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

特許第4667281号
(P4667281)

(45)発行日　平成23年4月6日(2011.4.6)　(24)登録日　平成23年1月21日(2011.1.21)

(51)Int.Cl.		F I		
G06F 3/12	(2006.01)	G06F	3/12	D
G06F 3/00	(2006.01)	G06F	3/00	A
B41J 29/38	(2006.01)	B41J	29/38	Z
G06F 13/14	(2006.01)	G06F	13/14	310K

請求項の数 10　(全 21 頁)

(21)出願番号　特願2006-77711(P2006-77711)
(22)出願日　平成18年2月20日(2006.2.20)
(65)公開番号　特開2007-226755(P2007-226755A)
(43)公開日　平成19年9月6日(2007.9.6)
審査請求日　平成20年11月27日(2008.11.27)

(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　本田　秀樹
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】周辺装置制御方法及びその情報処理装置と制御プログラム

(57)【特許請求の範囲】
【請求項１】
周辺装置を管理する１つのキューに複数の周辺装置が割り当て可能な情報処理装置であって、
周辺装置と通信を行い前記周辺装置のステータスを、前記１つのキューを単位に接続されたポート毎に取得して、オペレーティングシステムが管理するステータス保持部に保持するステータス管理手段と、
前記周辺装置を接続したポートを識別するポート識別子を含むスキーマを引数として、前記オペレーティングシステムが提供するインタフェースの所定の関数をコールすることにより、周辺装置のステータスを前記ステータス管理手段へ問い合わせるステータス問合手段と、
前記ステータス問合手段から発行されたステータスの問い合わせメッセージに含まれる前記スキーマのポート識別子を認識するポート識別子認識手段と、
前記ポート識別子認識手段が認識したポート識別子に対応するポートに接続された周辺装置のステータスを、前記ステータス保持部から読み出して前記ステータス問合手段に返信するステータス返信手段と
を有することを特徴とする情報処理装置。
【請求項２】
前記ステータス問合手段から発行されたステータスの問い合わせメッセージに含まれる前記スキーマに前記ポート識別子が含まれているか否かを判定するポート識別子判定手段

を更に有し、
前記ステータス返信手段は、前記ポート識別子判定手段にてポート識別子が含まれてないと判定した場合に、予め決められたポートのステータスを返信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】
前記ステータス管理手段は、前記周辺装置のステータスを予め準備された記憶領域にポート単位で記録し、
前記ステータス問合手段は、前記ステータス管理手段への問合せと、前記記憶領域にポート単位で書き込まれたステータスの取得とを、予め定められた条件で切り替え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】
前記ステータス問合手段は、１つのキューに複数の周辺装置が割り当てられた場合に、通信メッセージに周辺装置が接続されているポートのポート識別子を含めることを特徴とする請求項１乃至３の何れか１項に記載の情報処理装置。
【請求項５】
周辺装置を管理する１つのキューに複数の周辺装置が割り当て可能な情報処理装置の周辺装置制御方法であって、
周辺装置と通信を行い前記周辺装置のステータスを、前記１つのキューを単位に接続されたポート毎に取得して、オペレーティングシステムが管理するステータス保持部に保持するステータス管理工程と、
前記周辺装置を接続したポートを識別するポート識別子を含むスキーマを引数として、前記オペレーティングシステムが提供するインタフェースの所定の関数をコールすることにより、周辺装置のステータスを前記ステータス管理工程で管理しているステータスに対して問い合わせるステータス問合工程と、
前記ステータス問合工程で発行されたステータスの問い合わせメッセージに含まれる前記スキーマのポート識別子を認識するポート識別子認識工程と、
前記ポート識別子認識工程で認識したポート識別子に対応するポートに接続された周辺装置のステータスを、前記ステータス保持部から読み出して返信するステータス返信工程と
を有することを特徴とする周辺装置制御方法。
【請求項６】
前記ステータス問合工程で発行されたステータスの問い合わせメッセージに含まれる前記スキーマに前記ポート識別子が含まれているか否かを判定するポート識別子判定工程を更に有し、
前記ステータス返信工程では、前記ポート識別子判定工程でポート識別子が含まれてないと判定した場合に、予め決められたポートのステータスを返信することを特徴とする請求項５に記載の周辺装置制御方法。
【請求項７】
前記ステータス管理工程では、前記周辺装置のステータスを予め準備された記憶領域にポート単位で記録し、
前記ステータス問合工程では、前記ステータス管理工程で管理されているステータスへの問合せと、前記記憶領域にポート単位で書き込まれたステータスの取得とを、予め定められた条件で切り替えることを特徴とする請求項５又は６に記載の周辺装置制御方法。
【請求項８】
前記ステータス問合工程では、１つのキューに複数の周辺装置が割り当てられた場合に、通信メッセージに周辺装置が接続されているポートのポート識別子を含めることを特徴とする請求項５乃至７の何れか１項に記載の周辺装置制御方法。
【請求項９】
請求項５乃至８のいずれか１項に記載の周辺装置制御方法の各工程をコンピュータに実行させるためのプログラム。

【請求項１０】
請求項９に記載のプログラムを記憶したコンピュータ読取り可能な記憶媒体。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は周辺装置制御方法及びその情報処理装置と制御プログラムに関し、例えばプリンタなどの周辺装置を制御する周辺装置制御方法及びその情報処理装置と制御プログラムに関するものである。
【背景技術】
【０００２】
コンピュータなどの情報処理装置に接続されるインクジェットプリンタ、レーザービームプリンタなどの周辺装置の管理方法として、Ｗｉｎｄｏｗｓ（登録商標）２０００，ＸＰなどのＯＳでは、キューという概念が導入されている。キューに周辺装置を割り当て、Ｗｉｎｄｏｗｓ（登録商標）２０００，ＸＰ上のアプリケーションは、印刷ジョブをキューに送ることで所望の周辺装置で印刷が行えるようになっている。
【０００３】
又、キューにはＬａｎｇｕａｇｅ Ｍｏｎｉｔｏｒ（以下、ＬＭとも示す）という周辺装置と通信を行うモジュールが登録できる仕組みになっている。周辺装置のステータスを表示するアプリケーション（以下、ステータスモニタ）は、ＬＭと、レジストリやＰｒｉｎｔｉｎｇ ａｎｄ Ｐｒｉｎｔ Ｓｐｏｏｌｅｒ Ｉｎｔｅｒｆａｃｅｓを用いて通信を行い、周辺装置のステータスを表示している。例えば特許文献１にこのステータス取得技術が開示されている。ここで、Ｐｒｉｎｔｉｎｇ ａｎｄ Ｐｒｉｎｔ Ｓｐｏｏｌｅｒ Ｉｎｔｅｒｆａｃｅｓは、Ｍｉｃｒｏｓｏｆｔ Ｄｅｖｅｌｏｐｅｒ Ｎｅｔｗｏｒｋ（以下、ＭＳＤＮ）で公開されている通信ツールである。
【特許文献１】特開２００３－３０８１９４号公報
【発明の開示】
【発明が解決しようとする課題】
【０００４】
通常は１つのキューに１つの周辺装置が割り当てられるが、Ｗｉｎｄｏｗｓ（登録商標）２０００，ＸＰにはプリンタ・プールという機能がある。これを用いると、１つのキューに複数の周辺装置を割り当てることができる。すなわち、キューに送られた印刷ジョブは、複数の周辺装置のいずれか１つに送られる。これによりキューに送られた複数の印刷ジョブを同時に印刷することができる。
【０００５】
ところが、プリンタ・プール機能で１つのキューに複数の周辺装置が割り当てられた場合、ステータスモニタは、周辺装置の状態を正常にモニタできない。
【０００６】
例えば、ステータスモニタとＬＭがステータスを格納するレジストリを使用して通信を行っている場合、キューに割り当てられているレジストリ領域は１箇所である。その箇所は、”ＨＫＥＹ＿ＬＯＣＡＬ＿ＭＡＣＨＩＮＥ¥ＳＹＳＴＥＭ¥ＣｕｒｒｅｎｔＣｏｎｔｒｏｌＳｅｔ¥Ｃｏｎｔｒｏｌ¥Ｐｒｉｎｔ¥Ｐｒｉｎｔｅｒｓ¥ＰｒｉｎｔｅｒＡＢＣ¥ＰｒｉｎｔｅｒＤｒｉｖｅｒＤａｔａ”である。このため、レジストリに格納されている１番目の周辺装置の情報が２番目の周辺装置の情報で上書きされてしまうと、周辺装置の状態が正常にモニタできない。
【０００７】
又、Ｐｒｉｎｔｉｎｇ ａｎｄ Ｐｒｉｎｔ Ｓｐｏｏｌｅｒ Ｉｎｔｅｒｆａｃｅｓもキュー単位で通信を行う仕様である。ステータスモニタとＬＭがＰｒｉｎｔｉｎｇ ａｎｄ Ｐｒｉｎｔ Ｓｐｏｏｌｅｒ Ｉｎｔｅｒｆａｃｅｓを使って通信する時も、キューに複数の周辺装置が割り当てられている場合、ステータスモニタは自分がモニタする周辺装置を指定することができない。そのため、キューに割り当てられた全ての周辺装置の

状態を正常にモニタすることはできず、例えモニタできたとしても、ある１台の周辺装置のみだった。

【０００８】

本発明は、これらの問題点に着目し、キューに複数の周辺装置が割り当てられている場合においても、個々の周辺装置のステータスを正常に管理して、表示することができる周辺装置制御方法及びその情報処理装置と制御プログラムを提供することを目的とする。

【課題を解決するための手段】

【０００９】

上記課題を解決するために、本発明の情報処理装置は、周辺装置を管理する１つのキューに複数の周辺装置が割り当て可能な情報処理装置であって、周辺装置と通信を行い前記周辺装置のステータスを、前記１つのキューを単位に接続されたポート毎に取得して、オペレーティングシステムが管理するステータス保持部に保持するステータス管理手段と、前記周辺装置を接続したポートを識別するポート識別子を含むスキーマを引数として、前記オペレーティングシステムが提供するインタフェースの所定の関数をコールすることにより、周辺装置のステータスを前記ステータス管理手段へ問い合わせるステータス問合手段と、前記ステータス問合手段から発行されたステータスの問い合わせメッセージに含まれる前記スキーマのポート識別子を認識するポート識別子認識手段と、前記ポート識別子認識手段が認識したポート識別子に対応するポートに接続された周辺装置のステータスを、前記ステータス保持部から読み出して前記ステータス問合手段に返信するステータス返信手段とを有することを特徴とする。

【００１２】

又、周辺装置制御方法は、周辺装置を管理する１つのキューに複数の周辺装置が割り当て可能な情報処理装置の周辺装置制御方法であって、周辺装置と通信を行い前記周辺装置のステータスを、前記１つのキューを単位に接続されたポート毎に取得して、オペレーティングシステムが管理するステータス保持部に保持するステータス管理工程と、前記周辺装置を接続したポートを識別するポート識別子を含むスキーマを引数として、前記オペレーティングシステムが提供するインタフェースの所定の関数をコールすることにより、周辺装置のステータスを前記ステータス管理工程で管理しているステータスに対して問い合わせるステータス問合工程と、前記ステータス問合工程で発行されたステータスの問い合わせメッセージに含まれる前記スキーマのポート識別子を認識するポート識別子認識工程と、前記ポート識別子認識工程で認識したポート識別子に対応するポートに接続された周辺装置のステータスを、前記ステータス保持部から読み出して返信するステータス返信工程とを有することを特徴とする。

【００１５】

更に、上記周辺装置制御方法を実現するコンピュータ実行可能なプログラム、該プログラムをコンピュータ読取り可能な形態で記憶する記憶媒体を提供する。

【発明の効果】

【００１６】

以上説明したように、本発明によれば、キューに複数の周辺装置が割り当てられている場合においても、個々の周辺装置のステータスを正常に管理して、表示する周辺装置制御方法及びその情報処理装置と制御プログラムを提供することができる。

【発明を実施するための最良の形態】

【００１７】

以下、添付図面を参照して、本発明の実施形態例を詳細に説明する。尚、本実施形態では複数のプリンタを接続したパーソナルコンピュータ（以下、ＰＣと略す場合がある）の制御を例に説明するが、かかる例に限定されない。例えば、周辺機器はプリンタに限らず複写機、ファクシミリ、および印刷、スキャナ、ファックスの機能などを備える複合機なども含む他の周辺機器でも同様の効果を示す。又、情報処理装置としてＰＣを例としたが、例えばＤＶＤビデオプレーヤー、ゲーム、セットトップボックス、インターネット家電等、同様な使用方法が可能な任意の端末に対して実現することができ、同様に有効である

。これらも本発明に含まれる。

【００１８】

＜本実施形態の周辺装置制御システムの構成例＞

図１は、本実施形態に係る情報処理装置及び周辺装置からなる周辺装置制御システムをネットワーク環境下で実現した時のシステムの構成例を表すブロック図である。

【００１９】

図１において、７０２，７０３は情報処理装置であり、一般的なＰＣで構成される。ＰＣ７０２，７０３は、図３で後述するようなハードウェアで構成され、ＯＳ（Ｏｐｅｒａｔｉｎｇ　Ｓｙｓｔｅｍ）として、例えば米国マイクロソフト社のＷｉｎｄｏｗｓ（登録商標）ＸＰがインストールされている。無論ＯＳの種類はこれに限定されず、例えばＬｉｎｕｘ等でも適用可能であるが、以下では、米国マイクロソフト者のＷｉｎｄｏｗｓ（登録商標）を例に説明を行う。ＰＣ７０２とＰＣ７０３は、Ｅｔｈｅｒｎｅｔ（登録商標）で構成されるネットワーク７０１を介して接続され、互いに双方向通信が可能である。本実施形態の周辺装置制御システムでは、ＰＣ７０３がサーバとなり、ＰＣ７０２がクライアントとなるような関係を持っている。すなわち、ＰＣ７０３は、プリンタ７０５、７０７を共有プリンタとしてネットワーク７０１を介して他の情報処理装置から印刷することができるようなプリントサーバの機能を備えている。

【００２０】

図２Ａは、本実施形態に係る情報処理装置及び周辺装置からなる周辺装置制御システム部分の構成例を示すブロック図である。図２Ａにおいて、図１の対応する要素には同じ参照番号が付与されている。

【００２１】

図２Ａにおいて、ＰＣ７０３は図３で後述するようなハードウェアで構成され、ＯＳとして米国マイクロソフト社のＷｉｎｄｏｗｓ（登録商標）ＸＰがインストールされている。７０５，７０７はプリンタであり、カラーインクジェットプリンタで構成され、本実施形態における周辺装置である。本発明における周辺装置としては、プリンタ、複写機、ファクシミリ、またはこれらの複合機などの画像形成装置、スキャナ、デジタルカメラであってもよい。プリンタ７０５，７０６は、図４で後述するようなハードウェアで構成され、ＰＣ７０３とＵＳＢインタフェース７０４，７０６を介して接続されており、互いに双方向通信が可能である。ＵＳＢとはＵｎｉｖｅｒｓａｌ　Ｓｅｒｉａｌ　Ｂｕｓの略であり、双方向通信が可能な公知のインタフェースである。なお、以下では情報処理装置６０１の通信インタフェースとして、ＵＳＢインタフェースを例に説明を行うが、これに限定されない。例えば、無線通信に適用しても良いし、セントロニクスインタフェース、イーサーネットケーブルを介したＬＡＮインタフェース等を適用しても良い。

【００２２】

６０５は、Ｌａｎｇｕａｇｅ　Ｍｏｎｉｔｏｒ（以下、ＬＭとも示す）であり、Ｗｉｎｄｏｗｓ（登録商標）用のダイナミックリンクライブラリで構成される。ここで、ＬＭについて詳しく説明する。ＬＭは以下の２つの機能を備えたモジュールである。１つ目にステータス管理手段として機能する。具体的には、プリンタと双方向通信を行い、プリンタのステータスを取得し所定の場所に格納・保持させる。取得及び格納されたプリンタのステータスはＳｐｏｏｌｅｒを経由して他のソフトウェアからアクセス可能となっている。２つ目に、プリンタに印刷ジョブのデータを含む各種データを送信する機能を備え、送信すべきデータにプリンタの制御コマンドなどを付加する機能等も有する。６０２は、アプリケーションである。アプリケーション６０２の一例としては、プリンタ７０５，７０７の状態を表示するステータスモニタ等が挙げられる。以下、本例では、このステータスモニタを中心に説明する。

【００２３】

６０３はレジストリであり、特定の記憶領域が割当てられており、ＯＳの一部として管理される。アプリケーション６０２やＬＭ６０５は、レジストリ６０３に任意の情報を格納したり参照することができる。特に本実施形態では、このレジストリにプリンタの各種

ステータスが保持される。

【００２４】

６０４はＰｒｉｎｔｉｎｇ ａｎｄ Ｐｒｉｎｔ Ｓｐｏｏｌｅｒ Ｉｎｔｅｒｆａｃｅｓである。アプリケーション６０２はＰｒｉｎｔｉｎｇ ａｎｄ Ｐｒｉｎｔ Ｓｐｏｏｌｅｒ Ｉｎｔｅｒｆａｃｅｓ６０４を利用して、ＬＭ６０５に情報を送ったり、ＬＭ６０５から情報を受け取ったりすることができる。

【００２５】

図２Ｂは、本実施形態でのプリンタ・プール機能の概要を示す図である。

【００２６】

プリンタ・プール機能とは、ＰＣに接続された複数のプリンタを１つのキューで管理する技術である。図２Ｂでは、ＰＣ７０３に２台のプリンタ７０７，７０７が接続されていているが、これらの２台のプリンタは、１つのキューで管理される。

【００２７】

（ＰＣのハードウエア構成例）

図３は、ＰＣのハードウェア構成例を示すブロック図である。

【００２８】

図３において、ＰＣ７０２，７０３は、演算制御用の制御部であるＣＰＵ８０４により制御される。ＣＰＵ８０４は、ランダムアクセスメモリ部（ＲＡＭ８０１）を一時記憶部として使用し、記憶部であるハードディスクドライブ部（ＨＤＤ８０２）からＲＡＭ８０１にロードされたプログラムにしたがってＰＣを制御する。入力部の一例であるキーボード部（ＫＢＤ８０３）はデータ入力あるいは動作指示に使用され、表示部の一例である表示用ディスプレイ（ＬＣＤ８０６）はデータ表示や状態報知に使用される。通信制御部の一例であるネットワークボード（ＮＢ８０５）は、ネットワーク７０１を介する通信を行なう。インタフェース部（Ｉ／Ｆ部８０８）は、周辺機器（本例ではプリンタ）との接続を制御する。バス８０７が、以上のＰＣの構成要素を互いに接続する。

【００２９】

尚、上記ＲＡＭ８０１には、以下の第１実施形態で示すスプーラやＬＭがアクセスするポート情報管理用構造体やポート情報構造体記憶部、ステータスモニタがスプーラを介してＬＭとやり取りするスキーマの記憶領域が確保される。又、第２実施形態で示すレジストリの記憶領域も確保される。更に、以下で示すエクスポート関数の引数や、各フローで種畜されたポート名、ＬＭがプリンタから取得したステータスなどが一時記憶される領域も確保される。又、記憶部８０２は、可搬性ＣＤ－ＲＯＭまたは内部据付のＲＯＭ、メモリカードなどであってもよく、プリントする画像データなどの大容量のデータや本実施形態で使用されるプログラムが記憶される。

【００３０】

（プリンタのハードウエア構成例）

図４は、プリンタのハードウェア構成例を示すブロック図である。

【００３１】

図４において、プリンタ７０５，７０７は、インタフェース部（Ｉ／Ｆ部）９０２と、ＲＡＭ９０３と、ＲＯＭ９０４と、ＣＰＵ９０５と、エンジン９０６とを有する。

【００３２】

そして、Ｉ／Ｆ部９０２は、コンピュータ（ＰＣ）のＵＳＢインタフェース（Ｉ／Ｆ部８０８）に接続してある。インタフェース部９０２は、無線通信及び有線通信の双方の何れに適用しても良い。又、インタフェース部９０２は、セントロニクスインタフェースなどのホストとプリンタとが１対１で接続される形態や、ホストやプリンタとがイーサーネットケーブルを介してＬＡＮ接続される形態にも適用できる。

【００３３】

ＲＯＭ９０４は制御プログラム等がストアしてある。ＣＰＵ９０５は、ＲＯＭ９０４にストアしてある制御プログラムに従ってプリンタ９０１の各部を制御するものである。又、ＲＡＭ９０３は、ＣＰＵ９０５の主メモリとワークメモリとして用いら、受信したデー

タを一旦保存するための受信バッファを有する。エンジン９０６は、ＲＡＭ９０３に保存されたデータに基づきプリントを行うものである。尚、大容量の画像データを保持するためのＨＤＤなどを有してもよい。本実施形態における周辺装置のエンジン９０６としてはインクジェット方式を例に説明を行うが、これに限定されることなく、例えば、電子写真方式、熱転写方式等のエンジン（媒体への記録手段）等を適用することもできる。

【００３４】

＜本実施形態の周辺装置制御システムの動作例＞

図５は、本実施形態に係る情報処理装置及び周辺装置からなる周辺装置制御システムのソフトウェア機能例を示すブロック図である。尚、本実施形態では、プリンタのプログラムは主要な要素でないので示さないが、本実施形態に係る処理としては、ＬＭからのポーリングに対するステータスの返送や、プリント終了時のステータス返送などが関連する動作である。

【００３５】

図５において、００１はＰＣ７０３のソフトウエア構成例である。ここでは、本実施形態に関連のあるソフトウエアのみを示す。ＯＳ（Ｏｐｅｒａｔｉｎｇ　Ｓｙｓｔｅｍ）として米国マイクロソフト社のＷｉｎｄｏｗｓ（登録商標）ＸＰがインストールされている。

【００３６】

００２，００３は、周辺装置の状態をモニタするステータスモニタなどのアプリケーションである。ステータスモニタ００２は、プリンタ７０５の状態をＬＭ等の各種もモジュールを介してモニタ（監視）する。ステータスモニタ００３はＬＭ等の各種もモジュールを介してプリンタ７０７の状態をモニタ（監視）する。００４は、ＯＳやアプリケーションの情報を格納するレジストリである。これはＷｉｎｄｏｗｓ（登録商標）ＸＰの枠組みで提供される機能であり、このレジストリ中に複数のキュー及び各キューに関連付けられたポートの情報が書き込まれている（後述の図１４を参照）。００５，００６は、Ｐｒｉｎｔｉｎｇ　ａｎｄ　Ｐｒｉｎｔ　Ｓｐｏｏｌｅｒ　ＩｎｔｅｒｆａｃｅｓでこれもＷｉｎｄｏｗｓ（登録商標）ＸＰの枠組みで提供される機能である。００７は、Ｓｐｏｏｌｅｒで、Ｗｉｎｄｏｗｓ（登録商標）ＸＰＯＳの一部であり、キュー００８を管理する。００８は、キューで、印刷ジョブをキューイングする。ここでキューについて少し詳しく説明する。キューとは、文書作成ソフトウェア等のアプリケーションデータに基づく印刷データを管理するモジュールである。キューにはプリンタ（ポート）を複数割り当てることができる。また、文書作成ソフトウェア等のアプリケーションから印刷指示を行うと、ＰＣのハードディスク上の特定の場所に印刷データのファイルが作成され、キューはそのファイルを読み取り、印刷データの一覧を作成し表示する。更に、キューは印刷データを順番にプリンタに送信し印刷を行う。また、キューは印刷データの操作（一時停止、削除など）を行うことも可能となっている。

【００３７】

００９，０１０，０１１は、ＬＭで、データの通信Ｉ／Ｆとしてデータの送受信を制御する。ＬＭ００９，０１０，０１１は、モジュールとしてはこの内、ＬＭ００９とＬＭ０１０は、キュー００８からの印刷ジョブをプリンタに送信するときにＳｐｏｏｌｅｒ００７から呼ばれる。印刷ジョブがプリンタ７０５に送られるときはＬＭ００９が呼ばれる。印刷ジョブがプリンタ７０７に送られるときはＬＭ０１０が呼ばれる。ステータスモニタ００２，００３がＰｒｉｎｔｉｎｇ　ａｎｄ　Ｐｒｉｎｔ　Ｓｐｏｏｌｅｒ　Ｉｎｔｅｒｆａｃｅｓ００５，００６を使ってＬＭと通信する場合は、ＬＭ０１１が呼ばれる。尚、図５では上記ＬＭ００９，０１０，０１１を独立して記載しているが、これらは同じＬＭとして実行されてもよい。

【００３８】

０１２，０１３は、ポートモニタであり、ＬＭ００９，０１０から送られてくるデータをＵＳＢポートに対して送信したり、プリンタ７０５，７０７から送られてくるデータを受信したりする。ポートモニタとは、キューもしくはＬＭから受け取った印刷データをポ

ートに書き込んだり、ポートに接続されたプリンタからプリンタの状態を読み込んだりするモジュールである。ポートへの書き込み、読み込みを行う際にはＩ／Ｆ毎に決められた手順があり、ポートモニタは手順に従い書き込み、読み込みを行う。ポートとは、ＵＳＢなどのインタフェース毎に作成され、データを書き込み、読み込みを行う際の場所を示すものである。

【００３９】

７０４，７０６は、上述のＵｎｉｖｅｒｓａｌ Ｓｅｒｉａｌ Ｂｕｓ（以降ＵＳＢ）であり、双方向通信が可能な公知のインタフェースである。ポート番号として、ＵＳＢ７０４にはＵＳＢ００１が、ＵＳＢ７０７にはＵＳＢ００２が割り当てられている。７０５，７０７は、上述のカラーインクジェットプリンタで本実施形態における周辺装置である。

【００４０】

上述のように、ＰＣ７０３にはプリンタ７０５とプリンタ７０７が接続されている。キュー００８は、プリンタ・プール機能がＯＮになっていて、プリンタ７０５とプリンタ７０７とが割り当てられている。

【００４１】

ステータスモニタ００２は、レジストリ００４を使ってＬＭ００９と通信し、Ｐｒｉｎｔｉｎｇ ａｎｄ Ｐｒｉｎｔ Ｓｐｏｏｌｅｒ Ｉｎｔｅｒｆａｃｅ００５を使ってＬＭ０１１と通信して、プリンタ７０５の状態をモニタする。ステータスモニタ００３は、レジストリ００４を使ってＬＭ０１０と通信し、Ｐｒｉｎｔｉｎｇ ａｎｄ Ｐｒｉｎｔ Ｓｐｏｏｌｅｒ Ｉｎｔｅｒｆａｃｅｓ００６を使ってＬＭ０１１と通信して、プリンタ７０７の状態をモニタする。ここで、ＬＭ０１１はＰｒｉｎｔｉｎｇ ａｎｄ Ｐｒｉｎｔ Ｓｐｏｏｌｅｒ Ｉｎｔｅｒｆａｃｅｓ００５，００６を使ったプリンタの状態の問合わせに対する返信を行なうために起動される。プリンタ７０５，７０７の状態をモニタして、ステータス情報をレジストリ００４やスプーラ００７（ＬＭ０１１）が管理する後述のポート情報構造体に設定するのは、てＬＭ００９と０１０である。

【００４２】

図６は、ステータスモニタがＬＭとの通信方式を決定する処理例を示すフローチャートである。より具体的には、ステータス問合手段として機能するステータスモニタが、ＬＭ（ステータス管理手段）への問合せと、レジストリ（予め準備された記憶領域）にポート単位で書き込まれたステータスの取得とを、予め定められた条件で切り替える処理を示す。レジストリ（予め準備された記憶領域）にポート単位でのステータスを記録する処理については、第２実施形態にて詳しく説明する。

【００４３】

ステータスモニタは、通信方式の決定処理を開始（Ｓ６０１）し、まず発行先がＰＣ７０２か７０３かを判断し、現在の印刷がローカル印刷かどうかを判定する（Ｓ６０２）。例えば印刷要求に含まれるポート名にＰＣ７０３の名称が含まれていれば、ＰＣ７０２で稼動するステータスモニタは、図６のＳ６０２でＮｏと判断する。ローカル印刷とは、図５で示したような印刷ジョブを発行するＰＣとプリンタが接続されているＰＣが同一である印刷のことである。例えば図１を例に説明すると、ＰＣ７０３が印刷ジョブの発行元の場合はローカル印刷となす。また、ＰＣ７０２が印刷ジョブの発行元の場合はＰＣ７０３を経由して何れかのプリンタを利用するのでリモート印刷（ローカル印刷でない）と判断する。このリモート印刷のことを共有プリンタを用いた印刷と呼ぶこともある。

【００４４】

現在の印刷がローカル印刷の場合は、ＬＭとの通信にはレジストリ００４を使用する（Ｓ６０５）。Ｓ６０５の処理は後述する第２実施形態で詳しく説明する。一方、現在の印刷がローカル印刷でない場合は、現在のクライアントＰＣ７０２、サーバＰＣ７０３上のＯＳがレジストリ通信を行っても問題がないＯＳかどうかを判定する（Ｓ６０３）。この問題があるか否かの判定は、例えば印刷システムが稼動するＯＳの種類等により行われる。問題がない場合は、ＬＭとの通信にはレジストリ００４を使用する（Ｓ６０５）。問題

がある場合は、ＬＭとの通信にはＰｒｉｎｔｉｎｇ ａｎｄ Ｐｒｉｎｔ Ｓｐｏｏｌｅｒ Ｉｎｔｅｒｆａｃｅ００５，００６を使用する（Ｓ６０４）。このＰｒｉｎｔｉｎｇ ａｎｄ Ｐｒｉｎｔ Ｓｐｏｏｌｅｒ Ｉｎｔｅｒｆａｃｅ００５，００６は同一ＰＣ内、異なるＰＣ間に係わらず使用できる。

【００４５】

かかるＳ６０４又はＳ６０４の経路によりプリンタ単位のステータスを確実に取得し、オペレータに対して表示して知らせる（Ｓ６０６）。なお、上の図６の説明では、Ｓ６０３の処理を行うよう説明したが、必要がなければ、適宜Ｓ６０３の処理は省略するようにしても良い。また、図６の説明では、ＬＭ（ステータス管理手段）への問合せと、レジストリ（予め準備された記憶領域）にポート単位で書き込まれたステータスの取得とを、切り替える条件として、Ｓ６０２、Ｓ６０３を例に説明したが、これに限定されない。例えば、他の予め定められた条件を適用するようにしても良い。

【００４６】

図７は、プリンタ・プール機能をＯＮにして印刷を行った場合のキューの状態を示した図である。後述の図１３のフローチャートの処理によりステータスモニタにより表示されるユーザインタフェースを示す。

【００４７】

１つのキュー１０１の名前は”Ｐｒｉｎｔｅｒ ＡＢＣ”であり、プリンタ７０５とプリンタ７０７とが割り当てられている。１つ目のジョブ１０２はＵＳＢ００１（７０４）に接続されたプリンタ７０５に送信されている。２つ目のジョブ１０３はＵＳＢ００２（７０６）に接続されたプリンタ７０７に送信されている。このようにプリンタ・プール機能をＯＮにして印刷を行った場合、一度に複数の印刷ジョブを処理することができる。

【００４８】

＜第１実施形態のステータスモニタ処理の動作例＞

第１実施形態では、図６のＳ６０４の処理によって、ＬＭとの通信にＰｒｉｎｔｉｎｇ ａｎｄ Ｐｒｉｎｔ Ｓｐｏｏｌｅｒ Ｉｎｔｅｒｆａｃｅｓが選択された場合を想定している。尚、第１実施形態は、図５で説明したソフトウエアで構成されるが、ステータスモニタとＬＭの通信はＰｒｉｎｔｉｎｇ ａｎｄ Ｐｒｉｎｔ Ｓｐｏｏｌｅｒ Ｉｎｔｅｒｆａｃｅｓ００５，００６を用いて実施されるものとする。

【００４９】

（ポート情報の管理構成例）

図８は、Ｓｐｏｏｌｅｒ００７とＬＭがポート情報をどのように管理しているかの例を示す図であり、ポート毎のステータスを保持する保持部を示す。尚、以下のポート情報はＳｐｏｏｌｅｒ００７が管理し、ポート情報の作成及び書込みはＬＭ００９，０１０が行ない、ポート情報のステータスモニタからの問合わせへの返信はＬＭ０１１が行なう。なお、図８の格納場所は、ＬＭが独自に作成した格納場所でも良いし、ＯＳが提供する予め定められた格納場所でも良いし、ＬＭ及びＯＳが共有する格納場所でも良い。

【００５０】

２０２と２０３は、ポート情報構造体で、プリンタが接続されている各ポートの情報を格納し、Ｓｐｏｏｌｅｒ００７がＬＭ０９０及び０１０のエクスポート関数であるＯｐｅｎＰｏｒｔＥｘ（）を呼ぶ毎に、各ＬＭにより接続ポート毎に作成される。すなわち、ＰＣ７０３が起動して各ポートに対応するＬＭ００９，０１０が起動された後、各ＬＭ００９，０１０に対してＳｐｏｏｌｅｒ００７からポート初期化に相当するＯｐｅｎＰｏｒｔＥｘ（）を呼んだ時に、各ポート情報構造体が作成される。図５における、ＬＭ００９，０１０が、ポートに接続されたプリンタから取得した紙無しエラーなどのステータス情報もポート情報構造体に格納される。また、後述の図１１で説明するポート名等のポート識別子もこのポート情報に含まれる。これら、ＬＭ００９，０１０の各々がプリンタ７０５、７０７からステータス情報を取得し、更新することによって、上に説明した図８のポート毎のステータス情報が更新される。

【００５１】

上記ポート情報構造体の管理は、ポート情報管理用構造体２０１を用いて行われる。ポート情報管理用構造体２０１は、Ｓｐｏｏｌｅｒ００７がＬＭのエクスポート関数であるＩｎｉｔｉａｌｉｚｅＰｒｉｎｔＭｏｎｉｔｏｒ２（）を呼んだときに、ＬＭによって作成される。すなわち、ＰＣ７０３が起動して各ポートに対応するＬＭ００９，０１０が起動された後、Ｓｐｏｏｌｅｒ００７がＬＭ００９，０１０いずれかのＩｎｉｔｉａｌｉｚｅＰｒｉｎｔＭｏｎｉｔｏｒ２（）を呼んだときに、ポート情報管理用構造体２０１が作成される。

【００５２】

ポート情報管理用構造体のメンバ変数のｐＦｉｒｓｔＰｏｒｔには１番目のポート情報構造体２０２（ここではＵＳＢ００２ポートのポート情報構造体）へのポインタが格納されている。１番目のポート情報構造体のメンバ変数ｐＮｅｘｔは２番目のポート情報構造体２０３（ここではＵＳＢ００１ポートのポート情報構造体）へのポインタが格納されている。ポート情報構造体にはポート情報管理用構造体のアドレスも格納されており、ポート情報構造体からポート情報管理用構造体を特定することにより全てのポートのポート情報構造体を参照することが可能になっている。尚、ＯｐｅｎＰｏｒｔＥｘ（）、ＩｎｉｔｉａｌｉｚｅＰｒｉｎｔＭｏｎｉｔｏｒ２（）は、上記ＭＳＤＮで公開されている既知の情報である。

【００５３】

以下、上記ポート情報管理用構造体及びポート情報構造体の作成手順例を示す。

【００５４】

（ポート情報管理用構造体の作成手順例）

図９は、ＬＭ００９又は０１０がエクスポート関数ＩｎｉｔｉａｌｉｚｅＰｒｉｎｔＭｏｎｉｔｏｒ２（）でポート情報管理用構造体２０１を作成する処理例を示すフローチャートである。

【００５５】

Ｓｐｏｏｌｅｒ００７がＯＳやアプリケーションからプリントモニタの初期化を指示されて、ＬＭ００９又は０１０のエクスポート関数であるＩｎｉｔｉａｌｉｚｅＰｒｉｎｔＭｏｎｉｔｏｒ２（）をコールする（Ｓ７０１）する。ＬＭ００９又は０１０がＩｎｉｔｉａｌｉｚｅＰｒｉｎｔＭｏｎｉｔｏｒ２（）の処理を開始する（Ｓ７０２）。ＬＭは、ポート情報管理用構造体２０１を作成する（Ｓ７０３）。作成したポート情報管理用構造体２０１をＩｎｉｔｉａｌｉｚｅＰｒｉｎｔＭｏｎｉｔｏｒ２（）の第２引数のｐｈＭｏｎｉｔｏｒにセットする。それを、ｍｏｎｉｔｏｒ　ｈａｎｄｌｅとしてＳｐｏｏｌｅｒ００７に返却し、ＬＭの処理を終了する（Ｓ７０４）。

【００５６】

Ｓｐｏｏｌｅｒ００７はＩｎｉｔｉａｌｉｚｅＰｒｉｎｔＭｏｎｉｔｏｒ２（）の第２引数を通して、ｍｏｎｉｔｏｒ　ｈａｎｄｌｅとしてポート情報管理用構造体２０１を受け取る。そして、ＩｎｉｔｉａｌｉｚｅＰｒｉｎｔＭｏｎｉｔｏｒ２（）のコールを終了する（Ｓ７０５）。

【００５７】

（ポート情報構造体の作成手順例）

図１０は、ＬＭ０９０又は０１０がエクスポート関数ＯｐｅｎＰｏｒｔＥｘ（）でポート情報構造体２０２又は２０３を作成する処理を表すフローチャートである。

【００５８】

Ｓｐｏｏｌｅｒ００７がＯＳやアプリケーションからポートの初期化を指示されて、、ポートに対応するＬＭ０９０又は０１０のエクスポート関数であるＯｐｅｎＰｏｒｔＥｘ（）をコールする（Ｓ８０１）。ＬＭ０９０又は０１０がＯｐｅｎＰｏｒｔＥｘ（）の処理を開始する（Ｓ８０２）。ＬＭはＯｐｅｎＰｏｒｔＥｘ（）の第２引数として渡されるポート情報管理用構造体２０１を取得する（Ｓ８０３）。このポート情報管理用構造体２０１は、図９のＳ７０３で作成したものである。ＬＭは、ポート情報構造体２０２又は２０３を作成（Ｓ８０４）する。

【００５９】
次に、作成したポート情報構造体をリンクする。例えば図８のようにポート情報構造体２０３が既に作成されている場合に、ポート情報構造体２０２を作成する場合を示す。この場合は、ＬＭ００９が、Ｓ８０４で新たに作成されたポート情報構造体２０２のメンバ変数ｐＮｅｘｔに、ポート情報管理用構造体２０１のメンバ変数ｐＦｉｒｓｔＰｏｒｔで指定されているポート情報構造体２０３のアドレスを格納する（Ｓ８０５）。さらに、ポート情報管理用構造体２０１のメンバ変数ｐＦｉｒｓｔＰｏｒｔに、Ｓ８０４で新たに作成されたポート情報構造体２０２のアドレスを格納する（Ｓ８０６）。ポート情報構造体２０２のメンバ変数にポート情報管理用構造体２０１のアドレスを格納する（Ｓ８０７）。ポート情報構造体２０２をＯｐｅｎＰｏｒｔＥｘ（）の第５引数ｐＨａｎｄｌｅにセットして、ｐｏｒｔ　ｈａｎｄｌｅとしてＳｐｏｏｌｅｒ００７に返却し、ＬＭの処理を終了する（Ｓ８０８）。Ｓｐｏｏｌｅｒ００７はＯｐｅｎＰｏｒｔＥｘ（）の第５引数を通して、ｐｏｒｔ　ｈａｎｄｌｅとしてポート情報構造体を受け取りＯｐｅｎＰｏｒｔＥｘ（）のコールを終了する（Ｓ８０９）。

【００６０】
（プリンタのステータス情報の通信構成例）
図１１は、ステータスモニタとＬＭ０１１がＰｒｉｎｔｉｎｇ　ａｎｄ　Ｐｒｉｎｔ　Ｓｐｏｏｌｅｒ　Ｉｎｔｅｒｆａｃｅｓ００５，００６を用いてプリンタのステータス情報に関する通信を行うときに使われるスキーマである。スキーマはＸＭＬ等のマークアップ言語で記述されている。スキーマは、ステータスモニタとＬＭ０１１が通信を行う際にステータスモニタによって作成される。ここで、スキーマとは一般的には、データベース全体の構造のことまたは、それらを記述したファイルを意味するが、ここでは周辺装置の状態を表現する表現方法・形式のこと意味するものとする。

【００６１】
定義名はＳｔａｔｕｓ、ＮｏｄｅＴｙｐｅはＶａｌｕｅ、スキーマのフルパスは、￥Ｐｒｉｎｔｅｒ．Ｉｎｆｏｒｍａｔｉｏｎ．＜ＰｏｒｔＮａｍｅ＞：Ｓｔａｔｕｓ、Ｄａｔａ　ｔｙｐｅはＢｉｎａｒｙ形式、である。＜ＰｏｒｔＮａｍｅ＞には、プリンタが接続されているポート識別子が入る。実施例１ではポート識別子として、ポート名を用いる。例えば、ＵＳＢ００１に接続されたプリンタのステータス情報を取得する場合は、スキーマのフルパスは、￥Ｐｒｉｎｔｅｒ．Ｉｎｆｏｒｍａｔｉｏｎ．ＵＳＢ００１：Ｓｔａｔｕｓとなる。プリンタのステータス情報１１０１はＢｉｎａｒｙ形式で格納される。

【００６２】
図１２は、Ｂｉｎａｒｙ形式で格納されるプリンタのステータス情報１１０１の一例である。

【００６３】
図１２の例では、プリンタのデバイスＩＤ、インクが少なくなっているなどの警告情報、紙無しエラーなどのエラー情報、インク残量、印刷中のページ情報、印刷済みのページ情報、などが格納されているが、これに限定されない。

【００６４】
（プリンタのステータス情報の通信手順例）
図１３は、ステータスモニタがＰｒｉｎｔｉｎｇ　ａｎｄ　Ｐｒｉｎｔ　Ｓｐｏｏｌｅｒ　Ｉｎｔｅｒｆａｃｅｓを用いてプリンタのステータス情報を取得する処理を表すフローチャートである。

【００６５】
図１３で、ステータスモニタは、印刷ジョブやオペレータからの問い合せなどのアプリケーションやＯＳの要求に従って、プリンタステータスの取得処理を開始する（Ｓ３０１）。１つのキューに複数の周辺装置（ポート）が割り当てられた場合、つまりプリンタ・プールが実行されている場合に印刷ジョブやオペレータからの問い合せなどに基づいてプリンタのポート名を取得する（Ｓ３０２）。一方、プリンタ・プールが指定されていない場合にはＳ３０２でポートの取得は行わず、ＬＭに対してポート識別子を含まないスキー

マを発行する。この場合にはＳ３０７でＮＯと判定される。取得したポート名を元に、Ｐｒｉｎｔｉｎｇ ａｎｄ Ｐｒｉｎｔ Ｓｐｏｏｌｅｒ Ｉｎｔｅｒｆａｃｅｓ００５，００６で使うスキーマのパスを作成（Ｓ３０３）する。作成したスキーマを引数にＰｒｉｎｔｉｎｇ ａｎｄ Ｐｒｉｎｔ Ｓｐｏｏｌｅｒ Ｉｎｔｅｒｆａｃｅｓの関数－ＩＢｉｄｉＳｐｌ：：ＳｅｎｄＲｅｃｖ（）をコールする（Ｓ３０３）。該ＳｅｎｄＲｅｃｖ（）のコールに基づき、Ｓｐｏｏｌｅｒ００７からＬＭ０１１のエクスポート関数ＳｅｎｄＲｅｃｖＢｉｄｉＤａｔａＦｒｏｍＰｏｒｔ（）が呼ばれる（Ｓ３０５）。

【００６６】

Ｓｐｏｏｌｅｒ００７からはＳｅｎｄＲｅｃｖＢｉｄｉＤａｔａＦｒｏｍＰｏｒｔ（）の引数として、ステータスモニタがＳ３０３で作成したポートに対応するスキーマ及びポート情報構造体が、ＬＭ０１１に渡される。ポート情報構造体は、Ｓｐｏｏｌｅｒ００７がＬＭ０９０又は０１０のエクスポート関数ＯｐｅｎＰｏｒｔ（）をコールした際、ＬＭ０９０又は０１０が作成したものである。

【００６７】

キュー００８にはポートが２つ割り当てられている（ＵＳＢ００１とＵＳＢ００２）が、渡されるポート情報構造体は１つだけである。ステータスモニタがスキーマにポート識別子を指定している場合は、指定されたポート情報構造体が渡されるが、指定がなければどちらのポートのポート情報構造体が渡されるかはＳｐｏｏｌｅｒの仕様に依存する。

【００６８】

ＳｅｎｄＲｅｃｖＢｉｄｉＤａｔａＦｒｏｍＰｏｒｔ（）で引数として渡されたスキーマを取得し（Ｓ３０６）、スキーマにポート識別子が含まれているかどうかを確認する（Ｓ３０７）。ポート識別子が含まれている場合は、図８で説明した方法で管理されているポート情報構造体からポート識別子と一致するポート情報構造体を取得する（Ｓ３０８）。より具体的には、図８のポート情報に含まれるポート識別子と、図１１のスキーマに含まれるポート識別子（＜ＰｏｒｔＮａｍｅ＞）と、の一致により、参照すべきポート情報を特定する。該特定されたポート情報中に保持されるステータスを読込み、ステータス要求元に返信する。

【００６９】

ポート情報構造体の取得に成功した場合は（Ｓ３０９）、Ｓ３１１に進む。
ポート識別子が含まれていない場合は（Ｓ３０９）、Ｓｐｏｏｌｅｒ００７から渡されたポート情報構造体（従来から知られている予め決められたポートのステータス）を使い（Ｓ３１０）、Ｓ３１１に進む。

【００７０】

選別されたポート情報構造体に保存されているプリンタのステータスを取得し（Ｓ３１１）、取得したステータスをスキーマの定義に変換する（Ｓ３１２）。ＳｅｎｄＲｅｃｖＢｉｄｉＤａｔａＦｒｏｍＰｏｒｔ（）関数の引数としてスキーマの定義に変換されたプリンタのステータスをＳｐｏｏｌｅｒ００７を介してステータスモニタに返す。そして、ＳｅｎｄＲｅｃｖＢｉｄｉＤａｔａＦｒｏｍＰｏｒｔ（）関数の処理を終了する（Ｓ３１３）。

【００７１】

ステータスモニタはステータスを取得し、必要であれば表示する（Ｓ３１４）。

【００７２】

＜第２実施形態のステータスモニタ処理の動作例＞

第２実施形態も、図５で説明したソフトウエアで構成されが、ステータスモニタとＬＭの通信は、図６のＳ６０５の処理によってレジストリ００４を経由する場合を想定している。

【００７３】

ステータスモニタとＬＭとの通信にレジストリ００４を使う理由は以下の理由である。Ｗｉｎｄｏｗｓ（登録商標）ＸＰ、２０００などのＯＳではキュー毎にレジストリの特定の場所が割り当てられていて、そこにアクセスするＧｅｔＰｒｉｎｔｅｒＤａｔａＥｘ（

）やＳｅｔＰｒｉｎｔｅｒＤａｔａＥｘ（）などのＡＰＩも用意されている。いわば、レジストリの使用は、ＯＳが推奨する通信方法の１つである。従って、以下のような利点がある。すなわち、この方法を使うことにより将来ＯＳの仕様が変更された場合でも、推奨の通信方法を使っている場合は仕様の差異をＯＳが吸収してくれたり、他のモジュールやアプリケーションがＡＰＩで容易にレジストリに書かれた情報を参照できる。

【００７４】

図１４は、プリンタのステータスがレジストリ００４のどの階層に記録されるかを示している。

【００７５】

”ＨＫＥＹ＿ＬＯＣＡＬ＿ＭＡＣＨＩＮＥ¥ＳＹＳＴＥＭ¥ＣｕｒｒｅｎｔＣｏｎｔｒｏｌＳｅｔ¥Ｃｏｎｔｒｏｌ¥Ｐｒｉｎｔ¥Ｐｒｉｎｔｅｒｓ”の階層に、キュー毎に一意のキーが作成される。ここでは、キュー００８用に”Ｐｒｉｎｔｅｒ　ＡＢＣ”というキーが作成されている。”Ｐｒｉｎｔｅｒ　ＡＢＣ”の下位の階層に、”ＰｒｉｎｔｅｒＤｒｉｖｅｒＤａｔａ”が作られ、その下位の層にポート毎に”ＵＳＢ００１”、”ＵＳＢ００２”というキーが作成されている。また、仮に別のキュー（例えば”Ｐｒｉｎｔｅｒ　ＥＤＦ”に対応のキュー）が存在すれば、図１４中には、”Ｐｒｉｎｔｅｒ　ＡＢＣ”と並列に、”Ｐｒｉｎｔｅｒ　ＥＤＦ”に係わる情報が記録される。

【００７６】

ＬＭ００９又は０１０は、プリンタのステータスをポート毎に記録する。ＵＳＢ００１に接続されているプリンタ７０５のステータスは”ＵＳＢ００１”の階層に記録する。ＵＳＢ００２に接続されているプリンタ７０７のステータスは”ＵＳＢ００２”の階層に記録する。

【００７７】

図１５は、レジストリ００４に記録されるプリンタのステータス情報の一例である。

【００７８】

値”Ｐｒｉｎｔｅｒ＿Ｓｔａｔｕｓ＿Ｃａｒｔｒｉｄｇｅ”は、プリンタに搭載されているカートリッジ情報であり、値の種類は”ＲＥＧ＿ＳＺ型”という文字列であり、中身には”Ｃｏｌｏｒ”という文字列が入っている。この情報から、プリンタにはＣｏｌｏｒカートリッジが搭載されていることがわかる。

【００７９】

値”Ｐｒｉｎｔｅｒ＿Ｓｔａｔｕｓ＿Ｅｒｒｏｒ”は、プリンタに発生しているエラー情報であり、値の種類は”ＲＥＧ＿ＳＺ型”という文字列であり、中身には”Ｎｏ”という文字列が入っている。この情報からプリンタにはエラーが発生していないことがわかる。

【００８０】

値”Ｐｒｉｎｔｅｒ＿Ｓｔａｔｕｓ＿Ｗａｒｎｉｎｇ”は、プリンタに発生している警告情報であり、値の種類は”ＲＥＧ＿ＤＷＯＲＤ型”というＤＷＯＲＤ値であり、中身は”５００”となっている。この情報からプリンタには警告コード５００という警告が発生していることがわかる。

【００８１】

値”Ｐｒｉｎｔｅｒ＿Ｓｔａｔｕｓ＿Ｉｎｋ＿Ｃｏｌｏｒ”は、プリンタのＣｏｌｏｒインクの残量情報であり、値の種類は”ＲＥＧ＿ＤＷＯＲＤ型”というＤＷＯＲＤ値であり、中身は”７０”となっている。この情報からプリンタのＣｏｌｏｒインクの残量は７０であることがわかる。

【００８２】

尚、プリンタのステータス情報は上記例に限定されない。

【００８３】

（ステータスモニタのステータス取得手順例）

図１６は、ステータスモニタがレジストリ００４を用いてプリンタのステータスを取得する方法を示すフローチャートである。

【００８４】
ステータスモニタが、印刷ジョブやオペレータからの問い合せなどのアプリケーションやＯＳの要求に従って、ステータスの取得を開始する（Ｓ４０１）。印刷ジョブやオペレータからの問い合せなどのアプリケーションから、ステータスを取得するプリンタのポート名を取得する（Ｓ４０２）。取得したポート名を元に、レジストリの階層を指定するパスを作成する。例えば、ＵＳＢ００１に接続されているプリンタ７０５のステータスを取得する場合のパスは、以下のようになる。
【００８５】
”ＨＫＥＹ＿ＬＯＣＡＬ＿ＭＡＣＨＩＮＥ¥ＳＹＳＴＥＭ¥ＣｕｒｒｅｎｔＣｏｎｔｒｏｌＳｅｔ¥Ｃｏｎｔｒｏｌ¥Ｐｒｉｎｔ¥Ｐｒｉｎｔｅｒｓ¥Ｐｒｉｎｔｅｒ　ＡＢＣ¥ＰｒｉｎｔｅｒＤｒｉｖｅｒＤａｔａ¥ＵＳＢ００１”
ステータスモニタは、作成したパスを指定してレジストリ００４にアクセスし（Ｓ４０３）、プリンタのステータスを取得し（Ｓ４０４）、終了する（Ｓ４０５）。パスには、キューとポートとの指定が含まれている。
【００８６】
（レジストリへのステータス格納手順例）
図１７は、ＬＭ００９又は０１０がレジストリ００４にプリンタのステータスを格納する方法を示すフローチャートである。
【００８７】
ＬＭ００９又は０１０は、定期的な（例えば４秒に１回）ポーリングや印刷処理の終了などで処理を開始する（Ｓ５０１）。上記ポーリングやプリンタからのイベント情報により、レジストリ００４にステータスを保存するプリンタのポート名を取得する（Ｓ５０２）。取得したポート名を元に、レジストリ００４の階層を指定するパスを作成する。例えば、ＵＳＢ００１に接続されているプリンタ０１６のステータスを保存する場合のパスは、以下のようになる。
【００８８】
”ＨＫＥＹ＿ＬＯＣＡＬ＿ＭＡＣＨＩＮＥ¥ＳＹＳＴＥＭ¥ＣｕｒｒｅｎｔＣｏｎｔｒｏｌＳｅｔ¥Ｃｏｎｔｒｏｌ¥Ｐｒｉｎｔ¥Ｐｒｉｎｔｅｒｓ¥Ｐｒｉｎｔｅｒ　ＡＢＣ¥ＰｒｉｎｔｅｒＤｒｉｖｅｒＤａｔａ¥ＵＳＢ００１”
作成したパスでレジストリ００４にアクセスし（Ｓ５０３）、プリンタから取得したステータスを保存し（Ｓ５０４）、終了する（Ｓ５０５）。
【００８９】
＜他の実施形態＞
上記実施形態では、アプリケーションの一例としてステータスモニタ００２，００３を挙げたが、この例に限られることはない。例えば周辺装置から情報を取得して、表示を行う任意のアプリケーションで実現可能であり、本発明はその場合でも有効である。
又、上記実施形態では、情報処理装置としてパーソナルコンピュータを想定したが、この例に限られることはない。例えばＤＶＤビデオプレーヤー、ゲーム、セットトップボックス、インターネット家電等、同様な使用方法が可能な任意の端末に対して実現することができ、本発明はその場合でも有効である。
【００９０】
又、上記実施形態では、周辺装置としてプリンタを例示しているが、周辺装置として他に、複写機、ファクシミリ、および印刷、スキャナ、ファックスの機能などを備える複合機などのいずれかが、本発明の適用対象となり得る。
【００９１】
又、上記実施形態では、ＯＳに例としてＷｉｎｄｏｗｓ（登録商標）ＸＰ、２０００を使用したが、これらのＯＳに限られることなく、任意のＯＳを使用することができる。
【００９２】
又、上記実施形態では、ＰＣ００１とプリンタ０１６、プリンタ０１７との間のインタフェースとして、ＵＳＢインタフェースを用いたが、このインタフェースに限られない。

例えば、Ｅｔｈｅｒｎｅｔ（登録商標）、無線ＬＡＮ、ＩＥＥＥ１３９４、Ｂｌｕｅｔｏｏｔｈ、ＩｒＤＡ、パラレル、シリアル等の任意のインタフェースを用いるようにしてもよい。

【００９３】

又、上記実施形態では、ＬＭとポートモニタを分けて実装している形態を示したが、ＬＭの機能をポートモニタに含めて実装しても実現可能であり、本発明はその場合でも有効である。

【００９４】

又、本発明の目的は、前述した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、システム或いは装置に供給する。そして、そのシステム或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、達成されることは言うまでもない。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、プログラムコード自体及び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。

【００９５】

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、フレキシブルディスク、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等を用いることができる。

【００９６】

又、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、前述した実施形態の機能が実現されるだけでない。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などが実際の処理の一部又は全部を行い、その処理によって前述した第１の実施形態や第２の実施の形態の機能が実現される場合も含まれることは言うまでもない。

【００９７】

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込まれる。その後、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等が実際の処理の一部又は全部を行い、その処理によって前述した実施形態の機能が実現される場合も含まれることは言うまでもない。

【００９８】

以上説明したように、本実施形態によれば、キューに複数の周辺装置が割り当てられている場合においても、個々の周辺装置のステータスを正常に、管理して表示することができる。

【図面の簡単な説明】

【００９９】

【図１】本実施形態に係る情報処理装置及び周辺装置からなる周辺装置制御システムをネットワーク環境下で実現した時のシステムの構成例を表すブロック図である。

【図２Ａ】本実施形態に係る情報処理装置及び周辺装置からなる周辺装置制御システム部分の構成例を示すブロック図である。

【図２Ｂ】本実施形態に係るプリンタ・プール機能の概要例を説明する図である。

【図３】本実施形態に係る情報処理装置であるパーソナルコンピュータ（ＰＣ）の構成例を説明するブロック図である。

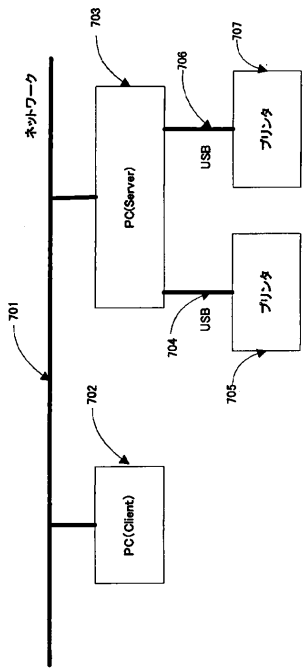
【図４】本実施形態に係る周辺装置であるプリンタの構成例を説明するブロック図である。

【図５】本実施形態に関わる情報処理装置及び周辺装置からなる周辺装置制御システムのソフトウェア機能例のブロック図である。

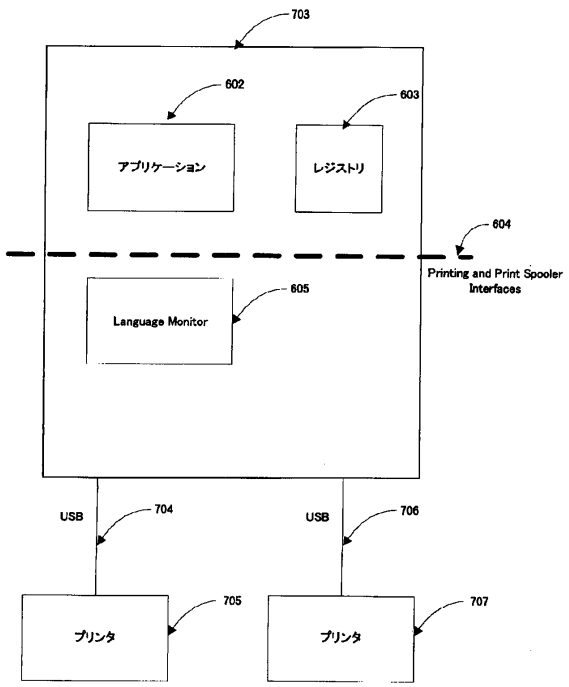
【図６】本実施形態のステータスモニタがどの方法でＬＭと通信を行うかを決定する処理例を表すフローチャートである。

【図７】本実施形態の複数の周辺装置が割り当てられたときのキューの表示例を示す図である。
【図８】第１実施形態でＬＭがポート情報を管理する方法を表す図である。
【図９】第１実施形態でＬＭがＩｎｉｔｉａｌｉｚｅＰｒｉｎｔＭｏｎｉｔｏｒ２（）でポート情報管理用構造体を作成する処理を表すフローチャートである。
【図１０】第１実施形態でＬＭがＯｐｅｎＰｏｒｔＥｘ（）でポート情報構造体を作成する処理を表すフローチャートである。
【図１１】第１実施形態でステータスモニタがプリンタのステータスをＬＭに要求する際に使用するスキーマ例を示す図である。
【図１２】図１１のスキーマ例で取得できるプリンタのステータスが格納されたバイナリデータ例を示す概念図である。
【図１３】第１実施形態でステータスモニタとＬＭがスプーラインタフェースを用いて通信を行い、キューに複数割り当てられたプリンタのステータスを表示する処理手順例を示すフローチャートである。
【図１４】第２実施形態でレジストリ内のプリンタステータスの格納場所を示す図である。
【図１５】図１４のレジストリ内に格納されるプリンタのステータスの一例を示す図である。
【図１６】第２実施形態でステータスモニタがレジストリに格納されているプリンタステータスを取得する処理手順例を示すフローチャートである。
【図１７】第２実施形態でＬＭが取得したプリンタのステータスをレジストリに格納する処理手順例を示すフローチャートである。

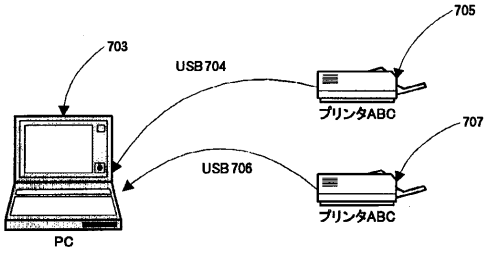
【図１】



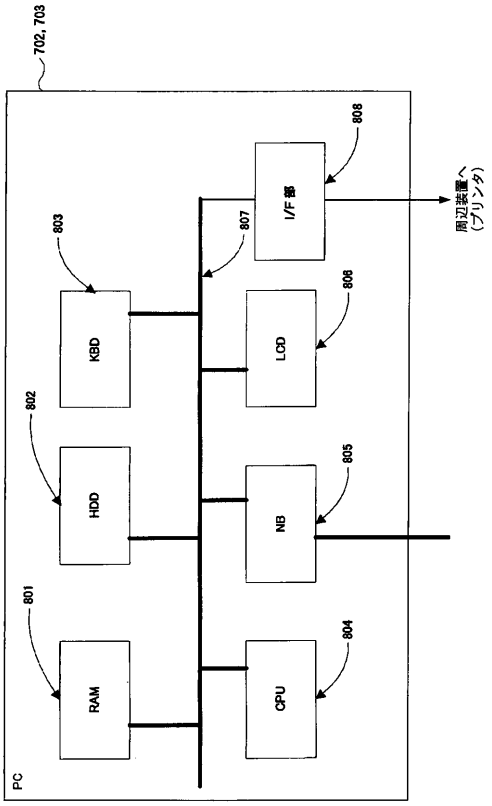
【図２Ａ】



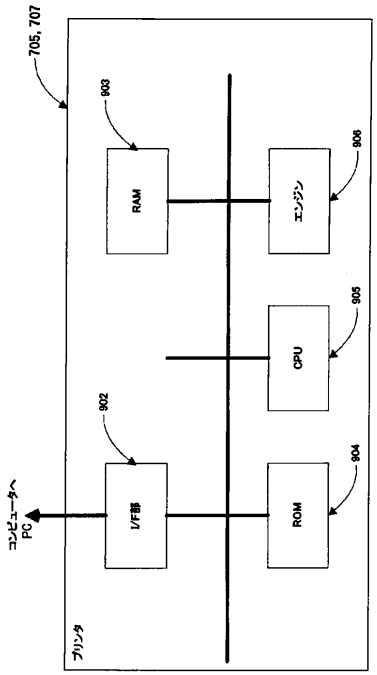
【図２Ｂ】


703
USB704
705
プリンタABC
USB706
707
プリンタABC
PC

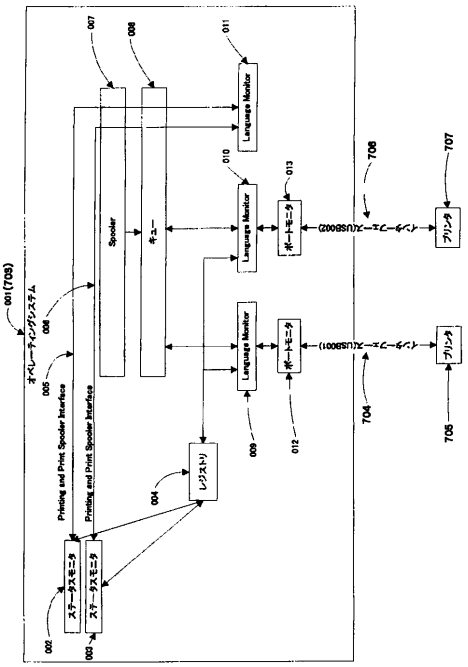
【図３】


702, 703
PC
CPU
RAM
HDD
KBD
NB
LCD
I/F部
804
801
802
803
805
806
807
808
周辺装置へ
(プリンタ)

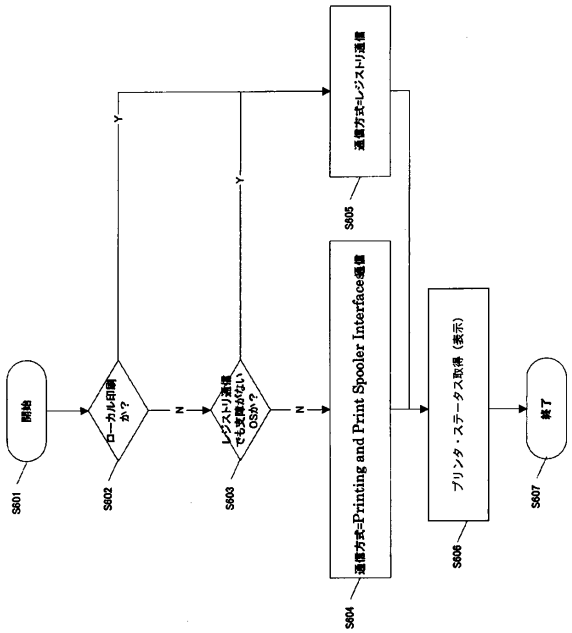
【図４】


705, 707
プリンタ
RAM
CPU
ROM
I/F部
エンジン
902
903
904
905
906
コンピューターへ
PC

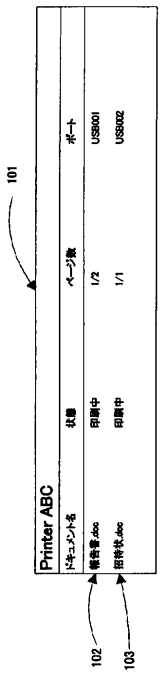
【図５】


Spooler
Printing and Print Spooler Interface
Language Monitor
ポートモニタ
プリンタ
レジストリ
704
705
707
708

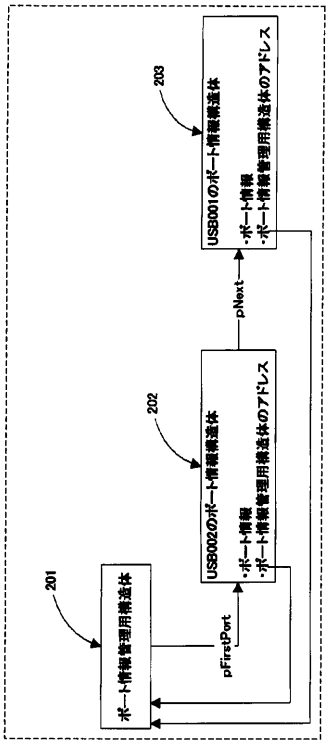
【図６】


開始
S601
ローカル印刷か？
S602
N
Y
レジストリ通信でも実現がないOSか？
S603
N
Y
通信方式=Printing and Print Spooler Interface通信
S604
通信方式=レジストリ通信
S605
プリンタ・ステータス取得（表示）
S606
終了
S607

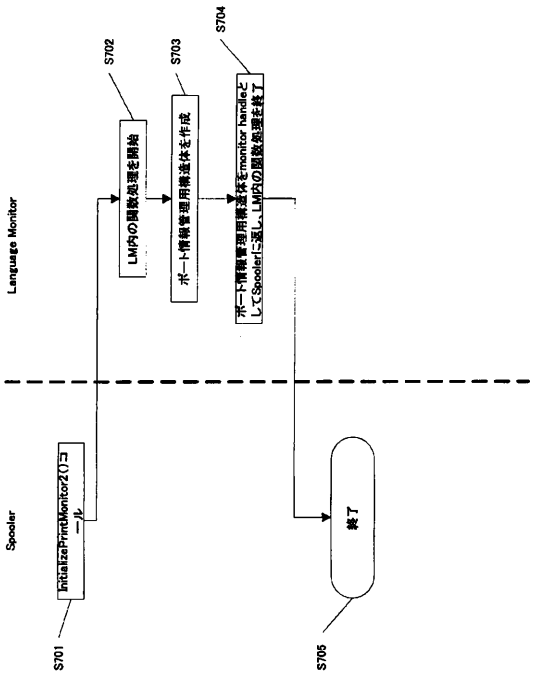
【図７】


101
Printer ABC
ドキュメント名
状態
ページ数
ポート
情報書.doc
印刷中
1/2
USB001
指示状.doc
印刷中
1/1
USB002
102
103

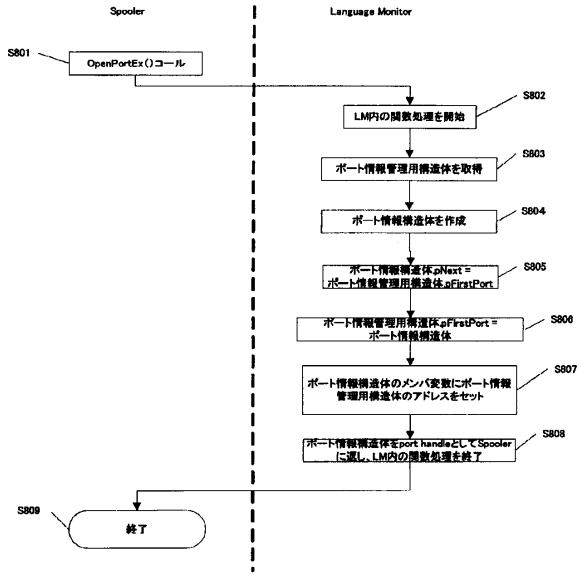
【図８】


201
ポート情報管理用構造体
pFirstPort
202
USB002のポート情報構造体
・ポート情報
・ポート情報管理用構造体のアドレス
pNext
203
USB001のポート情報構造体
・ポート情報
・ポート情報管理用構造体のアドレス

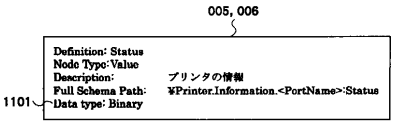
【図９】


Language Monitor
Spooler
InitializePrintMonitor2(コール)
S701
S702
LMPの関数処理を開始
S703
ポート情報管理用構造体を作成
S704
ポート情報管理用構造体をmonitor handleとしてSpooler(=返し、LMPの関数処理を終了
終了
S705

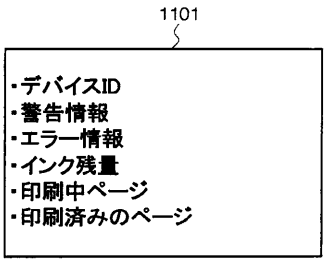
【図１０】



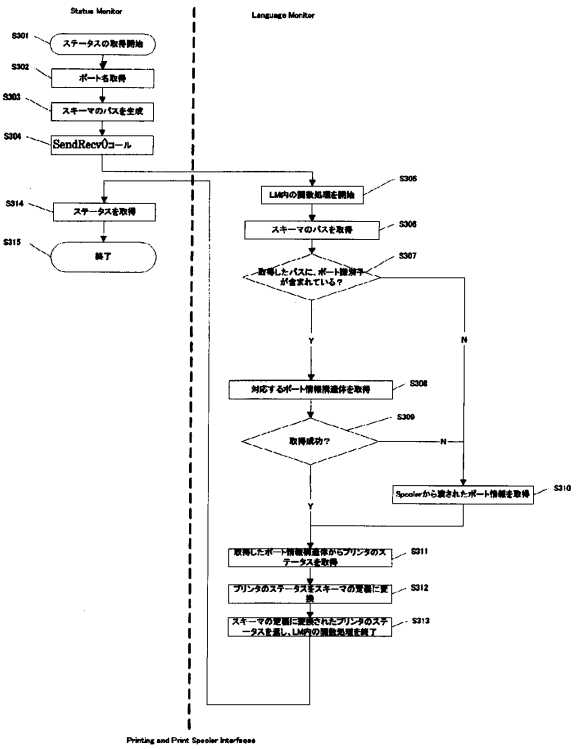
【図１１】



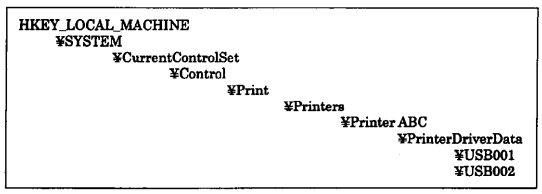
【図１２】



【図１３】



【図１４】

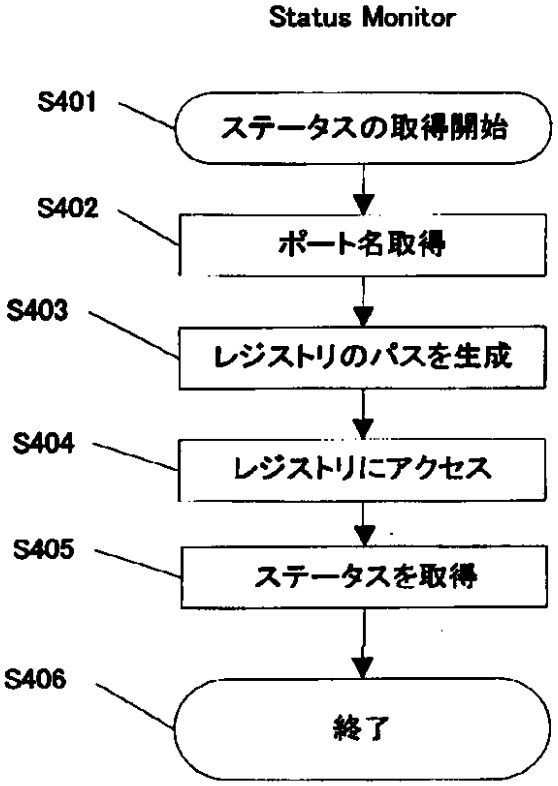


【図１５】

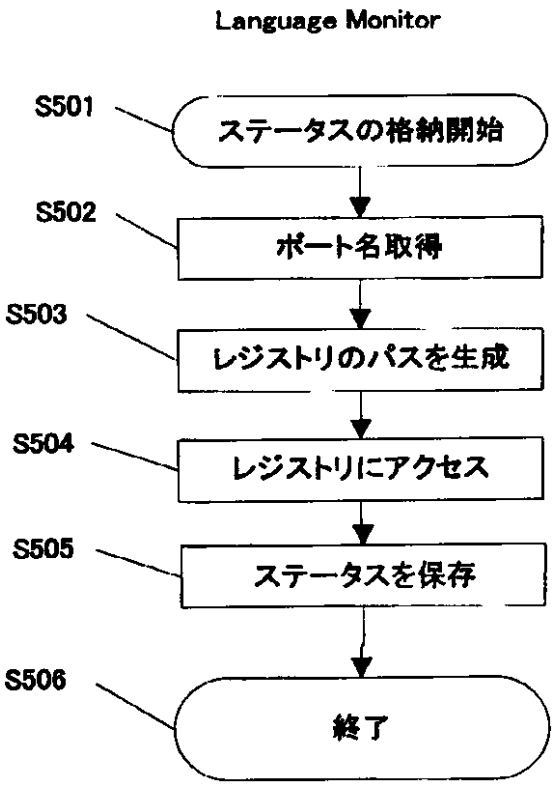
004

名前	種類	データ
Printer_Status_Cartridge	REG_SZ	Color
Printer_Status_Error	REG_SZ	No
Printer_Status_Warning	REG_DWORD	500
Printer_Status_Ink_Color	REG_DWORD	70

【図１６】



【図１７】



フロントページの続き

審査官　内田　正和

(56)参考文献　特開２００４－１８５２１７（ＪＰ，Ａ）
特開平０７－１０５１２０（ＪＰ，Ａ）
特開平１０－１２４４３７（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　　３／１２
Ｂ４１Ｊ　２９／３８
Ｇ０６Ｆ　　３／００
Ｇ０６Ｆ　１３／１４