(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第6944281号
(P6944281)
(45)発行日　令和3年10月6日(2021.10.6)　(24)登録日　令和3年9月14日(2021.9.14)

(51)Int.Cl.　F I
H04N 1/00 (2006.01)　H04N 1/00 C
H04N 1/00 127Z

請求項の数 14　(全 20 頁)

(21)出願番号　特願2017-119890(P2017-119890)
(22)出願日　平成29年6月19日(2017.6.19)
(65)公開番号　特開2019-4429(P2019-4429A)
(43)公開日　平成31年1月10日(2019.1.10)
審査請求日　令和2年6月15日(2020.6.15)

(73)特許権者　000001007
キヤノン株式会社
東京都大田区下丸子３丁目３０番２号
(74)代理人　110003281
特許業務法人大塚国際特許事務所
(72)発明者　久野　宏一
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　野口　俊明

最終頁に続く

(54)【発明の名称】情報処理装置、情報処理方法及び情報処理システム

(57)【特許請求の範囲】
【請求項１】
情報処理装置のコンピュータに、
画像処理装置に第１スキャン要求を送信する送信ステップと、
前記第１スキャン要求により前記画像処理装置によって実行されたスキャンにより生成された画像データを取得する取得ステップと、
前記取得された前記画像データに付随する付随情報が第１の内容であることに基づいて、前記情報処理装置による所定の画像処理の実行をユーザが指示するための所定の領域を表示するように制御し、前記付随情報が前記第１の内容と異なる第２の内容であることに基づいて、前記所定の領域を表示しないように制御する制御ステップとを実行させることを特徴とするプログラム。
【請求項２】
請求項１に記載のプログラムであって、
前記画像処理装置が、前記画像データに画像処理を実行していない場合、前記付随情報は前記第１の内容であり、前記画像処理装置が、前記画像データに画像処理を実行している場合、前記付随情報は前記第２の内容であることを特徴とするプログラム。
【請求項３】
請求項１又は２に記載のプログラムであって、
前記第１スキャン要求は、前記情報処理装置のオペレーティングシステムのメーカーによって定められた標準に基づく標準ドライバを介して前記画像処理装置に送信されること

を特徴とするプログラム。
【請求項４】
請求項３に記載のプログラムであって、
前記画像データおよび前記付随情報は、前記標準ドライバを介して取得されることを特徴とするプログラム。
【請求項５】
請求項１乃至４のいずれか一項に記載のプログラムであって、
前記画像処理装置のケーパビリティに関する情報を取得するケーパビリティ取得ステップと、
前記第１スキャン要求と異なる第２スキャン要求のための設定画面を表示する表示ステップと、をさらに実行させ、
前記取得された前記画像処理装置のケーパビリティに関する情報に基づいて、前記画像処理装置に送信される第２スキャン要求のための設定画面の内容が制御されることを特徴とするプログラム。
【請求項６】
請求項３又は請求項４に従属する請求項５に記載のプログラムであって、
前記画像処理装置のケーパビリティに関する情報は、前記標準ドライバを介して取得されることを特徴とするプログラム。
【請求項７】
請求項１乃至６のいずれか一項に記載のプログラムであって、
前記第１スキャン要求により実行されるスキャンは、最小サイズのスキャンであることを特徴とすることを特徴とするプログラム。
【請求項８】
請求項１乃至７のいずれか一項に記載のプログラムであって、
前記付随情報が前記第１の内容であることに基づいて、前記第１スキャン要求と異なる第２スキャン要求のための設定画面において前記所定の領域を表示するように制御し、前記付随情報が前記第２の内容であることに基づいて、前記設定画面において前記所定の領域を表示しないように制御することを特徴とするプログラム。
【請求項９】
請求項１乃至８のいずれか一項に記載のプログラムであって、
前記所定の画像処理は、輪郭強調であることを特徴とするプログラム。
【請求項１０】
請求項１乃至９のいずれか一項に記載のプログラムであって、
前記第１スキャン要求と異なる第２スキャン要求を送信する要求送信ステップと、
前記第２スキャン要求により前記画像処理装置によって実行されたスキャンにより生成された画像データを取得するデータ取得ステップと、
前記所定の領域を表示するよう制御されており、前記所定の領域により前記所定の画像処理の実行が前記ユーザにより指示されている場合、前記画像データに前記所定の画像処理を実行する実行ステップと、をさらに実行させることを特徴とするプログラム。
【請求項１１】
請求項１乃至１０のいずれか一項に記載のプログラムであって、
前記画像データは、ＪＰＥＧデータであって、
前記付随情報は、ＥＸＩＦ情報であることを特徴とするプログラム。
【請求項１２】
請求項１乃至１１のいずれか一項に記載のプログラムであって、
アプリケーションプログラムであることを特徴とするプログラム。
【請求項１３】
情報処理装置であって、
画像処理装置に第１スキャン要求を送信する送信手段と、
前記第１スキャン要求により前記画像処理装置によって実行されたスキャンにより生成

された画像データを取得する取得手段と、
前記画像データに付随する付随情報が第１の内容であることに基づいて、前記情報処理装置による所定の画像処理の実行をユーザが指示するための所定の領域を表示するように制御し、前記付随情報が前記第１の内容と異なる第２の内容であることに基づいて、前記所定の領域を表示しないように制御する制御手段とを有することを特徴とする情報処理装置。
【請求項１４】
画像処理装置と、情報処理装置を含む情報処理システムであって、
前記情報処理装置は、
画像処理装置に第１スキャン要求を送信する送信手段を有し、
前記画像処理装置は、
前記第１スキャン要求によりスキャンを実行するスキャン手段と、
前記スキャンにより生成された画像データに、付随情報として、前記画像処理装置で実行された画像処理の有無に対応する情報を付随させる付随手段と、
前記スキャンにより生成された画像データを前記情報処理装置に送信する送信手段と、を有し、
前記情報処理装置は、さらに、
前記第１スキャン要求により前記画像処理装置によって実行されたスキャンにより生成された画像データを取得する取得手段と、
前記取得手段により取得された前記画像データに付随する前記付随情報が第１の内容であることに基づいて、前記情報処理装置による所定の画像処理の実行をユーザが指示するための所定の領域を表示するように制御し、前記付随情報が前記第１の内容と異なる第２の内容であることに基づいて、前記所定の領域を表示しないように制御する制御手段とを有することを特徴とする情報処理システム。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、画像処理装置に各種情報を送信する情報処理装置、情報処理方法及び情報処理システムに関する。
【背景技術】
【０００２】
情報処理装置のＯｐｅｒａｔｉｎｇ　Ｓｙｓｔｅｍ以下ＯＳ上で動作するアプリケーションプログラムがスキャナなどの画像処理装置にアクセスする際には、ＯＳメーカーや標準化団体が定める標準規約を用いて行うことできる。ＯＳメーカーや標準化団体はアプリケーションプログラムからの要求に応じてドライバと呼ばれるプログラムを制御し画像処理装置へのアクセスを実現する。
ドライバには画像処理装置を提供しているベンダーが自ら作り提供するベンダードライバがある。ベンダードライバは標準規約に準拠しているが、独自の処理を盛り込むことが可能である。例えばスキャナの標準規約であるＴＷＡＩＮにおいては機能拡張用にＣｕｓｔｏｍ Ｃａｐａｂｉｌｉｔｙ機能があり、ベンダーがベンダードライバにこれを定義することで独自の機能を実装し、アプリケーションプログラムから利用することが可能である。
【０００３】
しかしながら、近年のセキュリティ問題やＯＳのアップグレード期間の短縮などにより、ベンダードライバではなく、ＯＳメーカーが用意する標準ドライバを使用するケースが増えてきている。標準ドライバでは基本機能のみが実装されておりベンダー固有の独自機能はサポートされない。結果として標準ドライバによる機能不足が懸念されている。
【０００４】
標準ドライバを使いながら機能を拡張する方法はいくつか提案されている。例えば、特許文献１においては、ＴＷＡＩＮのスタブデータソースからユーザーインタフェースマネ

ージャを経由し実際のＴＷＡＩＮプロトコルマネージャ経由からきた要求に対して、画像マネージャで画像処理を加えて結果をＴＷＡＩＮプロコルマネージャ経由で変換する処理が提案されている。アプリケーションプログラムから見た実際のデータの経由はＴＷＡＩＮプロトコルに準拠しているが、ＴＷＡＩＮの制御外の仕組みをスタブでオーバーラップしているため、ＯＳから見ると標準規約の範囲内での実装ではない。

【０００５】

また特許文献２においては、デバイスドライバとアプリケーションとの間でデバイスドライバに固有の設定情報を授受するための拡張インタフェースをオペレーティングシステムのインタフェースで実現し、前記デバイスドライバの固有の設定情報に基づく設定が可能な設定画面の表示をさせている。こちらの拡張インタフェースも標準規約内で定義されたものでない別のプログラムとなっており、標準プロトコルと標準ドライバだけでの解決策にはなっていない。

【０００６】

このように、機能拡張は標準規約の範囲で納めることができず、標準プロトコルが存在し標準ドライバが提供されている環境においてはベンダーの拡張機能が使えないことがある。機能拡張を利用するためには、ベンダードライバを用いたり、従来技術で説明した技術を用いたりすることで、それが可能となる。

【０００７】

一方、スキャナ等の画像処理装置は、ホストコンピューター等に比べると処理能力が低く、組み込みプログラムであるので処理の柔軟性にも乏しい。そこで画像処理そのものはより柔軟で処理能力も高いホストコンピューター等で実行する傾向にある。

【先行技術文献】

【特許文献】

【０００８】

【特許文献１】特許第４２０８２７８号明細書

【特許文献２】特許第４３１０１７２号明細書

【発明の概要】

【発明が解決しようとする課題】

【０００９】

しかしながら、スキャナ等の画像処理装置で標準的な画像処理を超えた画像処理が施されている場合には、アプリケーションで更に画像処理を施すと、画質が劣化することがある。そのため、アプリケーションは、例えばそれを防止するために、画像処理装置から取得する画像データが既に標準を超える画像処理済みかを知る必要がある。ところが、ベンダードライバを利用すれば、画像処理装置に対する画像処理の指定を取得できるかもしれないが、標準ドライバを用いる場合、それをアプリケーションが知ることができないことがある。そのため、画像処理装置を用いる情報処理装置において、画像処理装置の利用に際して実行する機能を適切に決定できないことがある。

【００１０】

本発明は上記従来例に鑑みて成されたもので、標準的なドライバを利用しつつ、実行する機能を適切に決定することができる情報処理装置、情報処理方法及び情報処理システ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】

【００１１】

上記目的を達成するために本発明は以下の構成を有する。

本発明の一側面によれば、情報処理装置のコンピュータに、

画像処理装置に第１スキャン要求を送信する送信ステップと、

前記第１スキャン要求により前記画像処理装置によって実行されたスキャンにより生成された画像データを取得する取得ステップと、

前記取得された前記画像データに付随する付随情報が第１の内容であることに基づいて、前記情報処理装置による所定の画像処理の実行をユーザが指示するための所定の領域を

表示するように制御し、前記付随情報が前記第１の内容と異なる第２の内容であることに基づいて、前記所定の領域を表示しないように制御する制御ステップとを実行させることを特徴とするプログラムが提供される。

本発明の他の側面によれば、情報処理装置であって、

画像処理装置に第１スキャン要求を送信する送信手段と、

前記第１スキャン要求により前記画像処理装置によって実行されたスキャンにより生成された画像データを取得する取得手段と、

前記画像データに付随する付随情報が第１の内容であることに基づいて、前記情報処理装置による所定の画像処理の実行をユーザが指示するための所定の領域を表示するように制御し、前記付随情報が前記第１の内容と異なる第２の内容であることに基づいて、前記所定の領域を表示しないように制御する制御手段とを有することを特徴とする情報処理装置が提供される。

本発明のさらに他の側面によれば、画像処理装置と、情報処理装置を含む情報処理システムであって、

前記情報処理装置は、

画像処理装置に第１スキャン要求を送信する送信手段を有し、

前記画像処理装置は、

前記第１スキャン要求によりスキャンを実行するスキャン手段と、

前記スキャンにより生成された画像データに、付随情報として、前記画像処理装置で実行された画像処理の有無に対応する情報を付随させる付随手段と、

前記スキャンにより生成された画像データを前記情報処理装置に送信する送信手段と、

を有し、

前記情報処理装置は、さらに、

前記第１スキャン要求により前記画像処理装置によって実行されたスキャンにより生成された画像データを取得する取得手段と、

前記取得手段により取得された前記画像データに付随する前記付随情報が第１の内容であることに基づいて、前記情報処理装置による所定の画像処理の実行をユーザが指示するための所定の領域を表示するように制御し、前記付随情報が前記第１の内容と異なる第２の内容であることに基づいて、前記所定の領域を表示しないように制御する制御手段とを有することを特徴とする情報処理システムが提供される。

【発明の効果】

【００１２】

本発明によれば、標準的なドライバを利用しつつ、実行する機能を適切に決定することができる。

【図面の簡単な説明】

【００１３】

【図１】画像処理装置１５０と情報処理装置１００の構成図

【図２】ベンダードライバ４０１のソフトウェア構成図

【図３】標準ドライバ４０２のソフトウェア構成図

【図４】Ｃｕｓｔｏｍ Ｃａｐａｂｉｌｉｔｙを使ったベンダードライバ４０１のソフトウェア構成図

【図５】標準ドライバ４０２を使う場合のシーケンスチャート

【図６】ベンダードライバ４０１を使う場合のシーケンスチャート

【図７】アプリケーションプログラム２００のフローチャート１

【図８】原稿台だけの時のユーザーインタフェースを示す図

【図９】ＡＤＦがある場合のユーザーインタフェースを示す図

【図１０】両面ＡＤＦがある場合のユーザーインタフェースを示す図

【図１１】アプリケーションプログラム２００のフローチャート２

【図１２】画像処理がある場合のユーザーインタフェースを示す図

【図１３】アプリケーションプログラム２００のフローチャート３

【図１４】アプリケーションプログラム２００のフローチャート４
【図１５】更新ボタンがある場合のユーザーインタフェースを示す図
【図１６】画像処理装置１５０でのスキャン時のフローチャートを示す図
【図１７】アプリケーションプログラム２００でのスキャン制御のフローチャート
【発明を実施するための形態】
【００１４】
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しく説明する。尚、以下の実施の形態は特許請求の範囲に係る本発明を限定するものでなく、また本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本発明の解決手段に必須のものとは限らない。
【００１５】
本実施形態においては、スキャナ等の画像処理装置と、該画像処理装置を用いる情報処理装置を含む情報処理システムについて説明する。このような情報処理装置において、例えば、オペレーティングシステムがアップグレードなどで変更された場合など、ベンダードライバが提供されず、標準ドライバを使用しなければならないことがある。また、画像処理装置のベンダーによってはベンダードライバを提供せず、標準ドライバが利用されることもある。画像処理装置には、たとえばスキャンの都度画像処理の指定をせず、予め指定しておけばその後のスキャンによる画像データについてはその都度指定しなくとも予め指定した画像処理が施されるものがある。この様な画像処理装置を用いて、たとえば古いオペレーティングシステムの下でベンダードライバを用いて画像処理装置に標準を超える画像処理を指定しておき、新たなオペレーティングシステムの導入に伴い標準ドライバを用いて画像処理装置を利用することも考えられる。この場合、標準ドライバでその画像処理装置を利用しても、取得する画像データには指定しておいた画像処理が施されている。しかしながら、その画像データに標準を超える画像処理が施されていることを、標準ドライバを用いる情報処理装置で知ることができない場合がある。この場合、画像処理装置の利用に関する機能を適切に決定できないことがある。
【００１６】
そこで本実施形態では、標準的なドライバを利用しつつ、画像処理装置に関する機能を適切に決定するための技術について詳細に説明する。
【００１７】
＜実施形態１＞
本実施形態における画像処理装置１５０と情報処理装置１００の構成を、図１を参照して説明する。
【００１８】
●ハードウェア構成
情報処理装置１００は入力インタフェース１０２とＣＰＵ１０３、ＲＯＭ１０４、ＲＡＭ１０５、外部記憶装置１０６、出力インタフェース１０８、表示部１０７、キーボード１０１、マウス１０９、ネットワークインタフェース１１０を有する。ネットワークインタフェース１１０はネットワークケーブル１９０を介して後述するスキャナやスキャナを含む多機能機などの画像処理装置１５０と接続してある。画像処理装置１５０は、要求に応じて画像データを読み取り、後述する付随情報と共に要求元の情報処理装置１０１（あるいはそのアプリケーション）に渡す（送信する）。ＲＯＭ１０４には初期化プログラムが入っており、外部記憶装置１０６にはＯＳ（Ｏｐｅｒａｔｉｎｇ　Ｓｙｓｔｅｍ）、アプリケーションプログラム２００、標準ライブラリ３０１、ベンダードライバ４０１、標準ドライバ４０２、その他各種のデータが保存されている。ＲＡＭ１０５は外部記憶装置１０６に格納される各種プログラムがワークメモリとして使用する。
【００１９】
画像処理装置１５０はネットワークインタフェース１５１とＲＡＭ１５２、スキャナエンジン１５３、ＲＯＭ１５４、ＣＰＵ１５５を有する。ネットワークインタフェース１５１はネットワークケーブル１９０を介して情報処理装置１００と接続される。この接続は図のようにケーブルで直接接続しても、ルーターやハブなどを経由して間接的に接続して

も問題はない。また、ネットワークケーブル１９０を介さず、無線によって接続しても問題はない。ＲＡＭ１５２はＣＰＵ１５５の主メモリとワークメモリとして用いられ、受信したスキャンジョブを処理するための各種のデータを保存する。スキャンエンジン１５３はＲＡＭ１５２に保存されたスキャンジョブに基づきスキャンを行う。スキャンエンジン１５３はスキャンするための光学センサーやセンサーを駆動するモーターなどから構成される。スキャンエンジン１５３は原稿台に原稿をおいてセンサーを駆動することでスキャンする方法の他に、スキャンする原稿を自動的に送るＡｕｔｏ Ｄｏｃｕｍｅｎｔ Ｆｅｅｄｅｒ(以下ＡＤＦ)がある。またＡＤＦにも原稿の片面だけを読む片面ＡＤＦと、光学センサーを２つ使い同時に原稿の両面を読み取る両面ＡＤＦがある。ここでは例として情報処理装置１００と画像処理装置１５０の処理分担を上記のように示したが、特にこの分担形態限らず他の形態であっても構わない。

【００２０】

●ソフトウェア構成

情報処理装置１００におけるソフトウェアの構成を、図２に示す。前述の情報処理装置１００は前述のＯＳで制御されており、その上でアプリケーションプログラム２００が動作している。アプリケーションプログラム２００は画像処理装置１５０から画像を取得し保存する機能を持っているものとする。

【００２１】

アプリケーションプログラム２００がスキャナなどの画像処理装置１５０にアクセスする際には、ＯＳメーカーや標準化団体が定める標準規約を用いて行うことできる。アプリケーションプログラム２００が標準規約を用いて画像処理装置１５０にアクセスする際には、標準規約に定義されているＡｐｐｌｉｃａｔｉｏｎ Ｐｒｏｇｒａｍ Ｉｎｔｅｒｆａｃｅ（以下「標準ＡＰＩ３００」と略す）を使用する。

【００２２】

アプリケーションプログラム２００は画像処理装置１５０にアクセスするにはＯＳに含まれている標準ライブラリ３０１を自分にプログラムに組み込み、その中に定義されている標準ＡＰＩ３００を呼び出す。標準ＡＰＩ３００の呼び出しに応じて、ＯＳの標準ライブラリ３０１は画像処理装置１５０に対応するドライバをロードする。ドライバはドライバＡＰＩ４００で求められる機能に応じて、画像処理装置１５０にアクセスし所望の動作を実現する。ドライバは図２で示すようにベンダーが作成し提供するベンダードライバ４０１であっても、図３で示すように標準プロトコル６００で画像処理装置１５０にアクセスするＯＳメーカーが作成しＯＳに含まれている標準ドライバ４０２であっても構わない。標準ドライバ４０２はこの実施例ではＯＳメーカーになっているが、それ以外の標準規約にのっとったドライバであっても構わない。また、標準ＡＰＩ３００と標準ドライバ４０２を一体としてドライバが存在しない場合でもアプリケーションプログラム２００からみた場合には同様と考えてよい。

【００２３】

図２に示すベンダードライバ４０１と、図３に示す標準ドライバ４０２は、アプリケーションプログラム２００から見た場合には、標準的な機能を実装する範囲では差がない。アプリケーションプログラム２００が、ＯＳ標準で提供されるデバイス制御に対する使いやすさを向上させたり機能アップを実現したりする場合は、図４に示すようにベンダードライバ４０１とＣａｐａｂｉｌｉｔｙを用いることで実現することができる。Ｃａｐａｂｉｌｉｔｙとはたとえばベンダードライバの機種固有の機能等の情報である。図４は基本的に図２と差はないが、アプリケーションプログラム２００は、標準ＡＰＩ３００で定義されたＳｅｔ Ｃｕｓｔｏｍ Ｃａｐａｂｉｌｉｔｙ３４０とＧｅｔ Ｃｕｓｔｏｍ Ｃａｐａｂｉｌｉｔｙ３５０を使用して、それをサポートするベンダードライバ４０１に対して影響を及ぼすことができる。例えば、アプリケーションプログラム２００が、Ｇｅｔ Ｃｕｓｔｏｍ Ｃａｐａｂｉｌｉｔｙ３５０を使えば、Ｇｅｔ ＤｒｉｖｅｒＣａｐａｂｉｌｉｔｙ ａｎｄ Ｓｔａｔｕｓ３５１で示すように、画像処理装置１５０と実際に通信することなくドライバ４０１内部に持っている画像処理装置１５０の読み取り可能サイズなど

の静的な情報を得る事も可能だし、標準定義を超える細かい情報を取得することも可能である。またＳｅｔ Ｃｕｓｔｏｍ Ｃａｐａｂｉｌｉｔｙ３４０をスキャンの直前に使えば、Ｃｏｎｔｒｏｌ Ｄｒｉｖｅｒ Ｂｅｈａｖｉｏｒ３４１で示すように、ベンダードライバ４０１の内部的な動作を変え、この後のスキャン指示を上書きするような指示も可能である。ベンダードライバ４０１の内部的な動作を変える事に伴って、Ｃｏｎｔｒｏｌ Ｄｅｖｉｃｅ Ｂｅｈａｖｉｏｒ３４２、Ｇｅｔ Ｄｅｖｉｃｅ Ｓｔａｔｕｓ３５２で示すように画像処理装置１５０そのものに対するステータス取得やコマンド発行を間接的に制御することも可能である。たとえば、Ｓｅｔ Ｃｕｓｔｏｍ Ｃａｐａｂｉｌｉｔｙ３４０によりＣｏｎｔｒｏｌ Ｄｅｖｉｃｅ Ｂｅｈａｖｉｏｒ３４２を画像形成装置１５０に設定し、その中で輪郭強調を実行するよう設定できる。この設定の後には、画像処理装置１５０は、画像のスキャンの要求に応じてスキャンした画像データに輪郭強調処理を施す。

【００２４】

ただし、Ｃｕｓｔｏｍ Ｃａｐａｂｉｌｉｔｙは名前の通り、ドライバ別に作成された設定値であり、すべての画像処理装置１５０で使用できる訳ではないし、使用方法自体もドライバを作成したベンダーでなければわからない。ベンダーはＳＤＫなどの情報でそれらの機能を公開し、デベロッパーに解放したりしているが、ベンダー間で統一されている訳ではない。また標準ドライバ４０２はベンダードライバ４０１がサポートしているＣｕｓｔｏｍ Ｃａｐａｂｉｌｉｔｙを知らないので、アプリケーションプログラム２００がそれらの処理を指定しても、標準ドライバ４０２からは無視されるか、エラーとされることになる。

【００２５】

●標準ドライバを用いたスキャン処理のフロー

図５は標準ドライバ４０２を前提としたアプリケーションプログラム２００の基本的な動作に関するデータフローである。全体的な流れを記す。まずステップ２１０でユーザーがアプリケーションプログラム２００を起動する。起動されると、アプリケーションプログラム２００は、ステップ２２０で画像処理装置１５０のリスト取得を行い、見つかった画像処理装置１５０をリスト表示する。ユーザーは、ステップ２３０でそのリストからスキャンを行う画像処理装置１５０を選択する。選択されると、アプリケーションプログラムは画像処理装置１５０の持つ機能に応じたユーザーインタフェースを生成し表示する。ステップ２４０でユーザーインタフェースが表示されたら、ユーザーは、スキャンの条件を設定してから、ステップ２５０でスキャンボタンを押下する。スキャンボタンが押下されたら、アプリケーションプログラム２００はスキャンを実行し結果のデータをステップ２６０でファイルに保存し、ステップ２４０に戻って再度画像処理装置１５０の持つ機能に応じたユーザーインタフェースを表示する。ユーザーは、必要な回数だけ、ステップ２４０→２５０→２６０を繰り返してスキャンすることができる。ユーザーが所望の結果を得られたら、ユーザーはステップ２７０でアプリケーションプログラム２００を終了させる。

【００２６】

次にアプリケーションプログラム２００からの内部的なシーケンスをみていく。ステップ２２０で画像処理装置１５０リスト取得が指示されると、アプリケーションプログラム２００は標準ライブラリ３０１に対して標準ＡＰＩ３００の１つである画像処理装置１５０リスト取得要求のＧｅｔ Ｄｅｖｉｃｅ Ｌｉｓｔ３１０を発行する。Ｇｅｔ Ｄｅｖｉｃｅ Ｌｉｓｔ３１０により、標準ライブラリ３０１はＵＳＢやＮｅｔｗｏｒｋ Ｓｅｒｖｉｃｅなどに登録された情報を調べ、必要であれば 標準ドライバ４０２に対してリスト要求であるＥｎｕｍｅｒａｔｅ４１０を発行して画像処理装置１５０のリストを生成しアプリケーションプログラム２００に返す。

【００２７】

つづいて、ステップ２３０で画像処理装置１５０の選択がなされると、アプリケーションプログラム２００は該当する画像処理装置１５０でできるスキャンの情報を取得するために、最初に標準ＡＰＩ３００の１つである画像処理装置１５０使用開始を宣言するため

のＤｅｖｉｃｅ Ｏｐｅｎ３２０を発行する。Ｄｅｖｉｃｅ Ｏｐｅｎ３２０により、標準ライブラリ３０１は標準ドライバ４０２に対してＤｅｖｉｃｅ Ｏｐｅｎ４２０の要求をする。標準ドライバ４０２は画像処理装置１５０を占有させるためにＤｅｖｉｃｅ Ｏｐｅｎ６２０の要求をして結果を受け取る。画像処理装置１５０はＤｅｖｉｃｅ Ｏｐｅｎ６２０を受けると実際がその要求に応えられるかを調べ、可能な場合はＯＫを返し、次のコマンドの準備を開始する。可能でない場合「ビジー」などのエラーを返すことも可能である。結果は標準ライブラリ３０１経由でアプリケーションプログラム２００に渡される。

【００２８】

画像処理装置１５０がオープンできたら、続いてアプリケーションプログラム２００は画像処理装置１５０の機能を調べるために標準ＡＰＩ３００の１つである画像処理装置１５０機能を調べるＧｅｔ Ｃａｐａｂｉｌｉｔｙ３３０を発行する。Ｇｅｔ Ｃａｐａｂｉｌｉｔｙ３３０で取得できるＣａｐａｂｉｌｉｔｙは、画像処理装置１５０で一般的である、解像度、原稿台のサイズ、ＡＤＦの有無などの情報を含んでいる。Ｇｅｔ Ｃａｐａｂｉｌｉｔｙ３３０は、そのまま標準ライブラリ３０１からＧｅｔ Ｃａｐａｂｉｌｉｔｙ４３０として標準ドライバ４０２に、さらにＧｅｔ Ｃａｐａｂｉｌｉｔｙ６３０として画像処理装置１５０まで伝達される。画像処理装置１５０はＧｅｔ Ｃａｐａｂｉｌｉｔｙ６３０に応じて自分の持つ情報をＣａｐａｂｉｌｉｔｙとして返し、その情報は最終的にはアプリケーションプログラム２００へ伝達される。

【００２９】

ステップ２４０で、アプリケーションプログラム２００は取得された情報（Ｃａｐａｂｉｌｉｔｙ）を基にしてスキャン設定画面のユーザーインタフェースを生成し表示する。ユーザーがスキャン設定のユーザーインタフェースを操作し所望する設定値に変更し、ユーザーインタフェースの「スキャン」ボタンを押下することで、ステップ２５０のスキャン開始がキックされる。

【００３０】

スキャン開始により、アプリケーションプログラム２００は標準ＡＰＩ３００の１つであるスキャン要求のＳｃａｎ Ｗｉｔｈ Ｓｅｔｔｉｎｇ３６０を発行する。それに応じて標準ライブラリ３０１はＳｅｔ ＳｃａｎＳｅｔｔｉｎｇｓ４６１を発行し、その応答が成功であれば続けてＥｘｅｃｕｔｅ Ｓｃａｎ４６２を発行する。これによりＥｘｅｃｕｔｅ Ｓｃａｎ６６２が画像処理装置１５０に発行されてスキャン１５１が実行される。標準ライブラリはＥｘｅｃｕｔｅ Ｓｃａｎ４６２に対する応答が成功であればＧｅｔ Ｉｍａｇｅ Ｆｉｌｅ４６３を標準ドライバ４０２に発行する。これに応じてスキャンのデータＩｍａｇｅＦｉｌｅが取得されアプリケーションプログラム２００へ渡される。

【００３１】

データが取得されると、アプリケーションプログラム２００は、ステップ２６０でそのデータをファイルに保存する。保存が成功したら、アプリケーションプログラム２００はステップ２４０のスキャン設定画面のユーザーインタフェースを表示する。ユーザーは所望するスキャンが終わるまで、原稿や設定を何回か変更して、ステップ２４０からの処理を繰り返す。ユーザーがスキャンを終えたら、ステップ２７０でアプリケーションプログラム２００を終了させる。ステップ２７０では、画像処理装置１５０の使用終了を宣言し他のアプリケーションプログラム２００が画像処理装置１５０を使用できるようにするため、標準ＡＰＩ３００の１つであるＤｅｖｉｃｅ Ｃｌｏｓｅ３７０を発行する。

【００３２】

●ベンダードライバを用いたスキャン処理のフロー

図６はベンダードライバ４０１を前提としたアプリケーションプログラム２００の基本的な動作に関するデータフローである。相違点は３つある。

【００３３】

まずは、アプリケーションプログラム２００が、Ｇｅｔ Ｃａｐａｂｉｌｉｔｙ３３０の後にＧｅｔ Ｃｕｓｔｏｍ Ｃａｐａｂｉｌｉｔｙ３５０を呼んでいるところである。こ

の機能はドライバが持つ内部的な機能に関する情報を取得することができる。よってアプリケーションプログラム２００は、標準では定義されていない画像処理の有無やスキャンの方法が可能かなどの情報をここで得て、ユーザーインタフェースに反映することもできる。標準では定義されていない画像処理（すなわち標準を超える画像処理）は、たとえば標準ドライバでは対応できない画像処理であり、特に指定がない限り実行されない画像処理である。

【００３４】

さらに、ステップ２５０のスキャンボタンの押下の後で通常のスキャン動作に入る前にＳｅｔ Ｃｕｓｔｏｍ Ｃａｐａｂｉｌｉｔｙ３４０が呼ばれている。これによってＧｅｔ Ｃｕｓｔｏｍ Ｃａｐａｂｉｌｉｔｙ３５０で取得されたベンダー固有の処理をベンダードライバ４０１に伝え、それに応じた動きを行わせることができるようになる。場合によっては、通常の方法で設定された内容を上書きしてしまっても構わない。

【００３５】

最後の違いが実際にスキャンを制御し画像を作る部分である。アプリケーションプログラム２００によりＳｃａｎ ｗｉｔｈ Ｓｅｔｔｉｎｇ３６０が実行されると、Ｓｅｔ Ｃｕｓｔｏｍ Ｃａｐａｂｉｌｉｔｙ３４０，４４０での設定に従って、ベンダードライバ４０１は、標準ドライバ４０２では実行されない独自のコマンド５８０の使用やベンダードライバ４０１による画像処理など、独自処理４８０を実行する。標準ドライバ４０２を使っている場合には、スキャン処理および画像処理を行うのは画像処理装置１５０の中にあるＣＰＵ１５５である。一方、ベンダードライバ４０１の場合には画像処理装置１５０の制御は基本的にベンダードライバ４０１にゆだねられており、ケースによってすべての制御および画像処理を情報処理装置１００のＣＰＵ１０３が行っても構わない。よって画像処理装置１５０では難しい複雑な画像処理もベンダードライバ４０１を使う場合には可能になる。

【００３６】

●ユーザーインタフェース表示までの標準フロー

図７はユーザーインタフェースをアプリケーションプログラムが表示するまでの標準フローであり、標準ドライバ４０２、ベンダードライバ４０１で共通する処理である。アプリケーションプログラムがステップ２１０で起動されると、ステップ２２０で画像処理装置リストが取得される。ステップ２３０で、リストから画像処理装置１５０選択をすると、ステップ２４１で、選択された画像処理装置１５０から標準ＡＰＩ３００に標準定義されたＣａｐａｂｉｌｉｔｙを取得する。そして取得したＣａｐａｂｉｌｉｔｙの内容に応じてユーザーインタフェースを構築していく。ここでは標準定義されたＣａｐａｂｉｌｉｔｙに「ＡＤＦの有無」と「両面ＡＤＦか否か」の情報が含まれているものとする。

【００３７】

まず、アプリケーションプログラム２００はステップ７００でＡＤＦの有無を判定する。ＡＤＦがない場合には原稿台しかないので、ステップ７０２では、アプリケーションプログラム２００は原稿台８１０のみ選択可能な図８に示すようなユーザーインタフェースを生成する。ＡＤＦがある場合にはさらに、ステップ７０１で両面ＡＤＦか否かを判定し、両面非対応の場合は、ステップ７０３で、アプリケーションプログラム２００は原稿台８１０とＡＤＦ８１１のどちらかが選択可能なラジオボタンを有する、図９に示すようなユーザーインタフェースを表示する。両面対応の場合には、ステップ７０４で、アプリケーションプログラム２００は図９に加えて両面のチェックボックス８２０を追加した示す図１０に示すようなユーザーインタフェースを表示する。なお、図８、図９、図１０で示すように画像処理装置１５０はポップアップによりリストから選択されるので、画像処理装置を選択し直すこともできる。その場合には改めて選択された画像処理装置について上記手順でユーザーインタフェースを表示し直す。また、スキャンボタン８５０は押下すると、ステップ２５０のスキャンボタンを押下が発生する。

【００３８】

●スキャン指示に応じたアプリケーション処理

図１７はスキャンボタンが押下された時のアプリケーションプログラム２００が標準ＡＰＩ３００のスキャンを呼び出してイメージファイルを取得するまでの処理である。アプリケーションプログラム２００はステップ１０００においてユーザーインタフェースでＡＤＦが選択されているかを識別し、選択されていない場合は１００３で「原稿台」を選択する。ステップ１０００においてユーザーインタフェースでＡＤＦが選択されている場合は、ステップ１００１においてユーザーインタフェースで両面が選択されているか識別し、両面が選択されていない場合はステップ１００３で「片面ＡＤＦ」を選択し、両面が選択されていた場合はステップ１００４で「両面ＡＤＦ」を選択する。アプリケーションプログラム２００は、選択した設定で標準ＡＰＩ３００のＳｃａｎ ｗｉｔｈ Ｓｅｔｔｉｎｇを呼び出す。選択した設定情報はステップ１００５において、ドライバＡＰＩ４００のＳｅｔ Ｓｃａｎ Ｓｅｔｔｉｎｇｓで標準ドライバ４０２に伝達される。設定情報が標準ドライバ４０２にセットされたらステップ１００６でドライバＡＰＩ４００のＥｘｅｃｕｔｅ Ｓｃａｎを用いてスキャンの開始を指示する。その後、ドライバＡＰＩ４００のＧｅｔ Ｉｍａｇｅ Ｆｉｌｅを呼び出す。このドライバＡＰＩ４００は、スキャンが完了した時点でアプリケーションプログラム２００に結果を返す。アプリケーションプログラムはステップ１００８でデータの取得が完了したら続く処理を行う。

【００３９】

アプリケーションプログラム２００が標準ＡＰＩ３００経由で画像データを受け取る際に、例えば画像データがＪＰＥＧであればＥＸＩＦ情報のような付随的な情報を検知することが可能である。ＥＸＩＦのような付随情報に含まれる情報には、たとえば画像データの記録日時や装置のベンダー名、モデル名、解像度、色空間、スキャンの設定情報、Ｃｕｓｔｏｍ Ｒｅｎｄｅｒｅｄタグなどがある。Ｃｕｓｔｏｍ Ｒｅｎｄｅｒｅｄタグは、画像処理装置において実行される機能を示し、例えば画像処理装置で遂行された画像処理（あるいは実行された機能）の有無を示す。アプリケーションプログラム２００がＪＰＥＧを要求し、画像処理装置１５０がＪＰＥＧ画像データを生成して標準ドライバ４０２に送信する場合には、標準ライブラリ３０１は画像処理装置１５０が生成したＪＰＥＧをそのままアプリケーションプログラム２００に渡す。アプリケーションプログラム２００はＪＰＥＧのＥＸＩＦ情報から知りえた画像処理装置１５０の特性などを基準に、アプリケーションプログラム２００でサポートするべき処理を決めることが可能である。例えばアプリケーションプログラム２００は、画像処理装置１５０が生成したＪＰＥＧに埋め込まれたＥＸＩＦ情報で汎用的に定義されているＣｕｓｔｏｍＲｅｎｄｅｒｅｄタグを使って、取得したＪＰＥＧデータに対して画像処理が可能かを判断することが可能である。

【００４０】

●画像処理装置によるスキャン結果の生成処理

図１６は画像処理装置１５０がスキャン結果を生成する時のフローである。画像処理装置１５０が標準ドライバ４０２から標準プロトコル６００を経由してスキャン要求をステップ９００で受け取ったら、ステップ９０１でその要求に対してＥＸＩＦ情報を出力するか否かを判定する。例えば特定のＯＳからの要求である場合のみ出力するとか、ファイルフォーマットの指定に応じて出力するとか、自分がこの後の機能に対応する機種であるか否かなどで判断してもよい。ステップ９０１でＥＸＩＦ情報を出力すると決定したら、ステップ９０２で、タグに影響する特定の処理、たとえばスキャンした画像データに対して、追加的な処理に耐えられないような画像処理を施すか否かを識別する。ここでは特定の処理として輪郭強調を行ったか否かを判定している。ドライバからの輪郭強調処理の指定は、たとえばベンダードライバ４０１であれば、独自コマンドにより行われる。一方、標準ドライバ４０２であれば、標準機能ではないので通常してされない。画像処理装置１５０が輪郭強調をかけて画像補正した後にアプリケーションプログラム２００で再び輪郭強調をかけると輪郭強調が利きすぎて画像が乱れるおそれがあるので、ステップ９０２で輪郭強調がオンであると判定した場合には画像処理装置１５０は追加したＥＸＩＦ情報に対してＣｕｓｔｏｍ Ｒｅｎｄｅｒｅｄタグを追加し、その値を１（Ｃｕｓｔｏｍ）にする。ステップ９０２で輪郭強調がオフであると判定した場合には画像処理装置１５０は追加

したＥＸＩＦ情報に対してＣｕｓｔｏｍ Ｒｅｎｄｅｒｅｄタグを追加し、その値を０（Ｎｏｒｍａｌ）にする。

【００４１】

ＥＸＩＦにおけるＣｕｓｔｏｍＲｅｎｄｅｒｅｄタグは、特殊な画像処理をしていることを示しており、その値が１の場合にはこれ以上の画像処理を禁止するフラグである。アプリケーションプログラム２００は、ＣｕｓｔｏｍＲｅｎｄｅｒｅｄタグに０がセットされてきている画像処理装置１５０からのスキャン結果（画像データ）については、自分が画像処理をすることが可能と判断し、独自の画像処理を適用することが可能である。一方、アプリケーションプログラム２００は、ＣｕｓｔｏｍＲｅｎｄｅｒｅｄタグが１にセットされたている画像処理装置１５０からの画像データについては、既に画像処理をかけられていると判断し、その後の独自の画像処理を禁止し、画像処理のユーザー選択も不可とする。もしＣｕｓｔｏｍＲｅｎｄｅｒｅｄタグそのものがない場合には、アプリケーションプログラム２００が画像処理をかけてよいかどうか不明なため、べつの方法で該当画像処理装置１５０のスキャン画像ファイルに画像処理をしてよいかを判定するか、判定方法がない場合には画像処理ができないものとして処理する。

【００４２】

●アプリケーションによるスキャン画像データの受信処理

図１１を用いてアプリケーションプログラム２００での実装を説明する。アプリケーションプログラム２００は、図７の標準ＡＰＩ３００のＣａｐａｂｉｌｉｔｙによる処理を行った後に、続けて図１１のステップ７１０に示すように標準ＡＰＩ３００でデータを取得する。この時のスキャン設定は任意のサイズや解像度で構わない。必要なのは画像ではなくデータの付随情報であるので、スキャンでエラーにならない最小サイズのスキャンで構わない。もちろん特定の設定でスキャンを行った場合だけ所望の情報が埋め込まれるようにしてもよいし、付随情報を画像そのものに埋め込んでも問題はない。ＥＸＩＦ情報を見る場合はＪＰＥＧを指定することだけが必要だが、ＪＰＥＧ以外でもなんらかの付随的な情報が取得できるなら、そのフォーマットでも構わない。

【００４３】

図１１ではＪＰＥＧのタグからＥＸＩＦ情報を取得するために、ステップ７１１でアプリケーションプログラム２００はＪＰＥＧのヘッダーを取得する。つづいてステップ７１２でアプリケーションプログラム２００は、ＥＸＩＦタグの中にＣｕｓｔｏｍＲｅｎｄｅｒｅｄタグが存在するかをチェックする。存在しない場合は、この画像を処理してよいかいけないか不明な為、安全のため画像処理である「輪郭強調」を指定するためのコントロールはユーザーインタフェースに表示しないものとする。ＣｕｓｔｏｍＲｅｎｄｅｒｅｄタグが存在する場合には、ステップ７１３でアプリケーションプログラム２００はその値をチェックする。ＣｕｓｔｏｍＲｅｎｄｅｒｅｄタグの値が１であれば画像処理装置１５０が画像処理を禁止しているため、アプリケーションプログラム２００は画像処理である「輪郭強調」処理の指定はユーザーインタフェースに表示しない。ＣｕｓｔｏｍＲｅｎｄｅｒｅｄタグの値が０であれば情報処理装置１００側での画像処理を許しているとして、ステップ７１４でアプリケーションプログラム２００はユーザーインフェースに「輪郭強調」の指定を追加する。

【００４４】

図１２は標準ＡＰＩ３００のＣａｐａｂｉｌｉｔｙにより図１０のユーザーインタフェースが作成されている状態で、図１１のステップ７１４の処理によりアプリケーションプログラム２００が輪郭強調８３０を追加したユーザーインタフェースである。もちろん図１０に限らず、図８、図９においても、ステップ７１４の処理によってアプリケーションプログラム２００は「輪郭強調」が追加されたユーザーインタフェースを作成しうる。ユーザーインタフェースには、複数の画像処理装置から一つを選択するための選択部８００のほか、原稿台やＡＤＦ、両面スキャンといった画像処理装置１５０やドライバから取得した機能を選択する選択部、またＣｕｓｔｏｍＲｅｎｄｅｒｅｄタグに基づいて決定した輪郭強調機能の選択部８３０を含む。

【００４５】
●アプリケーションの起動から終了までの処理フロー
図１３はアプリケーションプログラム２００の起動から終了までのフローチャートである。まずステップ２１０で、アプリケーションプログラム２００が起動されると、ステップ２２０でアプリケーションプログラム２００は標準ＡＰＩ３００を使用して画像処理装置１５０リストを取得し、それを表示して、ユーザーに選択させる。ステップ２３０でユーザーが画像処理装置１５０を選択すると、アプリケーションプログラム２００は標準ＡＰＩ３００を使用して画像処理装置１５０をオープンし、標準ＡＰＩ３００を使用して画像処理装置１５０のＣａｐａｂｉｌｉｔｙを取得する(ステップ７２０)。さらにステップ７２１で、アプリケーションプログラム２００は標準ＡＰＩ３００を使って標準の方法でスキャンを行い、データを受け取りその中にある付随情報をつかってアプリケーションプログラム２００が可能な処理を判定する。本例の付随情報とは、具体的には、図１６で説明したように、画像処理装置１５０が画像データに添付して返すＣｕｓｔｏｍ Ｒｅｎｄｅｒｅｄタグの値を指す。
【００４６】
ステップ７２０とステップ７２１で受け取った情報をベースに、ステップ２４０でアプリケーションプログラム２００はユーザーインタフェース（例えば図１０或いは図１２に示す）を生成して表示し、ユーザーからの入力を受け付ける。例えばＣｕｓｔｏｍＲｅｎｄｅｒｅｄタグが１であれば、そのユーザーインタフェースでは輪郭協調の設定は受け付けず、０であれば受け付ける。ステップ２５０でユーザーがユーザーインタフェースの中の「スキャン」ボタンを押下したら、アプリケーションプログラム２００はユーザーインタフェースに設定された値に基づいて標準ＡＰＩ３００を使った標準的な方法で画像処理装置１５０からデータを受け取る。そして、ユーザーインタフェースにおいてユーザーが「輪郭強調」を指定していたら、ステップ７１５でアプリケーションプログラム２００は画像処理をデータに対して実施する。ユーザーが「輪郭強調」を指定していなかったり指定自体ができなかったりしていた場合には、データはそのままとなる。ステップ２６０でアプリケーションプログラム２００はデータを保存してスキャン動作を終了する。ユーザーは必要に応じて、ステップ２４０のユーザーインタフェースの設定からステップ２６０のスキャンデータの保存までを繰り返しても良い。ユーザーは所望するスキャンが全て終了したら、ステップ２７０で終了を選択することでアプリケーションプログラム２００は動作を完了する。
【００４７】
以上のようにして、本実施形態では、画像処理装置で標準を超える処理が行われたことを、標準ドライバを用いていてもアプリケーションは知ることができる。標準を超える処理は例えば輪郭強調を含むが、もちろん他の標準外の処理であってもよい。例えば画像の反転や回転、ぼかし、ノイズ添加、ノイズ除去、アンシャープマスク、カスタムフィルタ、特殊効果、色補正など、標準ではない処理であればどのような画像処理も候補になり得る。そしてそのよう標準ではない画像処理が実施済みであることをＣｕｓｔｏｍＲｅｎｄｅｒｅｄタグの値によりアプリケーションは知ることができるので、アプリケーションは適切な対応を行うことでき、スキャナ等の画像処理装置における画像処理と、アプリケーションによる画像処理とを両立できる。
【００４８】
＜実施形態２＞
上記実施形態では、図１３の方法は画像処理装置１５０選択毎にユーザーが意図しない画像処理装置１５０の特性を知るためのスキャン動作を行うため、ユーザーインタフェースが表示されるまでに時間がかかってしまう。前述のＣｕｓｔｏｍＲｅｎｄｅｒｅｄタグは画像処理装置１５０側での画像処理を変更するような指示するようなオプションがなければ常に同じ値がセットされるという特徴がある。本実施形態では、この特性を利用することで、前回取得した情報を画像処理装置の識別情報と関連付けて保存し、画像処理装置１５０選択時に該当画像処理装置１５０と同じ画像処理装置１５０があったら保存してあ

る情報（登録情報と呼ぶ）を使用することにより２回目以降の画像処理装置１５０選択を高速にすることができる。ただしその場合には、画像処理装置１５０選択情報がユーザー同じ名前で別の特性を持つ画像処理装置１５０にリプレースされた場合などに正しい登録情報に更新することが難しくなる。画像処理装置１５０のファームウェアなどがアップデートされ機能が変わった場合なども同様である。なお本実施形態は、図１３を図１４に、図１２を図１５に置き替える点で相違するが、他の点は実施形態１と同じであり、相違点のみを説明する。

【００４９】

●アプリケーションの起動から終了までの処理フロー（第二の例）

図１４は画像処理装置１５０の情報が変わった場合でもユーザーが対応できるように、意図的な「画像処理装置１５０に関する情報更新」が行えるアプリケーションプログラム２００のフローチャートである。画像処理装置単位で更新をさせる方法も考えられるが、図１４では一括して画像処理装置１５０を登録・更新できるものとして実装している。アプリケーションプログラム２００は起動後に自分のプリファレンスファイルを参照し前回取得した画像処理装置１５０の登録情報があるかを、ステップ７５０で判定する。初回起動時には画像処理装置１５０の登録情報はないのでステップ７５０の結果は「ない」になる。「ない」場合には、アプリケーションプログラム２００は画像処理装置１５０のリスト取得をステップ２２０で行うが、その結果をそのままユーザーに提示するのではなく、その画像処理装置１５０の持つＣａｐａｂｉｌｉｔｙを取得し登録情報に保持する。よって、標準ＡＰＩ３００の画像処理装置１５０のリストにおいてリストアップされていても、他の情報処理装置１００が使用中などで画像処理装置１５０にアクセスできない場合は、通信エラーとなって登録できない場合がある。

【００５０】

アプリケーションプログラム２００は、ステップ２２０で得た画像処理装置１５０のリストを順番に登録していくために最初の画像処理装置１５０を、ステップ７６０で内部的に選択する。画像処理装置１５０が選択されたら、ステップ７２０でアプリケーションプログラム２００は標準ＡＰＩ３００で標準定義のＣａｐａｂｉｌｉｔｙを取得する。さらにステップ７２１でアプリケーションプログラム２００は標準ＡＰＩ３００で、選択した画像処理装置にスキャンを行わせて画像データを取得し、付随する情報（たとえばＣｕｓｔｏｍ Ｒｅｎｄｅｒｅｄタグ）を得る。この２つは図１３のステップ７２０とステップ７２１と同じである。情報取得に成功したら、ステップ７３２でアプリケーションプログラム２００はこの情報を画像処理装置１５０の登録情報に追加もしくは更新する。そしてステップ７６１でアプリケーションプログラム２００は次の画像処理装置１５０を選択し、ステップ７２０からの処理を繰り返す。ステップ７６２でアプリケーションプログラム２００は画像処理装置１５０がこれ以上なくなったら処理を画像処理装置１５０の選択に戻す。

【００５１】

一方ステップ７５０で登録情報があると判定した場合には、ステップ２３０で画像処理装置を選択させる。画像処理装置１５０の選択がされた際には、選択された画像処理装置の登録情報が取得済みであることが保証されている。そこで、画像処理装置１５０の選択時には、アプリケーションプログラム２００は、ステップ７３０においてステップ７３２で保存されていた標準ＡＰＩ３００で取得された標準定義のＣａｐａｂｉｌｉｔｙを取り出し、ステップ７３１においてステップ７３２で保存されていた標準ＡＰＩ３００取得されたデータから取得された情報を取り出し、それをベースにステップ２４０で図１５に示すユーザーインタフェースを表示する。

【００５２】

画像処理装置１５０の機能アップ、同一名称のリプレース、画像処理装置１５０の追加などが行われた場合にはユーザーは図１４のステップ２２０から開始される画像処理装置１５０の登録情報を更新する必要がある。図１５はそのための更新ボタン８５０を追加したユーザーインタフェースである。ステップ７５１では、ユーザーが更新を指示するため

の指示部である更新ボタン８５０を押下したかどうかを判定し、押された場合には画像処理装置１５０の登録情報更新指示がきたものとしてステップ２２０から開始される一連の処理を行う。アプリケーションプログラム２００はこの事により毎回同じ画像処理装置１５０で情報を取得するためのスキャンをすることなく、画像処理装置１５０の追加・更新を行った場合にも対応できるものとなる。

【００５３】

さらにステップ７５１では、更新ボタン８５０の押下の受け付けに加えて、選択された画像処理装置１５０が保存された登録情報と同じものを適用してよいかアプリケーションプログラム２００が判断してもよい。この場合にも、ステップ７５１により異なると判断した場合に、ステップ２２０に分岐して画像処理装置１５０の登録情報更新を行う。同一の画像処理装置１５０かどうかの判定は、画像処理装置１５０のモデルＩＤ、ファームのバージョン、シリアル番号、ＵＵＩＤ番号などを、単独もしくは組み合わせて判断する事が可能である。アプリケーションプログラムが、更新の必要ありと判断した場合には、内部的にステップ２２０からの処理を呼び出し処理させて、該当する画像処理装置１５０の正しいユーザーインタフェースを表示する。ステップ２５０以下は図１３と同様である。

【００５４】

以上の手順により、画像処理装置を選択する都度、画像処理装置から機能情報を取得し、また、標準でない画像処理を行ったから否かを示す非標準画像処理情報（たとえばＣｕｓｔｏｍ Ｒｅｎｄｅｒｅｄタグ）を取得する。いったん取得すると、それを画像処理装置情報として登録しておく。それにより、画像処理装置の選択の都度、上記情報を取得する必要がなくなり、処理の迅速化および画像処理装置の負荷の軽減を図ることができる。さらに、実施形態１と同様に、選択した画像処理装置により取得した画像データに対して画像処理を行ってよいか否かを、用いるドライバが標準ドライバであるかベンダードライバであるかに関わらずアプリケーションは知ることができる。

【００５５】

［その他の実施例］

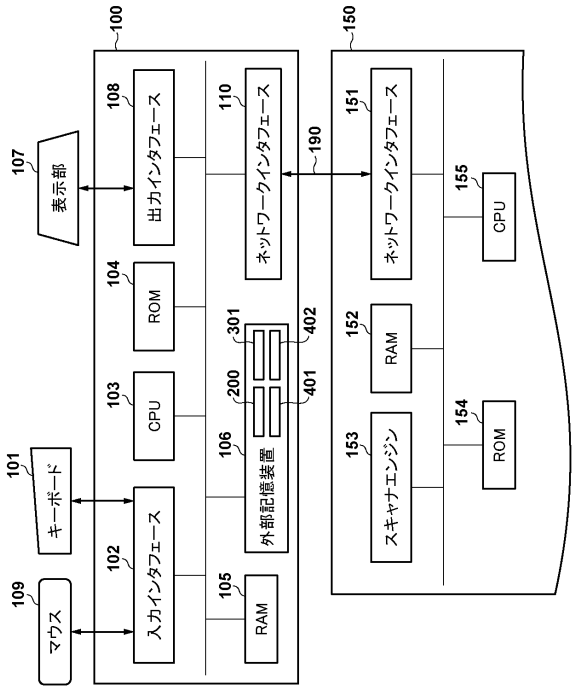
本発明は、上述の実施形態の１以上の機能を実現するプログラムを、ネットワーク又は記憶媒体を介してシステム又は装置に供給し、そのシステム又は装置のコンピュータにおける１つ以上のプロセッサーがプログラムを読出し実行する処理でも実現可能である。また、１以上の機能を実現する回路例えば、ＡＳＩＣによっても実現可能である。

【符号の説明】

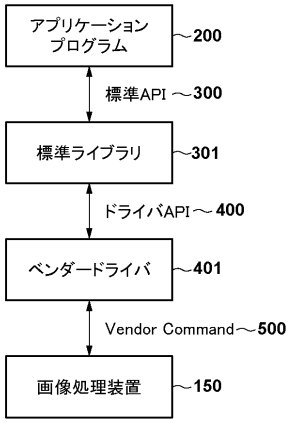
【００５６】

２００　アプリケーションプログラム、３０１　標準ライブラリ、４０１　ベンダードライバ、４０２　標準ドライバ、１５０　画像処理装置

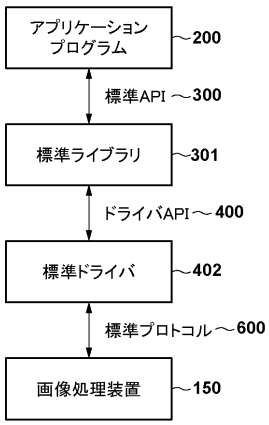
【図１】


100
150
107
表示部
108
出力インタフェース
110
ネットワークインタフェース
190
151
ネットワークインタフェース
155
CPU
104
ROM
301
402
152
RAM
103
CPU
200
401
106
外部記憶装置
154
ROM
101
キーボード
153
スキャナエンジン
102
入力インタフェース
105
RAM
109
マウス

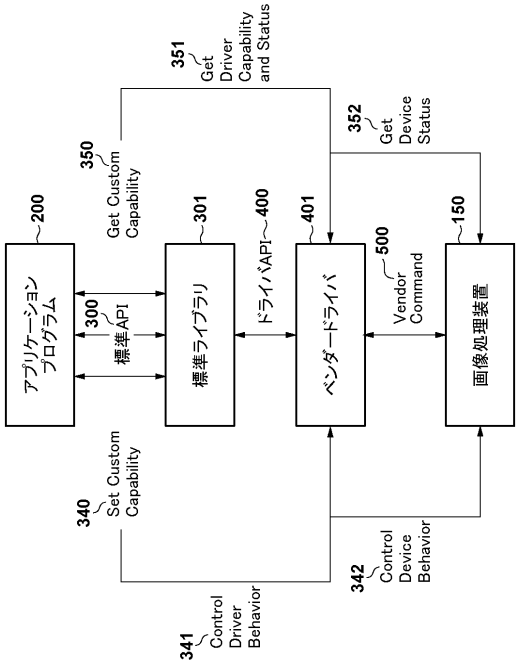
【図２】


アプリケーション
プログラム
200
標準API
300
標準ライブラリ
301
ドライバAPI
400
ベンダードライバ
401
Vendor Command
500
画像処理装置
150

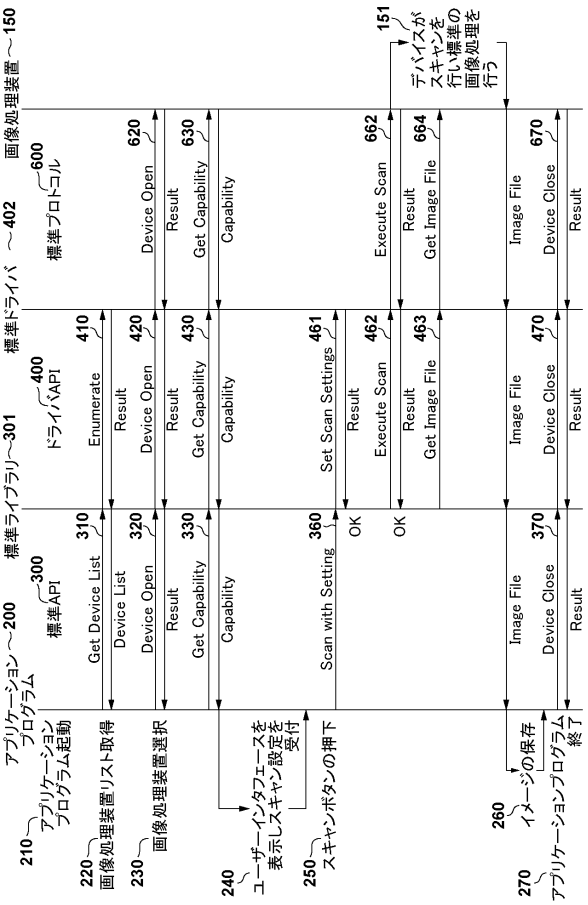
【図３】


アプリケーション
プログラム
200
標準API
300
標準ライブラリ
301
ドライバAPI
400
標準ドライバ
402
標準プロトコル
600
画像処理装置
150

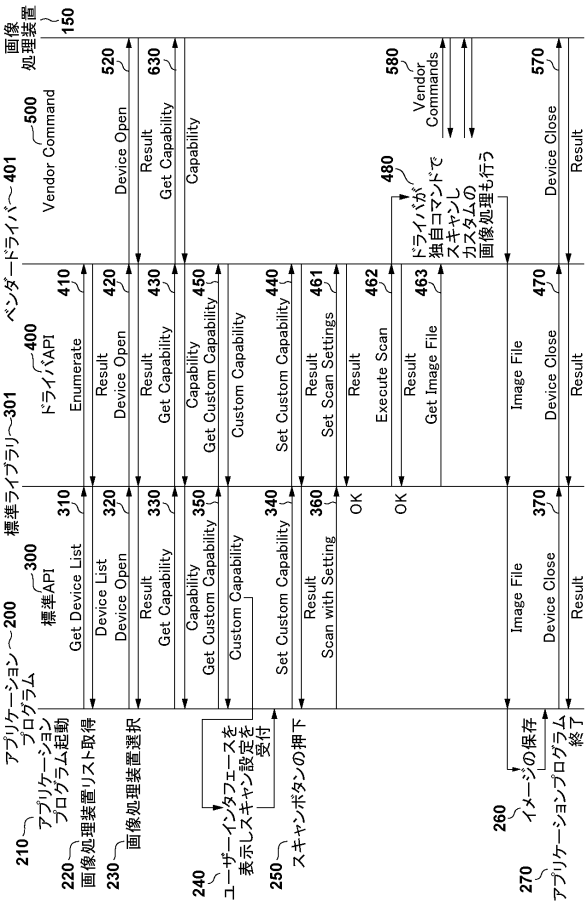
【図４】


351
Get
Driver
Capability
and Status
352
Get
Device
Status
350
Get Custom
Capability
200
301
400
401
500
150
アプリケーション
プログラム
300
標準API
標準ライブラリ
ドライバAPI
ベンダードライバ
Vendor
Command
画像処理装置
340
Set Custom
Capability
342
Control
Device
Behavior
341
Control
Driver
Behavior

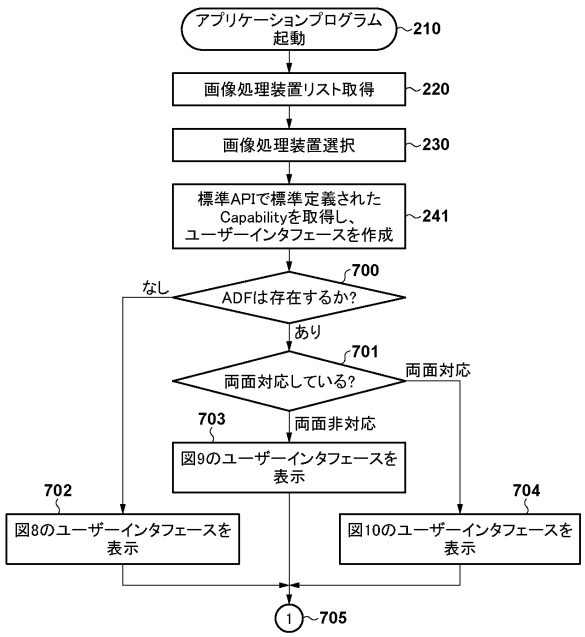
【図５】


アプリケーション プログラム
標準API
ドライバAPI
標準プロトコル
標準ドライバ
標準ライブラリ
画像処理装置
Get Device List
Enumerate
Device List
Result
Device Open
Device Open
Result
Get Capability
Capability
Capability
Set Scan Settings
Result
Execute Scan
Result
Get Image File
Image File
Device Close
Result
Scan with Setting
OK
OK
デバイスがスキャンを行い標準の画像処理を行う
アプリケーションプログラム起動
画像処理装置リスト取得
画像処理装置選択
ユーザーインタフェースを表示しスキャン設定を受付
スキャンボタンの押下
イメージの保存
アプリケーションプログラム終了

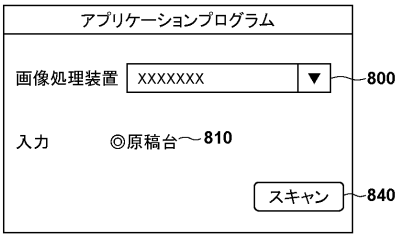
【図６】


アプリケーション プログラム
標準API
ドライバAPI
Vendor Command
ベンダードライバ
標準ライブラリ
画像処理装置
Get Device List
Enumerate
Device List
Result
Device Open
Device Open
Result
Get Capability
Capability
Capability
Get Custom Capability
Custom Capability
Set Custom Capability
Result
Set Scan Settings
Result
Execute Scan
Result
Get Image File
Image File
Device Close
Result
Scan with Setting
OK
OK
Vendor Commands
ドライバが独自コマンドでスキャンしカスタムの画像処理も行う
アプリケーションプログラム起動
画像処理装置リスト取得
画像処理装置選択
ユーザーインタフェースを表示しスキャン設定を受付
スキャンボタンの押下
イメージの保存
アプリケーションプログラム終了

【図７】


アプリケーションプログラム起動
210
画像処理装置リスト取得
220
画像処理装置選択
230
標準APIで標準定義されたCapabilityを取得し、ユーザーインタフェースを作成
241
ADFは存在するか?
700
なし
あり
画面対応している?
701
画面対応
画面非対応
703
図9のユーザーインタフェースを表示
702
図8のユーザーインタフェースを表示
704
図10のユーザーインタフェースを表示
1
705

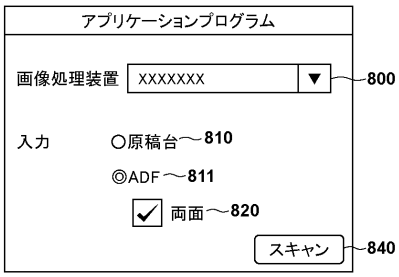
【図８】


アプリケーションプログラム
画像処理装置
XXXXXXX
800
入力
◎原稿台
810
スキャン
840

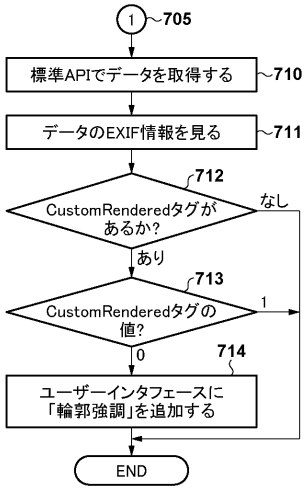
【図９】


アプリケーションプログラム
画像処理装置
XXXXXXX
800
入力
○原稿台
810
◎ADF
811
スキャン
840

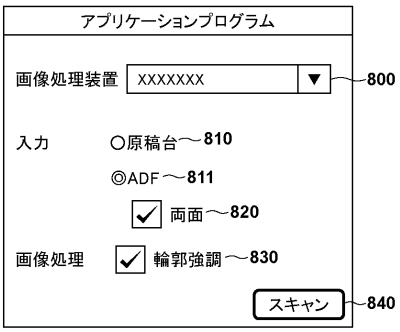
【図１０】



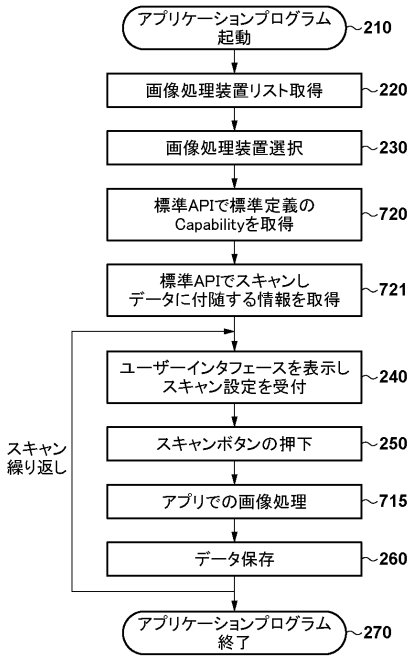
【図１１】



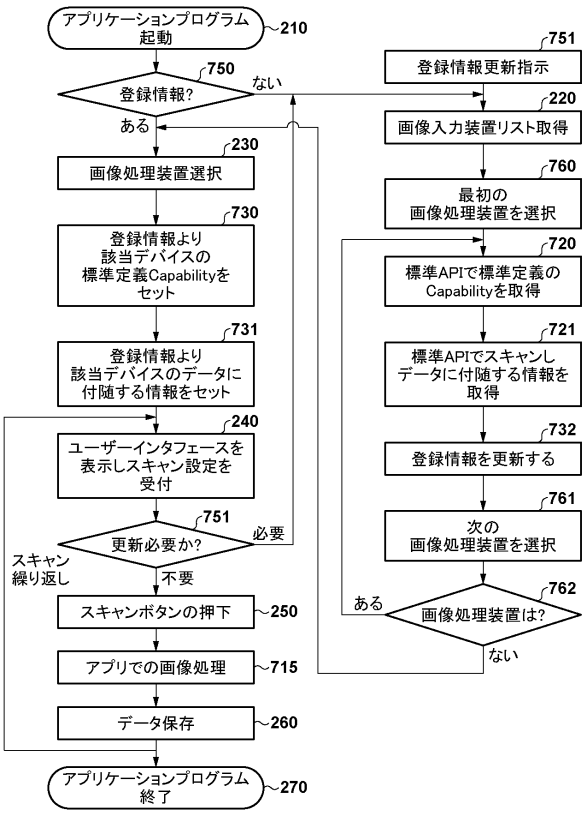
【図１２】



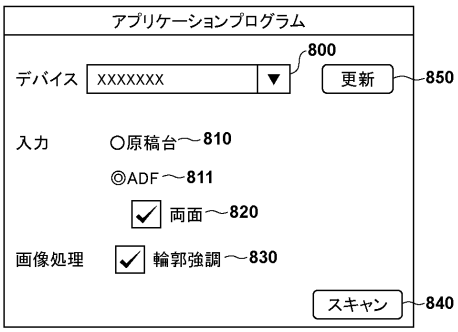
【図１３】



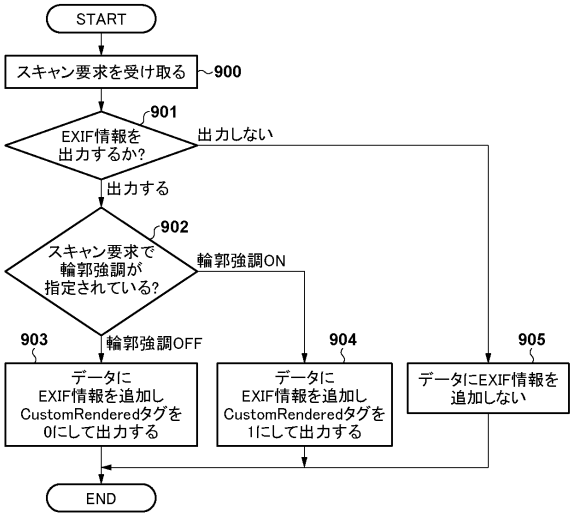
【図１４】



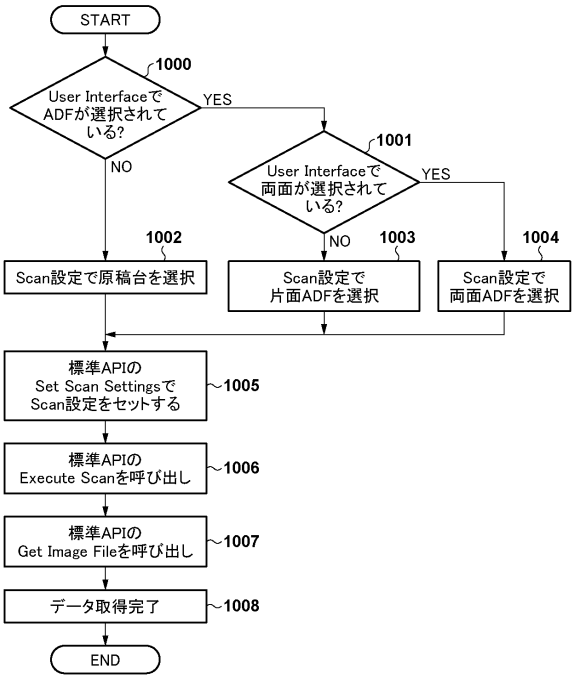
【図１５】


アプリケーションプログラム
デバイス
XXXXXXX
800
更新
850
入力
○原稿台
810
◎ADF
811
両面
820
画像処理
輪郭強調
830
スキャン
840

【図１６】


START
スキャン要求を受け取る
900
EXIF情報を出力するか?
901
出力しない
出力する
スキャン要求で輪郭強調が指定されている?
902
輪郭強調ON
輪郭強調OFF
903
データにEXIF情報を追加しCustomRenderedタグを0にして出力する
904
データにEXIF情報を追加しCustomRenderedタグを1にして出力する
905
データにEXIF情報を追加しない
END

【図１７】


START
User Interfaceで ADFが選択されている?
1000
YES
NO
User Interfaceで両面が選択されている?
1001
YES
NO
1002
1003
1004
Scan設定で原稿台を選択
Scan設定で片面ADFを選択
Scan設定で両面ADFを選択
標準APIの Set Scan Settingsで Scan設定をセットする
1005
標準APIの Execute Scanを呼び出し
1006
標準APIの Get Image Fileを呼び出し
1007
データ取得完了
1008
END

フロントページの続き

(56)参考文献 特開２００６－２４８２１７（ＪＰ，Ａ）
特開２０１４－２４１５７０（ＪＰ，Ａ）
特開２００５－１８４０９１（ＪＰ，Ａ）
特開２００４－０２３５６９（ＪＰ，Ａ）
特開２０１０－０２１５９６（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｈ０４Ｎ　　１／００