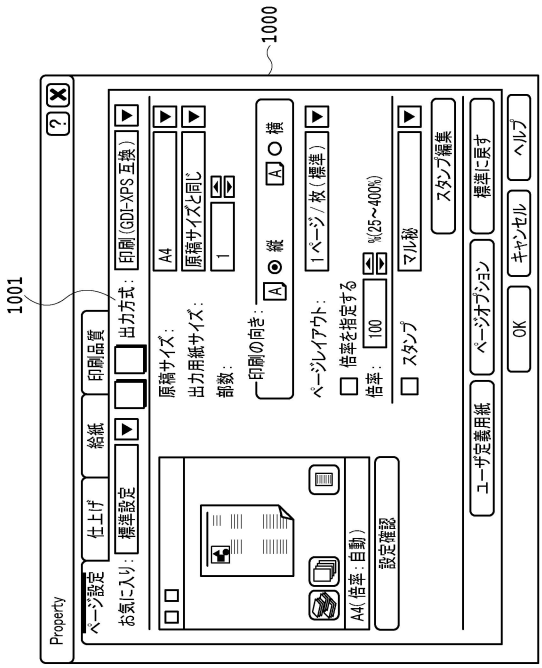
(b)

```
<?xml version="1.0" encoding="UTF-8"?>
<psf:PrintTicket xmlns:psk="http://schemas.microsoft.com/windows/2003/08/printing/printschemakeywords" version="1 "
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:psf="http://
schemas.microsoft.com/windows/2003/08/printing/printschemaFramework" xmlns:ns0000="http://schemas.microsoft.com/windows/
2006/06/printing/printschemakeywords/microsoftxpsdocumentwriter" >

<psf:Feature name="psk:PageMediaSize">
 <psf:Option name="psk:ISOA4">
  <psf:ScoredProperty name="psk:MediaSizeWidth">
   <psf:Value xsi:type="xsd:integer">210000</psf:Value>
  </psf:ScoredProperty>
  <psf:ScoredProperty name="psk:MediaSizeHeight">
   <psf:Value xsi:type="xsd:integer">297000</psf:Value>
  </psf:ScoredProperty>
 </psf:Option>
</psf:Feature>

<psf:Feature name="psk:PageResolution">
 <psf:Option name="psk:Option 1 ">
  <psf:ScoredProperty name=" psk:ResolutionX">
   <psf:Value xsi:type="xsd:integer">100</ psf:Value>
  </psf:ScoredProperty>
  <psf:ScoredProperty name=" psk:ResolutionY">
   <psfValue xsi:type="xsd:integer">100</psf:Value>
  </psf:ScoredProperty>
 </psf:Option>
</psf:Feature>

</psf:PrintTicket>
```

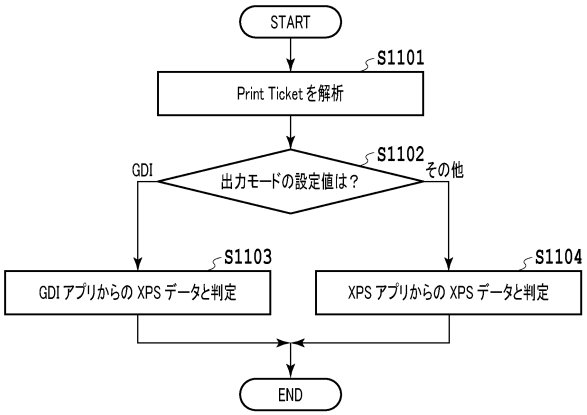
【図９】



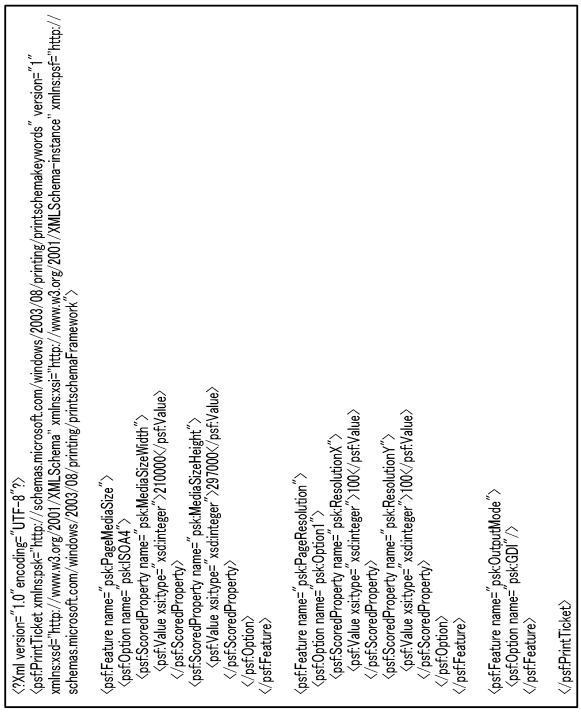
【図１０】



【図１１】



【図１２】


<?Xml version="1.0" encoding="UTF-8"?>
<psf:PrintTicket xmlns:psk="http://schemas.microsoft.com/windows/2003/08/printing/printschemakeywords" version="1"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:psf="http://
schemas.microsoft.com/windows/2003/08/printing/printschemaFramework">
<psf:Feature name="psk:PageMediaSize">
<psf:Option name="psk:ISOA4">
<psf:ScoredProperty name="psk:MediaSizeWidth">
<psf:Value xsi:type="xsd:integer">210000</psf:Value>
</psf:ScoredProperty>
<psf:ScoredProperty name="psk:MediaSizeHeight">
<psf:Value xsi:type="xsd:integer">297000</psf:Value>
</psf:ScoredProperty>
</psf:Option>
</psf:Feature>
<psf:Feature name="psk:PageResolution">
<psf:Option name="psk:Option1">
<psf:ScoredProperty name="psk:ResolutionX">
<psf:Value xsi:type="xsd:integer">100</psf:Value>
</psf:ScoredProperty>
<psf:ScoredProperty name="psk:ResolutionY">
<psf:Value xsi:type="xsd:integer">100</psf:Value>
</psf:ScoredProperty>
</psf:Option>
</psf:Feature>
<psf:Feature name="psk:OutputMode">
<psf:Option name="psk:GDI"/>
</psf:Feature>
</psf:PrintTicket>

フロントページの続き

(56)参考文献 特開２００８－２７６７４５（ＪＰ，Ａ）
特開２００９－０３７５１０（ＪＰ，Ａ）
特開２００７－２４９８５７（ＪＰ，Ａ）
特開２００６－１６３７００（ＪＰ，Ａ）
特開２０１２－１７３９８９（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ３／１２
Ｂ４１Ｊ２９／００－２９／７０
Ｂ４１Ｊ５／００－５／５２；２１／００－２１／１８