

(19) 日本国特許庁(JP)

(12) 特 許 公 報(B2)

(11) 特許番号

特許第3733290号
(P3733290)

(45) 発行日 平成18年1月11日(2006. 1. 11)

(24) 登録日 平成17年10月21日(2005. 10. 21)

(51) Int. Cl.

F I

G 0 6 F 3/12 (2006.01)
B 4 1 J 29/38 (2006.01)G O 6 F 3/12 C
B 4 1 J 29/38 Z

請求項の数 20 (全 19 頁)

(21) 出願番号 特願2000-401218 (P2000-401218)
 (22) 出願日 平成12年12月28日 (2000. 12. 28)
 (65) 公開番号 特開2002-202865 (P2002-202865A)
 (43) 公開日 平成14年7月19日 (2002. 7. 19)
 審査請求日 平成15年12月10日 (2003. 12. 10)

(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 100071711
 弁理士 小林 将高
 (72) 発明者 川本 浩一
 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 内田 正和

最終頁に続く

(54) 【発明の名称】 印刷制御装置および印刷制御方法および記憶媒体

(57) 【特許請求の範囲】

【請求項1】

アプリケーションから印刷装置を制御するプリンタドライバに対して引き渡される、異なる複数の設定値間のコンフリクト処理を行なう印刷制御装置であって、

前記プリンタドライバの2つ以上の設定値の間で成立する依存関係を示すコンフリクト処理ルールを記憶する記憶手段と、

前記記憶手段に記憶されたコンフリクト処理ルールを読み込み、該読み込んだコンフリクト処理ルールに基づいて、補完的に成立するコンフリクト処理ルールを生成する生成手段と、

前記プリンタドライバが有するコンフリクトマネージャプログラムを実行して、前記生成手段により生成されたコンフリクト処理ルールを用いて、前記印刷装置に対する印刷実行に必要な設定値の不整合を回避するコンフリクト処理を適用して当該設定値の変更を行なうコンフリクト処理手段と、

前記コンフリクト処理手段によるコンフリクト処理の結果を、前記コンフリクトマネージャプログラムからアプリケーションに対して通知する通知手段と、
 を備えることを特徴とする印刷制御装置。

【請求項2】

アプリケーションと、アプリケーションから入力される印刷実行に必要な設定値のコンフリクト処理を行なうコンフリクトマネージャプログラムを有するプリンタドライバを搭載した印刷制御装置において、

10

20

アプリケーションから前記コンフリクトマネージャプログラムに対して印刷実行に必要な設定値を入力する入力手段と、

前記入力手段が前記コンフリクトマネージャプログラムに対して入力した設定値の不整合を回避するコンフリクト処理ルールを適用するコンフリクト処理手段と、

前記コンフリクト処理手段によるコンフリクト処理ルールの適用結果をコンフリクトマネージャプログラムから前記アプリケーションに対して通知する通知手段と、
を備えることを特徴とする印刷制御装置。

【請求項 3】

前記コンフリクト処理手段がコンフリクト処理ルールを適用した結果を前記通知手段がアプリケーションへ通知した後、アプリケーションからの指示に基づく前記印刷に必要な設定値を取得する取得手段をさらに備えることを特徴とする請求項 2 に記載の印刷制御装置。

10

【請求項 4】

前記アプリケーションは、前記通知手段から通知されてくるコンフリクト処理手段によるコンフリクト処理ルールの適用結果を取得して、当該処理内容に基づく設定を続行するか否かを確認する確認手段をさらに備えることを特徴とする請求項 2 又は 3 に記載の印刷制御装置。

【請求項 5】

前記コンフリクト処理手段は、コレート印刷を示す設定値とグループ印刷を示す設定値との競合、又は、ステープルを示す設定値と製本を示す設定値との競合のいずれかを回避するコンフリクト処理ルールを適用することを特徴とする請求項 1、又は、2 乃至 4 のいずれかに記載の印刷制御装置。

20

【請求項 6】

前記通知手段は、前記アプリケーションが読み出し可能な記憶手段に対してコンフリクト処理ルールの適用結果を書き込むことにより、前記アプリケーションに対して当該適用結果を通知することを特徴とする請求項 2 乃至 5 のいずれかに記載の印刷制御装置。

【請求項 7】

前記コンフリクト処理ルールを補完するルールを生成する生成手段をさらに有することを特徴とする請求項 2 乃至 6 のいずれかに記載の印刷制御装置。

【請求項 8】

前記記憶手段から読み込んだコンフリクト処理ルールが保持する二値を取り得る状態変数において、逆側の関係にあるコンフリクト処理ルールを、補完的に成立するコンフリクト処理ルールとして前記生成手段が生成することを特徴とする請求項 2 又は 7 に記載の印刷制御装置。

30

【請求項 9】

前記記憶手段から読み込んだコンフリクト処理ルールが保持する状態変数の値と、前記生成手段が生成した補完的に成立するコンフリクト処理ルールが保持する状態変数の値とが、論理的に排他のあることを特徴とする請求項 2 又は 7 に記載の印刷制御装置。

【請求項 10】

アプリケーションと、プリンタドライバを搭載した印刷制御装置において、
アプリケーションから前記プリンタドライバに対して印刷実行に必要な設定値を入力する入力手段と、

40

アプリケーションから、前記入力手段により入力された設定値の不整合を回避するコンフリクト処理ルールを適用するコンフリクト処理手段と、

前記コンフリクト処理手段によるコンフリクト処理ルールの適用結果を前記アプリケーションに対して通知する通知手段と、
を備えることを特徴とする印刷制御装置。

【請求項 11】

アプリケーションと、アプリケーションから入力される印刷実行に必要な設定値のコンフリクト処理を行なうコンフリクトマネージャプログラムを有するプリンタドライバを実

50

行する印刷制御装置における方法において、

アプリケーションから前記コンフリクトマネージャプログラムに対して印刷実行に必要な設定値を入力する入力ステップと、

前記入力ステップが前記コンフリクトマネージャプログラムに対して入力した設定値の不整合を回避するコンフリクト処理ルールを適用するコンフリクト処理ステップと、

前記コンフリクト処理ステップによるコンフリクト処理ルールの適用結果をコンフリクトマネージャプログラムから前記アプリケーションに対して通知する通知ステップと、を印刷制御装置に実行させることを特徴とする印刷制御方法。

【請求項 1 2】

前記コンフリクト処理ステップがコンフリクト処理ルールを適用した結果を前記通知ステップがアプリケーションへ通知した後、アプリケーションからの指示に基づく前記印刷に必要な設定値を取得する取得ステップをさらに備えることを特徴とする請求項 9 に記載の印刷制御方法。

10

【請求項 1 3】

前記アプリケーションは、前記通知ステップから通知されてくるコンフリクト処理手段によるコンフリクト処理ルールの適用結果を取得して、当該処理内容に基づく設定を続行するか否かを確認する確認ステップをさらに備えることを特徴とする請求項 1 1 又は 1 2 に記載の印刷制御方法。

【請求項 1 4】

前記コンフリクト処理ステップは、コレート印刷を示す設定値とグループ印刷を示す設定値との競合、又は、ステーブルを示す設定値と製本を示す設定値との競合のいずれかを回避するコンフリクト処理ルールを適用することを特徴とする請求項 1 0 乃至 1 3 のいずれかに記載の印刷制御方法。

20

【請求項 1 5】

前記通知ステップは、前記アプリケーションが読み出し可能な記憶手段に対してコンフリクト処理ルールの適用結果を書き込むことにより、前記アプリケーションに対して当該適用結果を通知することを特徴とする請求項 1 1 乃至 1 4 のいずれかに記載の印刷制御方法。

【請求項 1 6】

前記コンフリクト処理ルールを補完するルールを生成する生成ステップをさらに有することを特徴とする請求項 1 1 乃至 1 5 のいずれかに記載の印刷制御方法。

30

【請求項 1 7】

前記記憶手段から読み込んだコンフリクト処理ルールが保持する二値を取り得る状態変数において、逆側の関係にあるコンフリクト処理ルールを、補完的に成立するコンフリクト処理ルールとして前記生成ステップが生成することを特徴とする請求項 1 1 又は 1 6 に記載の印刷制御方法。

【請求項 1 8】

前記記憶手段から読み込んだコンフリクト処理ルールが保持する状態変数の値と、前記生成ステップが生成した補完的に成立するコンフリクト処理ルールが保持する状態変数の値とが、論理的に排他的関係にあることを特徴とする請求項 1 1 又は 1 7 に記載の印刷制御方法。

40

【請求項 1 9】

アプリケーションと、プリンタドライバを搭載した印刷制御装置における印刷制御方法において、

アプリケーションから前記プリンタドライバに対して印刷実行に必要な設定値を入力する入力ステップと、

アプリケーションから、前記入力ステップにより入力された設定値の不整合を回避するコンフリクト処理ルールを適用するコンフリクト処理ステップと、

前記コンフリクト処理ステップによるコンフリクト処理ルールの適用結果を前記アプリケーションに対して通知する通知ステップと、

50

を備えることを特徴とする印刷制御方法。

【請求項 20】

請求項 11 または 19 に記載の印刷制御方法をコンピュータに実行させるためのプログラムを格納したことを特徴とするコンピュータが読み取り可能な記憶媒体。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】

本発明は、アプリケーションから印刷装置を制御する印刷処理関連プログラムに対して引き渡される異なる複数の設定値間のコンフリクト処理を実行する印刷制御装置および印刷制御方法および記憶媒体に関するものである。

10

【0002】

【従来の技術】

従来、コンピュータ上で動作するアプリケーションプログラムが、印刷を実行する場合には、印刷開始命令と共に、印刷に必要な各種設定をパラメータとして、プリンタドライバ等の印刷関連処理プログラムに引き渡すのが一般的な方法である。このように印刷に必要な各種設定をパラメータとしてプログラムよりプリンタドライバに渡すことで、プログラムが意図した印刷出力を得られるようになっている。

【0003】

【発明が解決しようとする課題】

しかしながら、意図した印刷出力を得るために指定可能な設定値の種類の範囲はいわゆる 20
どのプリンタでもサポートが可能と思われる用紙の給紙口の指定や両面／片面印刷の指定等、非常に限られた範囲のものであり、プリンタまたはプリンタドライバが提供していても、アプリケーションプログラムより直接指定ができないその他の有用な機能については、プリンタドライバのユーザインタフェースを開いて、そこで指定をする必要があった。

【0004】

このため、企業の各種システムソフトウェアからの印刷出力に際して、プリンタドライバのユーザインタフェースを表示したがないケースでは、プリンタまたはプリンタドライバの持つ機能が十分に利用できないという問題点があった。

【0005】

そのため、昨今ではプリンタまたはプリンタドライバをフルに生かせるように、プリンタ 30
やプリンタドライバへの機種に依存する設定についても、アプリケーションから利用可能な特別なインタフェースを設けて、このインタフェースを利用した場合には、プリンタドライバユーザインタフェースで設定する場合と同等な機能の設定が可能となるようなプリンタドライバが出現してきている。

【0006】

こうなると、このインタフェースを介したアプリケーションプログラムからの設定についても、プリンタドライバユーザインタフェースと同等のコンフリクト処理（複数の設定値間での設定の競合状態を解消して、いずれかの設定値の調整を行い、設定値間の整合性を取るための処理）を必要とするので、効率よく、漏れの少ない、そして、ユーザインタフェースで利用されているコンフリクト処理と同等のコンフリクト処理をユーザインタフェ 40
ース以外の場面でも実現する必要が生じているわけである。

【0007】

本発明は、上記の問題点を解決するためになされたもので、本発明の目的は、コンフリクト処理を、例えばコンフリクトマネージャなどプリンタドライバ側で実行し、該コンフリクト処理の結果をアプリケーションに通知できる印刷制御装置および印刷制御方法および記憶媒体を提供することである。

【0008】

本発明に係る第 1 の発明は、アプリケーションから印刷装置を制御するプリンタドライバに対して引き渡される、異なる複数の設定値間のコンフリクト処理を行なう印刷制御装置であって、前記プリンタドライバの 2 つ以上の設定値の間で成立する依存関係を示すコ 50

ンフリクト処理ルールを記憶する記憶手段と、前記記憶手段に記憶されたコンフリクト処理ルールを読み込み、該読み込んだコンフリクト処理ルールに基づいて、補完的に成立するコンフリクト処理ルールを生成する生成手段と、前記プリンタドライバが有するコンフリクトマネージャプログラムを実行して、前記生成手段により生成されたコンフリクト処理ルールを用いて、前記印刷装置に対する印刷実行に必要な設定値の不整合を回避するコンフリクト処理を適用して当該設定値の変更を行なうコンフリクト処理手段と、前記コンフリクト処理手段によるコンフリクト処理の結果を、前記コンフリクトマネージャプログラムからアプリケーションに対して通知する通知手段とを備えることを特徴とする。

【0009】

本発明に係る第2の発明は、アプリケーションと、アプリケーションから入力される印刷実行に必要な設定値のコンフリクト処理を行なうコンフリクトマネージャプログラムを有するプリンタドライバを搭載した印刷制御装置において、アプリケーションから前記コンフリクトマネージャプログラムに対して印刷実行に必要な設定値を入力する入力手段と、前記入力手段が前記コンフリクトマネージャプログラムに対して入力した設定値の不整合を回避するコンフリクト処理ルールを適用するコンフリクト処理手段と、前記コンフリクト処理手段によるコンフリクト処理結果をコンフリクトマネージャプログラムから前記アプリケーションに対して通知する通知手段とを備えることを特徴とする。

【0010】

本発明に係る第3の発明は、前記コンフリクト処理手段がコンフリクト処理ルールを適用した結果を前記通知手段がアプリケーションへ通知した後、アプリケーションからの指示に基づく前記印刷に必要な設定値を取得する取得手段をさらに備えることを特徴とする。

【0011】

本発明に係る第4の発明は、前記アプリケーションは、前記通知手段から通知されてくるコンフリクト処理手段によるコンフリクト処理結果の適用結果を取得して、当該処理内容に基づく設定を続行するか否かを確認する確認手段をさらに備えることを特徴とする。

【0012】

本発明に係る第5の発明は、前記コンフリクト処理手段は、コレート印刷を示す設定値とグループ印刷を示す設定値との競合、又は、ステープルを示す設定値と製本を示す設定値との競合のいずれかを回避するコンフリクト処理ルールを適用することを特徴とする。

【0013】

本発明に係る第6の発明は、前記通知手段は、前記アプリケーションが読み出し可能な記憶手段に対してコンフリクト処理結果の適用結果を書き込むことにより、前記アプリケーションに対して当該適用結果を通知することを特徴とする。

【0014】

本発明に係る第7の発明は、前記コンフリクト処理ルールを補完するルールを生成する生成手段をさらに有することを特徴とする。

【0015】

本発明に係る第8の発明は、前記記憶手段から読み込んだコンフリクト処理ルールが保持する二値を取り得る状態変数において、逆側の関係にあるコンフリクト処理ルールを、補完的に成立するコンフリクト処理ルールとして前記生成手段が生成することを特徴とする。

【0016】

本発明に係る第9の発明は、前記記憶手段から読み込んだコンフリクト処理ルールが保持する状態変数の値と、前記生成手段が生成した補完的に成立するコンフリクト処理ルールが保持する状態変数の値とが、論理的に排他的関係にあることを特徴とする。

本発明に係る第10の発明は、アプリケーションと、プリンタドライバを搭載した印刷制御装置において、アプリケーションから前記プリンタドライバに対して印刷実行に必要な設定値を入力する入力手段と、アプリケーションから、前記入力手段により入力された

10

20

30

40

50

設定値の不整合を回避するコンフリクト処理ルールを適用するコンフリクト処理手段と、前記コンフリクト処理手段によるコンフリクト処理ルールの適用結果を前記アプリケーションに対して通知する通知手段とを備えることを特徴とする。

本発明に係る第11の発明は、アプリケーションと、アプリケーションから入力される印刷実行に必要な設定値のコンフリクト処理を行なうコンフリクトマネージャプログラムを有するプリンタドライバを実行する印刷制御装置における方法において、アプリケーションから前記コンフリクトマネージャプログラムに対して印刷実行に必要な設定値を入力する入力ステップと、前記入力ステップが前記コンフリクトマネージャプログラムに対して入力した設定値の不整合を回避するコンフリクト処理ルールを適用するコンフリクト処理ステップと、前記コンフリクト処理ステップによるコンフリクト処理ルールの適用結果をコンフリクトマネージャプログラムから前記アプリケーションに対して通知する通知ステップとを印刷制御装置に実行させることを特徴とする。

10

本発明に係る第12の発明は、前記コンフリクト処理ステップがコンフリクト処理ルールを適用した結果を前記通知ステップがアプリケーションへ通知した後、アプリケーションからの指示に基づく前記印刷に必要な設定値を取得する取得ステップをさらに備えることを特徴とする。

本発明に係る第13の発明は、前記アプリケーションは、前記通知ステップから通知されてくるコンフリクト処理手段によるコンフリクト処理ルールの適用結果を取得して、当該処理内容に基づく設定を続行するか否かを確認する確認ステップをさらに備えることを特徴とする。

20

本発明に係る第14の発明は、前記コンフリクト処理ステップは、コレート印刷を示す設定値とグループ印刷を示す設定値との競合、又は、ステープルを示す設定値と製本を示す設定値との競合のいずれかを回避するコンフリクト処理ルールを適用することを特徴とする。

本発明に係る第15の発明は、前記通知ステップは、前記アプリケーションが読み出し可能な記憶手段に対してコンフリクト処理ルールの適用結果を書き込むことにより、前記アプリケーションに対して当該適用結果を通知することを特徴とする。

本発明に係る第16の発明は、前記コンフリクト処理ルールを補完するルールを生成する生成ステップをさらに有することを特徴とする。

本発明に係る第17の発明は、前記記憶手段から読み込んだコンフリクト処理ルールが保持する二値を取り得る状態変数において、逆側の関係にあるコンフリクト処理ルールを、補完的に成立するコンフリクト処理ルールとして前記生成ステップが生成することを特徴とする。

30

本発明に係る第18の発明は、前記記憶手段から読み込んだコンフリクト処理ルールが保持する状態変数の値と、前記生成ステップが生成した補完的に成立するコンフリクト処理ルールが保持する状態変数の値とが、論理的に排他的関係にあることを特徴とする。

本発明に係る第19の発明は、アプリケーションと、プリンタドライバを搭載した印刷制御装置における印刷制御方法において、アプリケーションから前記プリンタドライバに対して印刷実行に必要な設定値を入力する入力ステップと、アプリケーションから、前記入力ステップにより入力された設定値の不整合を回避するコンフリクト処理ルールを適用するコンフリクト処理ステップと、前記コンフリクト処理ステップによるコンフリクト処理ルールの適用結果を前記アプリケーションに対して通知する通知ステップとを備えることを特徴とする。

40

本発明に係る第20の発明は、第11または第19の発明の印刷制御方法をコンピュータに実行させるためのプログラムをコンピュータが読み取り可能な記憶媒体に格納したことを特徴とする。

【0017】

【発明の実施の形態】

図1は、本発明の第1実施形態を示す印刷制御装置を適用可能な印刷処理システムの構成を説明するブロック図である。

50

【0018】

なお、特に断らない限り、本発明の機能が実行されるのであれば、単体の機器であっても、複数の機器からなるシステムであっても、LAN、WAN等のネットワークを介して接続が為され処理が行われるシステムであっても本発明を適用できることは言うまでもない。

【0019】

図1において、3000はホストコンピュータで、ROM3のプログラム用ROMあるいは外部メモリ11に記憶された文書処理プログラム等に基づいて図形、イメージ、文字、表（表計算等を含む）等が混在した文書処理を実行するCPU1を備え、システムバス4に接続される各デバイスをCPU1が総括的に制御する。

10

【0020】

また、このROM3のプログラム用ROMあるいは外部メモリ11には、CPU1の制御プログラムであるオペレーティングシステムプログラム（以下OS）等を記憶し、ROM3のフォント用ROMあるいは外部メモリ11には上記文書処理の際に使用するフォントデータ等を記憶し、ROM3のデータ用ROMあるいは外部メモリ11には上記文書処理等を行う際に使用する各種データを記憶する。

【0021】

2はRAMで、CPU1の主メモリ、ワークエリア等として機能する。5はキーボードコントローラ（KBC）で、キーボード（KB）9や不図示のポインティングデバイスからのキー入力を制御する。

20

【0022】

6はCRTコントローラ（CRTC）で、CRTディスプレイ（CRT）10の表示を制御する。7はディスクコントローラ（DKC）で、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、プリンタ制御コマンド生成プログラム（以下プリンタドライバ）等を記憶するハードディスク（HD）、フロッピーディスク（FD）等の外部メモリ11とのアクセスを制御する。

【0023】

8はプリンタコントローラ（PRTC）で、所定の双方向性インターフェース（インターフェース）21を介してプリンタ1500に接続されて、プリンタ1500との通信制御処理を実行する。なお、CPU1は、例えばRAM2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（ラスターライズ）処理を実行し、CRT10上でのWYSIWYGを可能としている。また、CPU1は、CRT10上の不図示のマウスカーソル等で指示されたコマンドに基づいて登録された種々のウインドウを開き、種々のデータ処理を実行する。ユーザは印刷を実行する際、印刷の設定に関するウインドウを開き、プリンタの設定や、印刷モードの選択を含むプリンタドライバに対する印刷処理方法の設定を与える。

30

【0024】

プリンタ1500において、12はプリンタCPU（CPU）で、ROM13のプログラム用ROMに記憶された制御プログラム等あるいは外部メモリ14に記憶された制御プログラム等に基づいてシステムバス15に接続される印刷部（プリンタエンジン）17に出力情報としての画像信号を出力する。

40

【0025】

また、このROM13のプログラム用ROMには、CPU12の制御プログラム等を記憶する。ROM13のフォント用ROMには上記出力情報を生成する際に使用するフォントデータ等を記憶し、ROM13のデータ用ROMにはハードディスク等の外部メモリ14がないプリンタの場合には、ホストコンピュータ上で利用される情報等を記憶している。

【0026】

CPU12は入力部18を介してホストコンピュータとの通信処理が可能となっており、プリンタ内の情報等をホストコンピュータ3000に通知可能に構成されている。

【0027】

50

19は前記CPU12の主メモリ、ワークエリア等として機能するRAMで、図示しない増設ポートに接続されるオプションRAMによりメモリ容量を拡張することができるように構成されている。

【0028】

なお、RAM19は、出力情報展開領域、環境データ格納領域、NVRAM等に用いられる。前述したハードディスク(HD)、ICカードやフロッピーディスク(FD)等の外部メモリ14は、メモリコントローラ(DKC)20によりアクセスを制御される。外部メモリ14は、オプションとして接続され、フォントデータ、エミュレーションプログラム、フォームデータ等を記憶する。

【0029】

また、1501はた操作パネルで、操作のためのスイッチおよびLED表示器等が配されている。また、前述した外部メモリは1個に限らず、少なくとも1個以上備え、内蔵フォントに加えてオプションフォントカード、言語系の異なるプリンタ制御言語を解釈するプログラムを格納した外部メモリを複数接続できるように構成されていてもよい。さらに、図示しないNVRAMを有し、操作パネル1501からのプリンタモード設定情報を記憶するようにしてもよい。

【0030】

図2は、図1に示したRAM3のメモリマップの一例を示す図であり、本実施形態における印刷処理関連プログラムがホストコンピュータ3000上のRAM2にロードされ実行可能となった状態のメモリマップを示している。

【0031】

図2において、201はアプリケーションで、外部メモリ11からロードされたものであり、単数または複数のアプリケーションがロードされる。

【0032】

202は空きメモリで、CPU1のワークとして機能する。203は関連データで、アプリケーション201の実行に伴う各種の関連データがロードされる。

【0033】

204は印刷処理関連プログラムで、外部メモリ11からロードされる。205はOSで、アプリケーション201等の実行を制御する。206はBIOSで、各種のデバイスの入出力処理を規定する。

【0034】

図3は、図2に示した印刷処理関連プログラムにおけるコンフリクト処理制御部の内部処理の内容を説明するシステム概要図である。

【0035】

図3において、301はコンフリクト処理ルールで、コンフリクトマネージャ303により図1に示したRAM2上に読み込まれる。302は推論エンジンで、図3に例示してあるように表記されたコンフリクト処理ルール301をコンフリクトマネージャ303を介して、RAM2中に読み込んで補完ルールを生成する。

【0036】

304は状態変数リストで、各プリンタ機能名A、B、C(図4参照)が推論エンジン302により初期化される。コンフリクト処理ルール403の中に出現するプリンタ機能名A、B、Cのそれぞれについて同名の状態変数が存在する。

【0037】

305は内部構造体で、プリンタ機能名A、B、Cに対応するプリンタドライバ内部構造体401のメンバをそれぞれcA、cB、cCとして管理する。306はプリンタドライバ管理部で、コンフリクトマネージャ303を呼び出し、状態変数リスト304にあるLayoutの状態変数を更新する。307はアプリケーションである。

【0038】

図4は、本発明に係る印刷制御装置におけるコンフリクト処理における各モジュールで扱うデータの関連性を説明する図である。

10

20

30

40

50

【0039】

図4において、401はプリンタドライバの内部構造体で、図3に示したコンフリクトマネージャ303またはプリンタドライバ管理部306により読み出しあるいは書き込み可能に構成されている。402は状態変数リストで、図3に示した状態変数リストと同等であって、コンフリクトマネージャ303により読み出しあるいは書き込み可能に構成されている。なお、図中のRは読み出しを表し、Wは書き込みを表し、R/Wは読み出しあるいは書き込みを表す。

【0040】

図5は、本発明に係る印刷制御装置における補完処理前のコンフリクト処理規則の例を示す図である。

10

【0041】

図5において、(1)～(3)は補間処理前のコンフリクト処理規則301の一例を示し、規則(1)はプリンタドライバ内部構造体401のメンバとして存在するGroupに、規則(2)はプリンタドライバ内部構造体401のメンバとして存在するStapleに、規則(3)はプリンタドライバ内部構造体401のメンバとして存在するLayoutに対応する。

【0042】

図6は、本発明に係る印刷制御装置における補完処理後のコンフリクト処理規則の例を示す図である。

【0043】

20

図6において、(1)～(9)は補完処理後のコンフリクト処理規則(規則)であり、特に、規則(6)～(9)における__Aは推論エンジン302で使用される変数を示す。

【0044】

図7は、本発明に係る印刷制御装置における設定変更処理関数の一例を示す図であり、アプリケーションからプリンタドライバへ設定変更処理要求をする場合に利用する関数のインタフェース(I/F)の例である。

【0045】

図8は、本発明に係る印刷制御装置におけるデータ処理手順の一例を示すフローチャートであり、本実施形態におけるコンフリクト処理手順に対応する。なお、(501)～(508)は各ステップを示す。

30

【0046】

まず、本実施形態に示す印刷処理関連プログラム(いわゆるプリンタドライバ)の処理は、OS205管理の下で動作するアプリケーション201よりプリンタドライバの備える初期化処理が呼び出されると、OS205の管理の下、RAM2に図2に示した印刷処理関連プログラム204がロードされる。

【0047】

そして、印刷処理関連プログラム204がRAM2にロードされると、プリンタドライバ管理部306に備えられたプリンタドライバの初期化処理部が呼び出され、初期化処理が行われる。

40

【0048】

まず、図3に示す推論エンジン302は、図5に例示してあるように表記されたコンフリクト処理規則301をコンフリクトマネージャ303を介して、RAM2中に読み込む(501)。

【0049】

続いて、ステップ(501)で読み込んだコンフリクト処理規則301を元に、2状態値(「ON」と「OFF」)を値として持つ規則については、補完規則を生成する(502)。

【0050】

なお、同じプリンタ機能名について以下の例のように左辺に「ONまたはOFF」のどち

50

らかだけが記述された場合、

A (ON) <- B (ON) , C (OFF) …… (1)

A (ON) <- D (V1) . …… (2)

B (OFF) <- E (OFF) . …… (3)

この場合には推論エンジン302がON/OFF逆側のルールを以下のように自動生成する。

【0051】

上記ルール(1) , (2)に対しては、

A (OFF) <- not A (ON) . …… (4)

が生成され、上記(3)に対しては、

B (ON) <- not B (OFF) . …… (5)

が生成される。この自動生成されたルール(4) , (5)は処理効率上最適化されて下記のルール(4)' , (5)'のように変形されるが、意味は全く同じとなる。

A (OFF) <- true . …… (4)'

B (ON) <- true . …… (5)'

なお、上記例のように逆側が自動生成された場合は、論理として考えた際に、A (ON)とA (OFF)は完全に排他的な関係にある事を意味している。

【0052】

つまり、A (ON)とA (OFF)とでAについての集合空間が100%埋め尽くされる事を意味している。B (ON)とB (OFF)も同様である。

【0053】

ユーザがA (ON)とA (OFF)を混在して記述する場合にはルール(4) , (4)'は自動生成されない。従って、この場合には全てのケースを網羅的に記述してAについての集合空間を100%埋め尽くすようにする必要がある。

【0054】

続いて、図4に示す状態変数リスト402とコンフリクト処理ルール403で使用するプリンタ機能名の値の初期化に関するコンフリクト処理ルールを補完ルールとして自動生成する(ステップ503)。

【0055】

コンフリクト処理ルール403における全てのプリンタ機能名についてはコンフリクトマネージャ303の内部に、図4に示すような状態変数リスト402として、その状態変数を持っている。

【0056】

この状態変数の値はプリンタドライバで使用される内部構造体の対応するメンバの値と連動している。全てのプリンタ機能名の状態変数の初期値はその内部構造体305のメンバの値が初期値となる。

例えば以下のコンフリクト処理ルールを記述した場合、「A (ON) <- B (ON) , C (OFF) .」は、図4に示した状態変数リスト402に示すようにコンフリクト処理ルール403の中に出現するプリンタ機能名A, B, Cのそれぞれについて同名の状態変数が存在している。

【0057】

プリンタ機能名A, B, Cに対応するプリンタドライバ内部構造体401のメンバをそれぞれcA, cB, cCとする。

【0058】

なお、int cAの初期値は「0」なので、それに対応する状態変数Aの値は「OFF」となっている。

【0059】

従って、推論エンジン302内のプリンタ機能名Aの状態値の初期値もOFFとなっている。

【0060】

10

20

30

40

50

今、推論エンジン 302 でコンフリクトチェックの推論が行われて以下のルール「A (ON) <- B (ON), C (OFF).」が成立した場合、推論エンジン 302 は左辺のプリンタ機能名 A の状態変数値を「ON」に変更する。

【0061】

コンフリクトチェックの推論が終了した後、コンフリクトマネージャ 303 は変更された状態変数の値をプリンタドライバ内部構造体 401 の対応するメンバ `int cA` に反映させる。

【0062】

つまり、`int cA` は上記ルールが成立した事によって「0」から「1」に変更される。

10

【0063】

推論エンジン 302 は、推論エンジンに組込まれている組込み関数 `status(a, __X)` によってプリンタ機能名 A の状態変数値を推論エンジン 302 で使用される変数 `__X` に受ける事ができる。推論エンジン 302 は、コンフリクト処理ルールをロードした後、そのコンフリクト処理ルール中出现する全てのルール名について、次のルールを自動的に生成する。

A (`__X`) <- `status` (A, `__X`).

B (`__X`) <- `status` (B, `__X`).

C (`__X`) <- `status` (C, `__X`).

これは他に適用するルールが存在しない場合には、プリンタドライバ内部構造体 401 の対応するメンバの値がそのプリンタ機能名の状態値となる事を意味する。

20

【0064】

プリンタ機能名 A については、ルール「A (ON) <- B (ON), C (OFF).」が成り立つのでプリンタ機能名 A の状態値はこれを受けて [ON] となる。プリンタ機能名 B がもし上記自動生成されたルール以外にルールが存在しないならば、ルール「B (`__X`) <- `status` (B, `__X`).' が適用される。この自動生成ルールは必ず成り立つので、プリンタ機種名 B の状態変数の値「ON」が「`__X`」にユニファイされ、それがプリンタ機種名 B の状態値となる。

【0065】

つまり、ユーザ定義ルールが存在しないか、または存在していても成り立つものが無いプリンタ機能については、プリンタドライバ内部構造体 401 の対応するメンバに格納されている値がそのプリンタ機能の状態値という事になる。

30

【0066】

続いて、その他の初期化処理を行い、処理化処理を終了しアプリケーションに制御を戻す (ステップ 504)。

【0067】

初期化処理終了後は、プリンタドライバ管理部 306 に備えられている図 7 に示す設定変更処理関数を呼び出してのアプリケーションからの設定変更処理を繰り返す (ステップ 505 ~ 507)。

【0068】

この関数で用いられている引数の中で、特に設定変更処理に関わるパラメータは、`pDevModeInOutPut` と、`pItemParam`, `dwItemID` である。それぞれのパラメータには以下のような情報がセットされる。

40

【0069】

パラメータ `dwItemID` には、変更を行いたいのはプリンタドライバ内部構造体 401 のどのメンバなのかを指定する。パラメータ `pItemParam` には、パラメータ `dwItemID` で示したメンバのパラメータを指定する。また、パラメータ `pDevModeInOutPut` には、プリンタドライバ内部構造体 401 へのポインタを指定する。

【0070】

50

例えば、アプリケーションからの設定変更要求がPrint Styleを1-side d PrintingからBooklet Printingに変更するものであった場合を例にとると、上記の各々のパラメータには、パラメータdwItemIDには、「DM__ITEM__LAYOUT」が、パラメータpItemParamには、「DM__PARAM__BOOKLET」が、パラメータpDevModeInOutputには、「pDevMode」がセットされた上で、DocumentPropertiesEx () が呼び出される。

【0071】

すると、図5のステップ(505)～(507)に示すプリンタドライバの設定変更処理が開始されることとなる。

10

【0072】

まず、ステップ(505)で実行されるコンフリクト処理規則の適用について説明する。

【0073】

上記の例を用いて説明すると、コンフリクト処理規則の適用は、図5に示すコンフリクト処理規則301の一部として読み込まれた規則と、ステップ(502)にて生成された補完規則の一部である図6に示した(4)、(5)とステップ(503)にて生成された規則の一部である図6に示した(6)～(9)に対して行われる。

【0074】

プリンタドライバ内部構造体401のメンバとして存在するCollate, Group, Staple, Layoutの各メンバのコンフリクト処理規則適用前の値は、Collateは「OFF」で、Groupが「ON」で、Stapleが「OFF」で、Layoutが「1-Side」となる。

20

【0075】

そして、アプリケーションからの変更要求がLayoutをLayoutからBookletに変更するものであるので、Layoutのメンバの内容は、Collateは「OFF」で、Groupが「ON」で、Stapleが「OFF」で、Layoutが「Booklet」となる。

【0076】

次に、プリンタドライバ管理部306はコンフリクトマネージャ303を呼び出し、状態変数リスト304にあるLayoutの状態変数が更新され、続いて、推論エンジン302がコールされて、コンフリクト処理規則301の適用が始まる。

30

【0077】

まず、図6に示した規則(6)～(9)が適用され、推論エンジン302内の各プリンタ機能名が状態変数リスト302の各メンバの持つ値で初期化される。続いて、図6に示した規則(3)が適用され、Groupの値は「ON」から「OFF」へと変化する。

【0078】

これにより、Layoutのメンバの内容は、Collateは「OFF」で、Groupが「OFF」で、Stapleが「OFF」で、Layoutが「Booklet」となる。

40

【0079】

さらに、図6に示す規則(4)が適用され、Collateは「OFF」から「ON」へと変化する。

【0080】

これにより、Layoutのメンバの内容は、Collateは「ON」で、Groupが「OFF」で、Stapleが「OFF」で、Layoutが「Booklet」となる。

【0081】

他に、適用される規則が存在しないので、以上で推論エンジン302でのコンフリクト処理規則の適用が終了する。

50

【0082】

次のステップでは、コンフリクトマネージャ303が上記の最終状態を元に状態変数リスト304の更新(ステップ506)と、プリンタドライバ内部構造体の更新(ステップ507)を行う。

【0083】

本来であればアプリケーション自身の手により、このLayoutの設定関連するメンバであるGroupとCollateのメンバの値を調整する必要があるが、DocumentPropertiesExを利用することで、この例の場合、アプリケーションは、Layoutを「1-Side」から「Booklet」に変更する処理をプリンタドライバに依頼することによって、Groupが「ON」から「OFF」に、Collate 10
が「OFF」から「ON」に更新されたプリンタドライバ内部構造体301を受け取ることができる。

【0084】

この後、ここで取得したプリンタドライバ内部構造体301をパラメータとして実際の印刷を開始すれば、アプリケーションが所望する印刷結果が簡便に得られることとなる。

【0085】

以上の処理は、アプリケーションによってプリンタドライバ管理部306に備えられた終了処理が呼び出されるまで、繰り返し実行される。アプリケーションよりプリンタドライバの終了処理が呼び出されると空きメモリ202に生成したコンフリクト処理ルール情報等を消去すると共にその他の終了処理を行う(ステップ508)。 20

【0086】

これにより処理は全て終了し、本実施形態における印刷処理関連プログラム204の処理も終了し、RAM2からはOS205の機能により消去される。

【0087】

なお、本実施形態においては、本印刷処理関連プログラム204を記録する媒体を外部メモリとしているが、外部メモリとしては、FD、HDドライブ、CD-ROMやICメモリカード等であってもよい。更に、本印刷プログラム単独、もしくはOSその他のホストコンピュータ上で動作するプログラムと共にROM3に記録しておき、これをメモリマップの一部となすように構成し、直接CPU1で実行することも可能である。

【0088】

〔第2実施形態〕

上記第1実施形態においては、プリンタドライバ内部構造体401の設定内容をコンフリクト処理を実施することで、自動的に最適な値に更新することとなるが、さらに、図9、図10に示すとおり、コンフリクトマネージャ303にコンフリクト処理結果リスト901を加え、状態変数リスト402が更新されるのと同時に、その中に状態変数の変更履歴を書きこむことで、コンフリクト処理を実行することにより、プリンタドライバ内部構造体のどのメンバが更新されたのかをアプリケーションが知ることができるようになる。 30

【0089】

図9は、本発明の第2実施形態を示す印刷制御装置におけるコンフリクト処理制御部の内部処理の内容を説明するシステム概要図であり、図3と同一のものには同一の符号を付してある。 40

【0090】

図10は、本発明に係る印刷制御装置におけるコンフリクト処理における各モジュールで扱うデータの関連性を説明する図であり、図4と同一のものには同一の符号を付してある。

【0091】

図11は、本発明に係る印刷制御装置における他の設定変更処理関数の一例を示す図である。

【0092】

図9に示すコンフリクト処理結果リストの例は、プリンタドライバ内部構造体401のメ 50

ンバの内、コンフリクト処理ルールにて更新がされたメンバは1個ということが、ResultNumに保存され、そして、更新されたメンバのリストが後続のModified[]に保存され、この場合はResultNumが1であるので、リスト内のメンバも1つで、ID__ca、すなわち、プリンタドライバ内部構造体401のcAが更新されたことを表現している。

【0093】

アプリケーション307は、このコンフリクト処理結果リストへのポインタを図11に示すDocumPropertiesEx関数の戻り値として受け取ることで、その情報にアクセスできるようになる。

【0094】

上記実施形態を例にとれば、コンフリクト処理結果リスト901の内容としては、ID__GroupとID__Collateが含まれる。アプリケーション307は、このコンフリクト処理結果リスト901の内容を確認して、実際にその値が格納されているプリンタドライバ内部構造体401の該当するメンバ、Group、Collateの内容を確認し、その設定内容がアプリケーション307が望まない設定変更で合った場合には、Layoutを「1-Side」から「Booklet」への設定の変更を取り消すことも可能である。

【0095】

上記各実施形態によれば、プログラム開発者などが用意するコンフリクト条件を補完するコンフリクト条件を自動生成するので、より品質の高いコンフリクト処理を実現できる。

【0096】

さらに、コンフリクト処理が実行されることによって、設定が変更されたかどうかの情報を各設定値について取得できるので、この情報に基づいた処理、例えば、他の設定項目に影響を与える元となった設定変更の取り消しやユーザへのメッセージ出力等を容易に実装できる。

【0097】

以下、図12に示すメモリマップを参照して本発明に係る印刷制御装置を適用可能な印刷システムで読み出し可能なデータ処理プログラムの構成について説明する。

【0098】

図12は、本発明に係る印刷制御装置を適用可能な印刷システムで読み出し可能な各種データ処理プログラムを格納する記憶媒体のメモリマップを説明する図である。

【0099】

なお、特に図示しないが、記憶媒体に記憶されるプログラム群を管理する情報、例えばバージョン情報、作成者等も記憶され、かつ、プログラム読み出し側のOS等に依存する情報、例えばプログラムを識別表示するアイコン等も記憶される場合もある。

【0100】

さらに、各種プログラムに従属するデータも上記ディレクトリに管理されている。また、各種プログラムをコンピュータにインストールするためのプログラムや、インストールするプログラムが圧縮されている場合に、解凍するプログラム等も記憶される場合もある。

【0101】

本実施形態における図8に示す機能が外部からインストールされるプログラムによって、ホストコンピュータにより遂行されていてもよい。そして、その場合、CD-ROMやフラッシュメモリやFD等の記憶媒体により、あるいはネットワークを介して外部の記憶媒体から、プログラムを含む情報群を出力装置に供給される場合でも本発明は適用されるものである。

【0102】

以上のように、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、システムあるいは装置に供給し、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、本発明の目的が達成されることは言うまでもない。

10

20

30

40

50

【0103】

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本発明の新規な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。

【0104】

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、フロッピーディスク、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리카ード、ROM、EEPROM等を用いることができる。

【0105】

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働しているOS（オペレーティングシステム）等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。

10

【0106】

さらに、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込まれた後、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等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。

20

【0107】

【発明の効果】

以上説明したように、本発明によれば、コンフリクト処理を、例えばコンフリクトマネージャなどプリンタドライバ側で実行し、該コンフリクト処理の結果をアプリケーションに通知できる。

【図面の簡単な説明】

【図1】本発明の第1実施形態を示す印刷制御装置を適用可能な印刷処理システムの構成を説明するブロック図である。

【図2】図1に示したRAMのメモリマップの一例を示す図である。

【図3】図2に示した印刷処理関連プログラムにおけるコンフリクト処理制御部の内部処理の内容を説明するシステム概要図である。

30

【図4】本発明に係る印刷制御装置におけるコンフリクト処理における各モジュールで扱うデータの関連性を説明する図である。

【図5】本発明に係る印刷制御装置における補完処理前のコンフリクト処理ルールの例を示す図である。

【図6】本発明に係る印刷制御装置における補完処理後のコンフリクト処理ルールの例を示す図である。

【図7】本発明に係る印刷制御装置における設定変更処理関数の一例を示す図である。

【図8】本発明に係る印刷制御装置におけるデータ処理手順の一例を示すフローチャートである。

40

【図9】本発明の第2実施形態を示す印刷制御装置におけるコンフリクト処理制御部の内部処理の内容を説明するシステム概要図である。

【図10】本発明に係る印刷制御装置におけるコンフリクト処理における各モジュールで扱うデータの関連性を説明する図である。

【図11】本発明に係る印刷制御装置における他の設定変更処理関数の一例を示す図である。

【図12】本発明に係る印刷制御装置を適用可能な印刷システムで読み出し可能な各種データ処理プログラムを格納する記憶媒体のメモリマップを説明する図である。

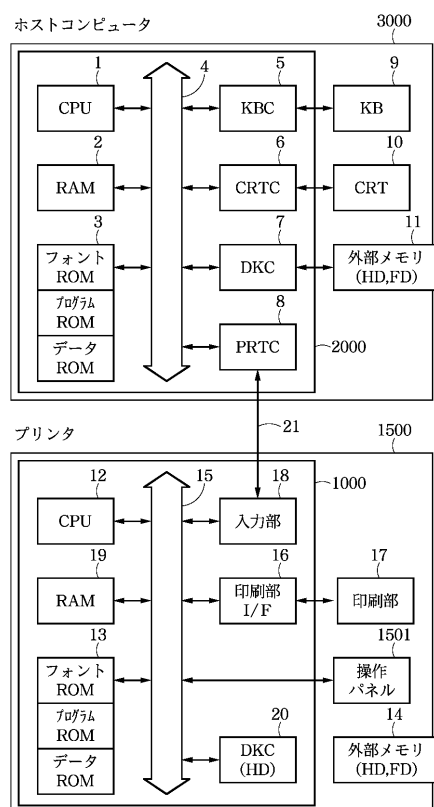
【符号の説明】

1 CPU

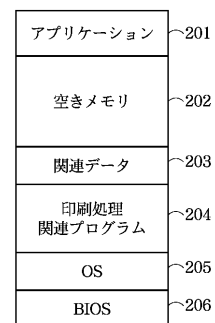
50

- 2 RAM
- 3 ROM
- 4 システムバス
- 12 CPU
- 13 ROM
- 19 RAM
- 3000 ホストコンピュータ

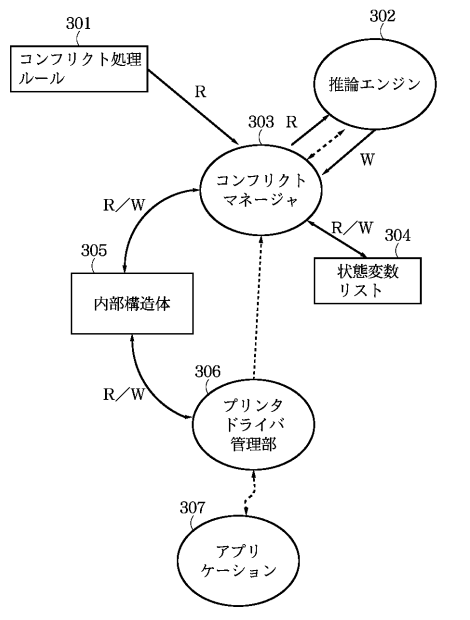
【図 1】



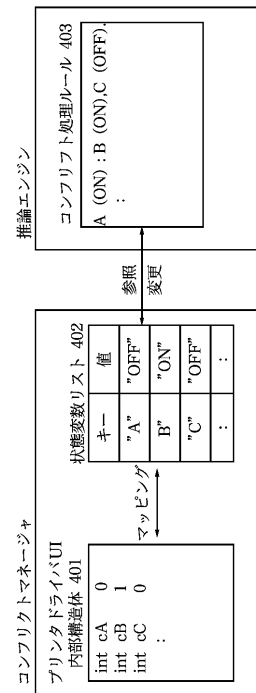
【図 2】



【図 3】



【図 4】



【図 5】

301

```

Collate(OFF) <- Group(ON)      (1)
Collate(OFF) <- Staple(ON)    (2)
Collate(OFF) <- Layout(BOOKLET) (3)
  
```

【図 6】

```

Collate(OFF) <- Group(ON)      (1)
Collate(OFF) <- Staple(ON)    (2)
Collate(OFF) <- Layout(BOOKLET) (3)

Collate(ON) <- true.          (4)
Group(ON) <- true.            (5)

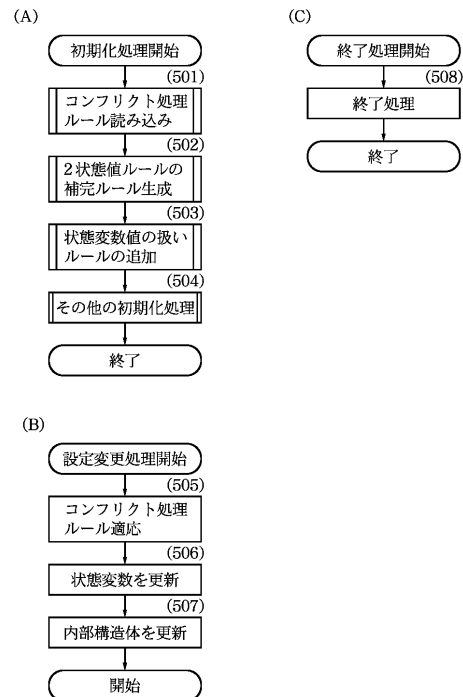
Collate(_X) <- status(Collate, _X). (6)
Group(_X) <- status(Group, _X).    (7)
Layout(_X) <- status(Layout, _X).  (8)
Staple(_X) <- status(Staple, _X).  (9)
  
```

【図 7】

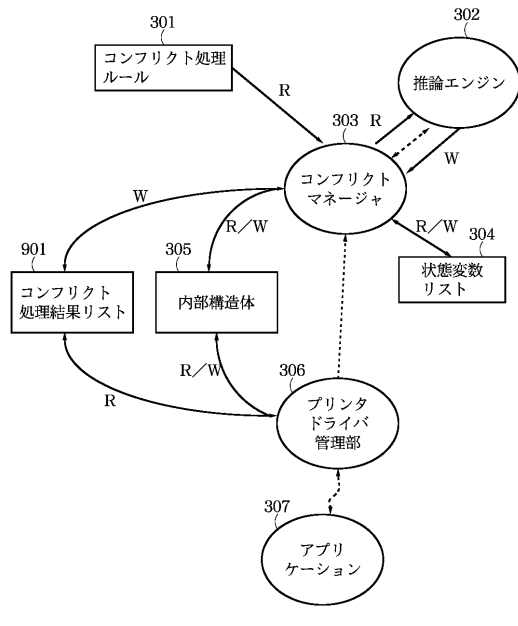
```

設定変更処理関数
LONG DocumentPropertiesEX(
    HWND hWnd,           //handle to parent window
    HANDLE hPrinter,     //handle to printer object
    LPCTSTR pDeviceName  //device name
    PDEVMODE pDevModeInOutput //original device mode &
                             //modified device mode
    DWORD dwItemID       //indicate ItemID
    LPVOID pItemParam    //Parameter indicated by ItemID
);
  
```

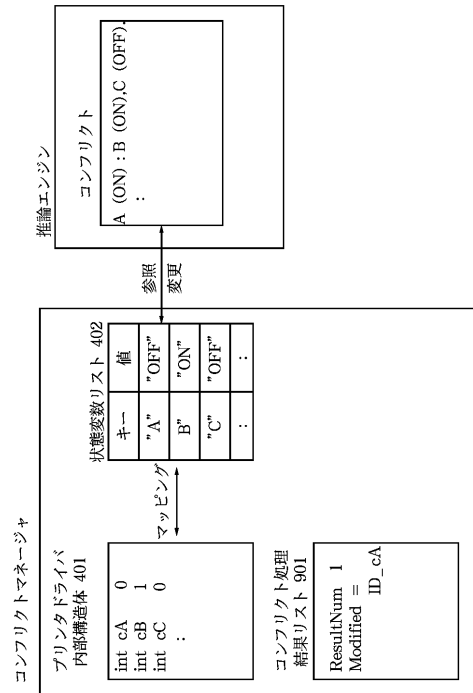
【図 8】



【図 9】



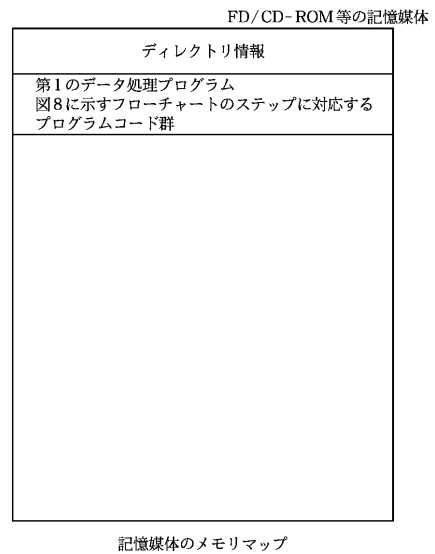
【図 10】



【図 11】

```
設定変更処理関数
LPRESULT LIST DocumentPropertiesEX(
    HWND hWnd,           //handle to parent window
    HANDLE hPrinter,      //handle to printer object
    LPTSTR pDeviceName    //device name
    PDEVMODE pDevModeInOutput //original device mode &
                             //modified device mode
    DWORD dwItemID        //indicate ItemID
    LPVOID pltemParam      //Parameter indicated by ItemID
);
```

【図 12】



フロントページの続き

- (56)参考文献 特開平11-184654 (JP, A)
特開2000-227847 (JP, A)
特開平11-126129 (JP, A)
特開2000-242597 (JP, A)

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