(19)日本国特許庁(JP)

(12)特許公報(B2)

(11)特許番号
特許第7549460号
(P7549460)

(45)発行日　令和6年9月11日(2024.9.11)

(24)登録日　令和6年9月3日(2024.9.3)

(51)国際特許分類		FI		
G06F	*9/451(2018.01)*	G06F	9/451	
G06F	*9/445(2018.01)*	G06F	9/445	
G06F	*9/455(2018.01)*	G06F	9/455	150
G06F	*3/12 (2006.01)*	G06F	3/12	311
		G06F	3/12	329

請求項の数　16　（全25頁）

(21)出願番号　特願2020-72372(P2020-72372)
(22)出願日　令和2年4月14日(2020.4.14)
(65)公開番号　特開2021-170173(P2021-170173A)
(43)公開日　令和3年10月28日(2021.10.28)
審査請求日　令和5年3月17日(2023.3.17)

(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
弁理士法人大塚国際特許事務所
(72)発明者　佐々木 英史
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　北川 純次

最終頁に続く

(54)【発明の名称】　画像形成装置、その制御方法、及びプログラム

(57)【特許請求の範囲】

【請求項１】

１以上の機能を提供する画像形成装置であって、

ユーザインタフェースを介して、それぞれが対応する機能の要求を受け付ける１以上の第１モジュールと、

前記１以上の第１モジュールのそれぞれに対応する機能を実行する１以上の第２モジュールと、

前記１以上の第１モジュールが受け付けた要求を、対応する前記第２モジュールへ通知する第１制御部と、

前記第１制御部からの通知に従って、対応する前記第２モジュールを制御する第２制御部と

を備え、

前記１以上の第１モジュールは、前記画像形成装置の起動時から起動され、

前記１以上の第２モジュールは、オペレーティングシステムからは独立した実行環境のコンテナとして生成され、前記第２制御部からの指示により起動状態が制御されることを特徴とする画像形成装置。

【請求項２】

前記第２制御部は、対応する機能のコンテナ管理情報に従って、前記１以上の第２モジュールの起動状態を制御することを特徴と請求項１に記載の画像形成装置。

【請求項３】

前記コンテナ管理情報には、さらに、コンテナごとに、ハードウェア資源の割り当てに関する情報が含まれることを特徴とする請求項２に記載の画像形成装置。
【請求項４】
前記第２制御部は、
対応する機能の前記コンテナ管理情報が本体連動起動と設定されていることを示す場合には、前記第１モジュールを起動する際に対応する前記第２モジュールを生成して起動し、
対応する機能の前記コンテナ管理情報が本体連動起動と設定されていることを示さない場合には、対応する前記第２モジュールを、前記第１モジュールが対応する機能の要求を受け付けた際に起動することを特徴とする請求項２又は３に記載の画像形成装置。
【請求項５】
前記第２制御部は、
前記ユーザインタフェースを介して対応する機能の設定が開始されると、対応する前記第２モジュールを生成して起動することをと特徴とする請求項４に記載の画像形成装置。
【請求項６】
前記第２制御部は、
前記画像形成装置のトレイにシートが載置されたことを検知すると、実行が予測される１以上の機能に対応する前記第２モジュールを生成して起動することを特徴とする請求項４又は５に記載の画像形成装置。
【請求項７】
人を検知する人感センサをさらに備え、
前記第２制御部は、
前記画像形成装置から所定の範囲において前記人感センサによって人が検知されると、実行が予測される１以上の機能に対応する前記第２モジュールを生成して起動することを特徴とする請求項４乃至６の何れか１項に記載の画像形成装置。
【請求項８】
前記第１モジュールは、
対応する機能の要求として該機能に対する制御コマンドを受信した際に、前記第２制御部から、対応するコンテナの状態を取得し、取得した前記コンテナの状態に従って前記第２制御部へ指示を行うことを特徴とする請求項１乃至７の何れか１項に記載の画像形成装置。
【請求項９】
前記コンテナの状態には、コンテナが生成されていない未存在状態、コンテナが起動されていない未起動状態、コンテナが起動している起動状態、及びコンテナが停止されている停止状態が少なくとも含まれることを特徴とする請求項８に記載の画像形成装置。
【請求項１０】
前記第２制御部は、前記第１制御部からの通知に従って、現在ログインしているユーザがログアウトした場合、対応する機能のジョブ終了後に該機能のジョブが投入されずに一定時間が経過した場合、他のコンテナを起動する際にハードウェア資源が所定値未満となり、かつ、使用頻度が低いコンテナである場合、又は、致命的なエラーが発生した場合に、対応するコンテナを前記起動状態から前記停止状態へ一時停止させることを特徴とする請求項９に記載の画像形成装置。
【請求項１１】
前記第１モジュールは、前記ユーザインタフェースとして、対応する機能の設定画面を制御することを特徴とする請求項１乃至１０の何れか１項に記載の画像形成装置。
【請求項１２】
前記第１モジュールは、対応する前記第２モジュールの少なくとも一部の機能を有することを特徴とする請求項１乃至１１の何れか１項に記載の画像形成装置。
【請求項１３】
前記第２モジュールは、
前記画像形成装置への前記要求を一時的にスプールする機能、及び、前記画像形成装置

への前記要求が実行可能であるか否かを判断する機能の少なくとも１つを含むことを特徴とする請求項１乃至１２の何れか１項に記載の画像形成装置。
【請求項１４】
前記要求が実行可能であるか否かを判別する機能は、
前記画像形成装置が備えるハードウェアを利用する他の機能の稼働状態に基づいて、実行可能か否かを判断することを特徴とする請求項１３に記載の画像形成装置。
【請求項１５】
ユーザインタフェースを介して、それぞれが対応する機能の要求を受け付ける１以上の第１モジュールと、前記１以上の第１モジュールのそれぞれに対応する機能を実行する１以上の第２モジュールと、を有する画像形成装置の制御方法であって、
第１制御部が、前記１以上の第１モジュールが受け付けた要求を、対応する前記第２モジュールへ通知する第１制御工程と、
第２制御部が、前記第１制御部からの通知に従って、対応する前記第２モジュールを制御する第２制御工程と
を含み、
前記１以上の第１モジュールは、前記画像形成装置の起動時から起動され、
前記１以上の第２モジュールは、オペレーティングシステムからは独立した実行環境のコンテナとして生成され、前記第２制御部からの指示により起動状態が制御されることを特徴とする画像形成装置の制御方法。
【請求項１６】
ユーザインタフェースを介して、それぞれが対応する機能の要求を受け付ける１以上の第１モジュールと、前記１以上の第１モジュールのそれぞれに対応する機能を実行する１以上の第２モジュールと、を有する画像形成装置の制御方法における各工程をコンピュータに実行させるためのプログラムであって、前記制御方法は、
第１制御部が、前記１以上の第１モジュールが受け付けた要求を、対応する前記第２モジュールへ通知する第１制御工程と、
第２制御部が、前記第１制御部からの通知に従って、対応する前記第２モジュールを制御する第２制御工程と
を含み、
前記１以上の第１モジュールは、前記画像形成装置の起動時から起動され、
前記１以上の第２モジュールは、オペレーティングシステムからは独立した実行環境のコンテナとして生成され、前記第２制御部からの指示により起動状態が制御されることを特徴とするプログラム。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、機内アプリケーションが利用するリソースをコンテナ技術を用いて管理する画像形成装置、その制御方法、及びプログラムに関する。
【背景技術】
【０００２】
画像形成装置に対するニーズが多様化する中で、必要となる機内アプリケーションが増加している。機内アプリケーションが増加すると、必要となるＣＰＵ資源、メモリ資源、ストレージ資源などのハードウェア資源も増加し、結果的には、ハードウェアコストも増加する。こうした中で、画像形成装置に搭載されたハードウェア資源を、多様化するニーズに対して有効活用できる手法が必要となっている。近年、サーバ分野では仮想化技術やコンテナ技術によるハードウェア資源を有効活用する技術が普及してきている。仮想化技術は、物理デバイス上で動作するホストＯＳ上に論理デバイスを構築し、論理デバイス上でサービスやアプリケーションを動かすためのゲストＯＳを動作させる技術である。一方、コンテナ技術は、物理デバイス上で動作するホストＯＳの資源の一部を、独立した実行環境（コンテナ）に割り当ててサービスやアプリケーションを動かす技術である。いずれ

も、ハードウェア資源をそれぞれのサービスやアプリケーションの稼働時に動的に割り当てることで、限られたハードウェア資源を有効活用しようとしている。

【０００３】

例えば、特許文献１には、仮想化技術を用いた組み込み機器の、ハードウェア資源を有効活用する技術が提案されている。詳細には、ユーザ（端末装置）からの要求に対して、ユーザ振り分け部が、当該要求に対する処理を１つ以上の仮想マシンＶＭ（Ｖｉｒｔｕａｌ　Ｍａｃｈｉｎｅ）に振り分ける技術が提案されている。仮想マシンＶＭは起動後に事前に待機状態に移行し、要求を受け付けたタイミングで稼働状態にすることで、要求を受け付けてから処理を開始するまでの時間を短縮している。

【先行技術文献】

【特許文献】

【０００４】

【文献】特開２０１８－１２９００３号公報

【発明の概要】

【発明が解決しようとする課題】

【０００５】

しかしながら、上記従来技術には以下に記載する課題がある。ニーズの多様化に対抗してリソースを有効活用すべく、組み込み機器にコンテナ技術を搭載することが考えられる。しかし、ユーザインタフェースを伴う画像形成装置ではリアルタイムの応答を確保する必要がある。例えば、上記従来技術では、仮想マシンＶＭ上で動作するアプリケーションは、ユーザインタフェースを提供する機能も包含している。即ち、設定画面や確認画面などをユーザに表示するだけでも、仮想マシンＶＭを稼働状態へ移行させ、そのうえで、画面提供をするアプリケーションを動かさなければならない。そのため、リソースがごく限られている組み込み機器に適用した場合、画面表示を要求してから、実際に表示されるまでの間に、少なからず待ち時間が生じる。人間工学では、１秒以上かかるような場合には、ユーザに応答性が低下していると認識させると言われており、待ち時間をいかに少なくするのかが重要となる。

【０００６】

特に、画像形成装置の中には、ホーム画面（又は、ポータル画面）と呼ばれるアプリケーション選択画面を備えているものがある。ユーザは、ホーム画面から１つの機能を表現したアイコンを選択することで、当該アイコンと関連付けられたアプリケーションの設定画面や実行画面を表示する機能を有している。なお、上記従来技術では、アプリケーション選択画面からアプリケーションを選択した後にアプリケーションを含む仮想マシンを停止状態から稼働状態へ移行させることとなる。停止状態から稼働状態へ移行させる処理は、アプリケーションが要するメモリ量やストレージ性能によっても異なるが、数秒程度を要する虞がある。

【０００７】

代替手法として、ユーザがログインをしたタイミングで、仮想マシンＶＭを稼働状態にしておくことも考えられる。これにより、アプリケーションを選択した際のアプリケーションの応答性を改善することができる。しかし、この場合には、ユーザがログイン要求をしてからログインが完了するまでに、数十秒程度の時間が必要となってしまう。さらに、多くのアプリケーションを稼働状態にすることで、本来のハードウェア資源の有効活用も実現困難となる。

【０００８】

本発明は、上述の課題の少なくとも一つに鑑みて成されたものであり、組み込みデバイスにコンテナ技術を好適に適用して、ハードウェア資源を有効に活用するとともに、ユーザの操作に対する応答性を担保する仕組みを提供する。

【課題を解決するための手段】

【０００９】

本発明は、例えば、１以上の機能を提供する画像形成装置であって、ユーザインタフェ

ースを介して、それぞれが対応する機能の要求を受け付ける１以上の第１モジュールと、前記１以上の第１モジュールのそれぞれに対応する機能を実行する１以上の第２モジュールと、前記１以上の第１モジュールが受け付けた要求を、対応する前記第２モジュールへ通知する第１制御部と、前記第１制御部からの通知に従って、対応する前記第２モジュールを制御する第２制御部とを備え、前記１以上の第１モジュールは、前記画像形成装置の起動時から起動され、前記１以上の第２モジュールは、オペレーティングシステムからは独立した実行環境のコンテナとして生成され、前記第２制御部からの指示により起動状態が制御されることを特徴とする。

【発明の効果】

【００１０】

本発明によれば、組み込みデバイスにコンテナ技術を好適に適用して、ハードウェア資源を有効に活用するとともに、ユーザの操作に対する応答性を担保することができる。

【図面の簡単な説明】

【００１１】

【図１】一実施形態に係る画像形成装置１００のハードウェア構成を示す図。
【図２】一実施形態に係る画像形成装置１００のソフトウェア構成を示す図。
【図３】一実施形態に係る制御コマンド／制御レスポンスのフォーマットの一例を示す図。
【図４】一実施形態に係る制御コマンドと制御レスポンス識別子の定義の一例を示す図。
【図５】一実施形態に係るデバイス・ジョブ制御部２１１によるコンテナ制御の流れを示す図。
【図６】一実施形態に係るコンテナ制御に際して参照されるＤｏｃｋｅｒｆｉｌｅ５１２、及びコンテナ管理情報ＤＢ５０１の一例を示す図。
【図７】一実施形態に係るデバイス・ジョブ制御部２１１によるアプリ制御の流れを示す図。
【図８】一実施形態に係るデバイス・ジョブ制御部２１１のフローチャート。
【図９】一実施形態に係る仮受信部２１２～２１５のフローチャート。
【図１０】一実施形態に係るコンテナ制御部２２１のフローチャート。

【発明を実施するための形態】

【００１２】

以下、添付図面を参照して実施形態を詳しく説明する。なお、以下の実施形態は特許請求の範囲に係る発明を限定するものでない。実施形態には複数の特徴が記載されているが、これらの複数の特徴の全てが発明に必須のものとは限らず、また、複数の特徴は任意に組み合わせられてもよい。さらに、添付図面においては、同一若しくは同様の構成に同一の参照番号を付し、重複した説明は省略する。

【００１３】

＜画像形成装置のハードウェア構成＞

以下では、本発明の一実施形態について説明する。まず、図１を参照して、本実施形態に係る画像形成装置１００のハードウェア構成について説明する。

【００１４】

画像形成装置１００は、ＰＣ１５１と接続される。画像形成装置１００は、ＰＣ１５１を介してユーザからの操作指示を受け付けることができる。また、画像形成装置１００は、ＵＳＢメディア１５２を始めとするＵＳＢデバイスを接続することができ、接続されたＵＳＢデバイスと通信できる。画像形成装置１００は、ＮＷ　ＨＵＢ１５３を始めとするネットワークデバイスと接続される。例えば、画像形成装置１００は、ＰＣ１５１と直接ネットワーク接続をしてもよい。或いは、画像形成装置１００は、ルータやスイッチなどを始めとする種々のネットワークデバイスを介して間接的にネットワーク接続する構成を適用することができる。

【００１５】

画像形成装置１００は、コントローラ基板１１０、電源スイッチ１３０、操作部ユニット１３１、プリンタユニット１３２、スキャナユニット１３３、及びＦＡＸユニット１３

４を備える。画像形成装置１００は、ＰＣ１５１やＵＳＢメディア１５２、ＮＷ　ＨＵＢ１５３、ＦＡＸ回線１５４などを介して、本体外に存在するデバイスと通信を行う機能を有する。コントローラ基板１１０は、画像形成装置１００を制御するためのコントローラボードである。コントローラ基板１１０は、ＣＰＵ１１１、ＲＯＭ１１２、ＲＡＭ１１３、及び内蔵ストレージ１１４を備える。また、コントローラ基板１１０は、画像形成装置１００内のユニット等と通信するために、操作部ユニットＩ／Ｆ１２１、プリンタユニットＩ／Ｆ１２２、スキャナユニットＩ／Ｆ１２３、及びＦＡＸユニットＩ／Ｆ１２４を備える。さらに、コントローラ基板１１０は、画像形成装置１００外と通信するために、ＵＳＢ－ＤＩ／Ｆ１４１、ＵＳＢ－Ｈ　Ｉ／Ｆ１４２、及びＬＡＮ　Ｉ／Ｆ　１４３を備える。

【００１６】

ＣＰＵ１１１は、画像形成装置１００の種々の機能を提供するための中央演算プロセッサである。ＣＰＵ１１１は、ＲＯＭ１１２や内蔵ストレージ１１４に格納されたプログラムやプログラムに関連したデータをＲＡＭ１１３上に展開し、当該プログラムを実行することによって、画像形成装置１００の種々の機能をユーザに対して提供する。ＲＯＭ１１２は、電源スイッチ１３０が押下され、コントローラ基板１１０に電源投入されたタイミングで、ＣＰＵ１１１が実行する初期プログラムが格納される記憶領域である。初期プログラムは、後述するＲＡＭ１１３や内蔵ストレージ１１４を始めとするコントローラ基板１１０上の各種デバイスを利用可能にするための一連の処理を含む。

【００１７】

ＲＡＭ１１３は、ＣＰＵ１１１が実行するプログラムや、プログラムが利用するデータを一時的に格納したりするための、記憶領域である。ＲＡＭ１１３は、後述する内蔵ストレージ１１４に比べると、速度は高速にアクセス可能である反面、容量は数分の一程度と小さい。そのため、ＲＡＭ１１３は、内蔵ストレージ１１４に対するアクセスを一時的にため込むディスクキャッシュに用いられることもある。内蔵ストレージ１１４は、画像形成装置１００が機能提供するためのプログラムや、プログラムが利用するデータを保存するための、記憶領域である。内蔵ストレージ１１４は、ＲＡＭ１１３に比べると、容量は非常に大きい分、アクセス速度は数分の一程度と遅い。そのため、内蔵ストレージ１１４は、プログラムが利用するメモリが不足した場合に、一時的にメモリの代替領域（スワップ領域）に用いられることもある。

【００１８】

操作部ユニットＩ／Ｆ１２１は、電源スイッチ１３０や操作部ユニット１３１と通信を行うためのインタフェースである。プリンタユニットＩ／Ｆ１２２は、プリンタユニット１３２と通信を行うためのインタフェースである。スキャナユニットＩ／Ｆ１２３は、スキャナユニット１３３と通信を行うためのインタフェースである。ＦＡＸユニットＩ／Ｆ１２４は、ＦＡＸユニット１３４と通信を行うためのインタフェースである。ＵＳＢ－ＤＩ／Ｆ１４１は、ＰＣ１５１と通信を行うためのインタフェースである。ＵＳＢ－Ｈ　Ｉ／Ｆ１４２は、各種ＵＳＢデバイスと通信を行うためのインタフェースである。図１では、ＵＳＢメディア１５２と接続する例を示している。ＬＡＮ　Ｉ／Ｆ　１４３は、有線ネットワークを用いて外部デバイスと通信を行うためのインタフェースである。図１では、ＮＷ　ＨＵＢ１５３に接続し、その先に接続されている不図示の外部デバイスと間接的に接続されている例を示している。

【００１９】

電源スイッチ１３０は、画像形成装置１００に対する電源投入や、電源切断などを指示するためのユニットである。電源スイッチ１３０は、例えばシーソースイッチを適用することができる。或いは、タクトスイッチを適用することができる。また、電源スイッチ１３０を、後述する高速復帰の電源切断モードへ突入したり、高速復帰の電源切断モードから復帰したりするためのイベントトリガーとして用いてもよい。

【００２０】

操作部ユニット１３１は、画像形成装置１００に対するユーザからの指示を受け付けたり、ユーザに対して情報や警告を報知するためのユニットである。例えば、操作部ユニッ

ト１３１は画像によって情報を報知するためにＬＣＤパネルを備えてもよい。操作部ユニット１３１は音によって情報を報知するためにブザーを備えてもよい。操作部ユニット１３１は光によって情報を報知するためにＬＥＤを備えてもよい。また、操作部ユニット１３１はユーザの指示を受け付けるための物理キーやタッチパネルを備えてもよい。その他、操作部ユニット１３１はジェスチャー入力を受け付けるためのカメラなどを備えてもよい。

【００２１】

プリンタユニット１３２は、ユーザの指示に基づいて、画像形成装置１００が物理用紙に情報を印字するためのユニットである。本発明を実施する上で、プリンタユニット１３２には、様々な印字機能を有するユニットを適用できる。様々な印字機能としては、例えば、カラー／モノクロ印刷、片面／両面印刷、ステイプル／針なしステイプルなどが含まれる。プリンタユニット１３２において、印字対象となる出力媒体は一般的な用紙にとどまらず、フィルムなどの媒体を対象にすることもできる。プリンタユニット１３２において、印字時に用いる色材はトナーのみならず、インクなどを用いることもできる。或いは、立体物を出力するプリンタユニットを備えた画像形成装置に対しても、本発明を適用することができる。

【００２２】

スキャナユニット１３３は、ユーザの指示に基づいて、画像形成装置１００が物理用紙に印字された情報を読み取るためのユニットである。本発明を実施する上で、スキャナユニット１３３には、様々な読み取り機能を有するユニットを適用できる。例：自動両面読み取り／１パス両面読み取り／カラー読み取り。

【００２３】

ＦＡＸユニット１３４は、ユーザの指示に基づいて、画像形成装置１００が機内に格納した画像データを、ＦＡＸ回線１５４を介して外部のデバイスへ送信するためのユニットである。また、画像形成装置１００が外部のデバイスから送信された画像データをＦＡＸ回線１５４を介して受信するためのユニットでもある。

【００２４】

＜ソフトウェア構成＞

次に、図２を参照して、本実施形態に係る画像形成装置１００のソフトウェア構成について説明する。コントローラ基板１１０上のＣＰＵ１１１は、内蔵ストレージ１１４に格納された本体ファーム２００をＲＡＭ１１３上に展開し、当該本体ファーム２００の内容に従って機能を実行する。

【００２５】

本体ファーム２００は、操作部制御部２０１、デバイス・ジョブ制御部２１１、及びコンテナ制御部２２１を備える。また、本体ファーム２００は、ＵＳＢ－Ｄ制御部２０７、ＵＳＢ－Ｈ制御部２０８、及びＬＡＮ制御部２０９を備える。操作部制御部２０１は、ＵＩアプリ部２０２～２０５の制御を行う制御モジュール（第１モジュール）である。ＵＩアプリ部には、例えばコピー機能を提供するためのコピーＵＩアプリ部２０２が含まれる。さらに、ＵＩアプリ部には、ＳｅｎｄＵＩアプリ部２０３、ＦＡＸＵＩアプリ部２０４、ＷｅｂブラウザＵＩアプリ部２０５、及びＵＳＢＤＩＲＥＣＴ　ＵＩアプリ部２０６が含まれる。

【００２６】

コピーＵＩアプリ部２０２は、操作部制御部２０１を介して、ユーザからの指示に基づき、スキャナユニット１３３で読み取った画像データを、プリンタユニット１３２で複写するためのコピー機能に関するアプリケーション操作部である。コピーＵＩアプリ部２０２は、部数の指定や両面指定、ステイプルの有無などを選択する機能を有し、これらの設定を行う設定画面を表示制御する。ＳｅｎｄＵＩアプリ部２０３は、ユーザからの指示に基づき、スキャナユニット１３３で読み取った画像データを、ＬＡＮ制御部２０９を介して、外部デバイスへ送信するためのスキャン機能に関するアプリケーション操作部である。ＳｅｎｄＵＩアプリ部２０３は、送信する宛先などを指定する機能を有する。ＦＡＸＵ

Ｉアプリ部２０４は、ユーザからの指示に基づき、スキャナユニット１３３で読み取った画像データを、ＦＡＸユニット制御部２８２を介して、外部デバイスへ送信するためのＦＡＸ機能に関するアプリケーション操作部である。また、ＦＡＸＵＩアプリ部２０４は、外部デバイスからＦＡＸユニット制御部２８２に対して画像データが送信された場合に、その内容を表示したり、プリンタユニット１３２で印字したりする機能を有する。Ｗｅｂブラウザ ＵＩアプリ部２０５は、ＬＡＮ制御部２０９を介して外部デバイスから取得したＷｅｂページをユーザへ表示したり、当該Ｗｅｂページ上のコンテンツへの操作を受け付けたりする機能に関するアプリケーション操作部である。ＵＳＢ Ｄｉｒｅｃｔ ＵＩアプリ部２０６は、ユーザからの指示に従い、ＵＳＢ－Ｄ制御部２０７を介して外部ＵＳＢデバイスに記録されている印刷データを読み出し、プリンタユニット１３２で印字するための機能に関するアプリケーション操作部である。本発明では、２０２～２０５のアプリケーション操作部が、それぞれのアプリケーションの実行部２４１～２５５とは独立して起動や動作が可能な構成をとる。つまり、本実施形態では、画像形成装置１００が提供する各機能を実現する各アプリケーションについて、そのＵＩを制御するアプリケーション操作部と、各機能を実行するための実行部とが独立して設けられる構成となる。

【００２７】

デバイス・ジョブ制御部２１１は、画像形成装置１００全体を制御するための制御モジュール（第１制御部）である。本実施形態においては、各ＵＩアプリ部２０２～２０６からの指示に基づき、各ジョブ仮受信部２１２～２１５や、ＦＡＸジョブ受信部２１６に当該指示を伝達する役割などを担う。また、本実施形態においては、ＵＳＢ－Ｄ制御部２０７とＵＳＢ－Ｄ Ｉ／Ｆ１４１を介してＰＣ１５１から指示を受け付けることもできる。さらに、本実施形態においては、ＬＡＮ制御部２０９とＬＡＮ Ｉ／Ｆ１４３を介して、ＮＷ ＨＵＢ１５３を始めとする、不図示の外部デバイスから指示を受け付けることもできる。なお、デバイス・ジョブ制御部２１１は、コンテナ化していないアプリケーションに対しての指示をジョブ受信部へ伝達することもある。例えば、ＦＡＸに関する処理に関しては、ＦＡＸジョブ受信部２１６へ伝達する。ＦＡＸジョブ受信部２１６は、ＦＡＸジョブ実行部２８１に指示を伝達して実行する。これは、ＦＡＸの送受信に関する処理に関しては、画像形成装置１００が動作中には常時動かすことが求められるため、コンテナの上ではなく直接動作する形態をとる。このように、本実施形態によれば、全ての機能アプリケーションに対して一律でコンテナ技術を適用するのではなく、画像形成装置１００が提供する機能の特性に合わせて適用を行う。

【００２８】

コンテナ制御部２２１は、コンテナ２２２～２２５（第２モジュール）の制御を行う制御モジュール（第２制御部）である。例えば、本体ファームウェア２００はコピー機能を実行するためのコピーコンテナ２２２を備える。コピーコンテナ２２２は、画像形成装置１００のオペレーティングシステムからは独立した実行環境のコンテナ（第２モジュール）として生成され、コピージョブ仮受信部２１２からの指示を受け付けるためのコピージョブ本受信部２３１と、受け付けた指示を実行するためのコピージョブ実行部２４１とを備える。コピージョブ実行部２４１は、プリンタユニット制御部２７１とプリンタユニットＩ／Ｆ１２２を介して、プリンタユニット１３２に印字指示を行う。同様に、コピージョブ実行部２４１は、スキャナユニット制御部２７２とスキャナユニットＩ／Ｆ１２３を介して、スキャナユニット１３３に画像読み取り指示を行う。

【００２９】

また、本体ファームウェア２００はＳｅｎｄ機能を実行するためのＳｅｎｄコンテナ２２３、ＰＤＬ機能を実行するためのＰＤＬコンテナ２２４、及びＷｅｂブラウザ機能を実行するためのＷｅｂブラウザコンテナ２２５を備える。Ｓｅｎｄコンテナ２２３は、Ｓｅｎｄジョブ仮受信部２１３からの指示を受け付けるためのＳｅｎｄジョブ本受信部２３２と、受け付けた指示を実行するためのＳｅｎｄジョブ実行部２４２を備える。ＰＤＬコンテナ２２４は、ＰＤＬジョブ仮受信部２１４からの指示を受け付けるためのＰＤＬジョブ本受信部２３３と、受け付けた指示を実行するためのＰＤＬジョブ実行部２４３を備える

。Ｗｅｂブラウザコンテナ２２５は、Ｗｅｂブラウザジョブ仮受信部２１５からの指示を受け付けるためのＷｅｂブラウザジョブ本受信部２３４と、受け付けた指示を実行するためのＷｅｂブラウザジョブ実行部２４４を備える。

【００３０】

また、本体ファーム２００は、プリンタユニット制御部２７１、スキャナユニット制御部２７２、及びＦＡＸユニット制御部２８２を備える。これらは、それぞれのユニットを制御するためのモジュールである。

【００３１】

＜制御コマンド／制御レスポンス＞

次に、図３を参照して、本実施形態に係る制御コマンド／制御レスポンスのフォーマットの一例について説明する。本実施形態では、説明を簡略化するために、本体内外で流通する、制御を実行する要求を意味する制御コマンドと、制御を実行した結果を意味する制御レスポンスを同一フォーマットで表現している。しかし本発明を限定する意図はなく、当該制御コマンド／制御レスポンスの形式や内容に関しては種々の内容を適用することができる。なお、本実施形態においては、制御コマンド／制御レスポンスは、ネットワークバイトオーダーの順番に従ってデータが配置されているものとする。

【００３２】

先頭＋０ｈに位置する、制御コマンドバージョンは、当該データの制御コマンド及び制御レスポンスのバージョンを意味する。ここでは、０ｘＣ０ＣａＦＦ０１としている。これは画像形成装置１００が受信した制御コマンドが、サポートしているバージョンであるかを判別することに利用できる。

【００３３】

先頭＋４ｈに位置する、制御コマンド／制御レスポンス識別子は、本制御コマンド／制御レスポンスの内容を意味する識別子である。制御コマンド識別子／制御レスポンス識別子はともに最大３２ｂｉｔデータで表現する。なお、本実施形態では将来の拡張性を考慮して、上位１６ｂｉｔ／２バイトは予約とし、下位１６ｂｉｔ／２バイトで表現されるデータを用いる。制御コマンド識別子／制御レスポンス識別子については、図４を用いて詳細を説明する。

【００３４】

先頭＋８ｈに位置する、ユーザ識別子は当該制御コマンドを送信した結果である制御レスポンスを受け取ろうとする送受信先を意味する識別子である。例えば、機器内外のどのアプリケーションやモジュールであるかを特定することのできる識別子となる。

【００３５】

先頭＋１０ｈに位置する、制御コマンド／制御レスポンスの固有データ長と、先頭＋１８ｈに位置する、制御コマンド／制御レスポンスの固有データボディ部は、制御コマンド／制御レスポンス識別子で表現しきれない情報を表現するための情報である。制御コマンドは様々な目的や用途で使われるため、全ての制御コマンド／制御レスポンスで共通的に利用可能なフォーマットを定義することは難しい。そこで、本実施形態においては、制御コマンド識別子に基づいて当該制御コマンド／制御レスポンスの固有データを伝達することのできる、データ長及びボディ部（中身）を伝達する形態をとる。

【００３６】

＜制御コマンド＞

以下では、図４乃至図７を参照してデバイス・ジョブ制御部２１１が扱う各種の制御コマンドについて説明する。まず、図４を参照して、本実施形態に係る制御コマンド／制御レスポンス識別子の定義の一例を説明する。制御コマンド識別子は、対象となるモジュールを特定するためのターゲットオプションを表現する下位バイトと、当該対象モジュールに対してどのような指示をするのかを特定するための制御コマンド種別を表現する上位バイトとを含む。

【００３７】

（ターゲットオプション）

ターゲットオプションは、指示対象となっている各アプリケーションを表現する情報である。例えばコピーアプリケーションに対する指示にはＣＣ＿ＴＡＲＧＥＴ＿ＣＯＰＹ（０ｘ１０）を割り当てる。Ｓｅｎｄアプリケーションに対する指示にはＣＣ＿ＴＡＲＧＥＴ＿ＳＥＮＤ（０ｘ２０）を割り当てる。ＰＤＬアプリケーションに対する指示にはＣＣ＿ＴＡＲＧＥＴ＿ＰＤＬ（０ｘ３０）を割り当てる。ウェブブラウザアプリケーションには、ＣＣ＿ＴＡＲＧＥＴ＿ＷＥＢＢＲＯＷ（０ｘ４０）を割り当てる。ＦＡＸアプリにはＣＣ＿ＴＡＲＧＥＴ＿ＦＡＸ（０ｘ５０）を割り当てる。本実施形態においては、画像形成装置１００にこれらの予め組み込まれたアプリケーションに加えて、ユーザが任意でインストールしたアプリケーションを対象にすることもできる。例えば、本実施形態では０ｘ８０から０ｘＦＥまでの範囲内をユーザアプリケーション用に割り当てており、インストールしたタイミングで任意の数値を割り当てることもできる。さらに、デバイス本体への指示に対して、ＣＣ＿ＴＡＲＧＥＴ＿ＤＥＶＩＣＥ（０ｘＦＦ）を割り当てる。これ以外にも、デバイスに備えられたフィーダーなどを始めとする外付けオプションに指示するべく、ターゲットオプションを拡張できる。

【００３８】

制御コマンド種別には、受信部に対する制御コマンド識別子、コンテナに対する制御コマンド識別子、アプリケーションに対する制御コマンド識別子、及び、デバイスに対する制御コマンド識別子が含まれる。また、制御レスポンス識別子は、汎用制御レスポンス識別子と、コンテナ状態制御レスポンス識別子とに分けられる。

【００３９】

（受信部制御コマンド）

受信部に対する制御コマンド識別子は、０ｘＦ０～０ｘＦＦから始まる。本実施形態では、受信部起動指示（ＣＣ＿ＲＥＣＥＰＴＯＲ＿ＣＭＤ＿ＳＴＡＲＴ、０ｘＦ０００）は受信部に対する起動指示を意味する。例えば、０ｘＦ０１０のコマンド識別子は、ＣＣ＿ＴＡＲＧＥＴ＿ＣＯＰＹと、ＣＣ＿ＲＥＣＥＰＴＯＲ＿ＣＭＤ＿ＳＴＡＲＴのＯＲであることから、コピーアプリケーションの受信部起動を意味する制御コマンド識別子となる。

【００４０】

（コンテナ制御コマンド）

コンテナに対する制御コマンド識別子は、０ｘＣ０～０ｘＣＦから始まる。図５及び図６を用いてその詳細を説明する。ここで、図５を参照して、デバイス・ジョブ制御部２１１によるコンテナ制御の流れを説明する。本実施形態では、ｄｏｃｋｅｒ（登録商標）ライブラリを、コンテナ制御ライブラリとして適用した例を説明する。なお、図５に示す構成は、本実施形態に係る画像形成装置１００に含まれるソフトウェア構成（図２）の一部を示すものであり、以下の説明に関連しない構成については省略している。

【００４１】

機器内外からデバイス・ジョブ制御部２１１に対して、制御コマンド識別子が０ｘＣ０１０である制御コマンドが送信された場合について説明する。デバイス・ジョブ制御部２１１は、当該制御コマンド識別子のターゲットオプションを表現する下位バイトが０ｘ１０であることから、コピーアプリケーションに対する制御コマンドを受信したと判別する。そして、デバイス・ジョブ制御部２１１はコピージョブ仮受信部２１２に対して、当該制御コマンドを通知する。コピージョブ仮受信部２１２は、当該制御コマンド識別子が０ｘＣ０から始まるため、コンテナ生成指示を行う制御コマンドと判断し、コンテナ制御部２２１へ対して、当該制御コマンドを通知する。

【００４２】

コンテナ制御部２２１は、当該制御コマンド識別子からコピーアプリケーションに対するコンテナ生成指示であると判断する。コンテナ制御部２２１は、当該コンテナ管理情報ＤＢ５０１を参照してＤｏｃｋｅｒｆｉｌｅ５１２の中から該当するＤｏｃｋｅｒｆｉｌｅを選択して、Ｄｏｃｋｅｒ　ｅｎｇｉｎｅ５１１に対しコンテナ生成指示を行う。その結果、Ｄｏｃｋｅｒ　ｅｎｇｉｎｅ５１１は、コピーコンテナ２２２を生成する。

【００４３】

ここで、図６を参照して、本実施形態で利用している、Ｄｏｃｋｅｒｆｉｌｅ５１２、及びコンテナ管理情報ＤＢ５０１について補足説明を行う。図６は、コンテナ制御に際して参照されるＤｏｃｋｅｒｆｉｌｅ５１２、及び、コンテナ管理情報ＤＢ５０１に格納されるコンテナ管理情報６００の例である。図６に示すＤｏｃｋｅｒｆｉｌｅ５１２は、コピーアプリケーションのためのコンテナを定義するＤｏｃｋｅｒｆｉｌｅである。「＃」から始まる行は説明を容易にするためのコメント行である。

【００４４】

４行目のＦＲＯＭは、コンテナ生成をする際のＤｏｃｋｅｒイメージが、プライベートレジストリ５１３上のａｐｐｉｍａｇｅ：１．０（５０４）であることを意味する。本実施形態においては、各コンテナのベースとなるＤｏｃｋｅｒイメージを画像形成装置１００内に保持している前提で説明を行っている。なお、本発明の実施においてはこの制約はなく、例えば、他の画像形成装置のレジストリを参照してもよいし、公開されているパブリックなレジストリを参照してもよい。

【００４５】

６行目のＲＵＮは、コンテナ生成時に実行するコマンドとして、ワークディレクトリ／ｖａｒ／ｃｏｐｙを生成することを意味する。また、８行目のＷＯＲＫＤＩＲは、コンテナ生成時に実行するコマンドとして、／ｖａｒ／ｃｏｐｙに作業ディレクトリを移動することを意味する。１０行目のＲＵＮは、コンテナ生成時に実行するコマンドとして、ｐｋｇ－ｍａｎａｇｅｒコマンドを、引数”ｉｎｓｔａｌｌ”と、”ｃｏｐｙｓｅｒｖｅｒ”とを伴って実行することを意味する。これにより、コピーコンテナ２２２上でｐｋｇ－ｍａｎａｇｅｒコマンドが実行されたことで、パッケージマネージャー５５１経由でパッケージレポジトリ５５２上から、ｃｏｐｙｓｅｒｖｅｒアプリケーションがインストールされる。なお、それぞれのコンテナは独立したストレージとして実装する以外に、例えばベースとなるイメージから更新差分のみを記録し、それらを透過的に表現するファイルシステムを用いて表現することもできる。

【００４６】

１２行目のＥＸＰＯＳＥは、コンテナ上で制御コマンドを受け取るために開放されるポート番号が２００００１であることを示す。１５行目のＣＭＤは、コンテナを動作させたときに、”／ａｐｐｓ／ｃｏｐｙｓｅｒｖｅｒ”を実行することを意味する。１８行目のＳＴＯＰＳＩＧＮＡＬは、コンテナ終了時にＣＭＤで起動しているプログラムへシグナルＳＩＧＴＥＲＭを通知することを意味する。これによって、プログラムはコンテナ終了を検知することができる。

【００４７】

ここでは、コピーアプリケーションのためのコピーコンテナ２２２を例としてＤｏｃｋｅｒｆｉｌｅの説明を行ったが、その他のアプリケーションとコンテナについても同様である。ここでは説明を簡略化するために、その他のアプリケーションについては説明を省略する。

【００４８】

６００は、コンテナ管理情報ＤＢ５０１で管理されているコンテナ管理情報である。ここでは説明上、ＤＢ、即ちデータベースという名称を用いている。当該コンテナに関する情報を管理することができればリレーショナルデーターベース以外のものを用いて当該情報を管理してもよい。例えば、単純なテキストファイルを管理書式として用いることもできる。６０１は各コンテナ管理情報の管理番号を示す。６０２はコンテナの名称を示す。６０３は本体連動起動を行うか否かを示す。本体連動起動とは、画像形成装置１００（本体）に連動してコンテナを生成して起動することを意味する。なお、本体連動起動以外のコンテナは、種々のタイミングで生成されてもよい。例えば、ユーザインタフェースを介して対応する機能の設定が開始されると、対応するコンテナが生成されて起動してもよい。また、画像形成装置１００の不図示のトレイにシートが載置されたことが検知されると、実行が予測される１以上の機能に対応するコンテナが生成されて起動されてもよい。この場合、例えば、コピー機能、スキャン機能、及びＦＡＸ機能に関連するコンテナの少な

くとも１つが生成されて起動されてもよい。さらに、画像形成装置１００が不図示の人感センサを備える場合においては、画像形成装置１００から所定の範囲において人感センサによって人が検知されると、実行が予測される１以上の機能に対応するコンテナが生成され起動されてもよい。６０４はｄｏｃｋｅｒを動かしているホスト側が制御コマンド通信用にどのポートを用いるのかを示す。６０５はコンテナ間でのＣＰＵリソース配分の優先度を示す。６０６はメモリリソース配分の最大値を示す。６０７はスワップメモリを含んだメモリリソース配分の最大値を示す。

【００４９】

６１０はコピーアプリケーションのコンテナ管理情報（コピーコンテナ２２２に関する情報）を示す。６１０に示すように、コピーコンテナ２２２の名称６０２は”ｃｏｐｙｓｅｒｖｅｒ”である。また、コピーコンテナ２２２では本体連動起動が「する」と設定されているため、画像形成装置１００の電源が投入されたタイミングで、同時にコピーコンテナ２２２は起動する。なお、ここでは、コピー機能、Ｓｅｎｄ機能、及びＰＤＬ機能が本体連動起動すると設定されているが、本発明を限定する意図はない。当該画像形成装置の使用環境や管理者の意図に応じて変更されうる設定である。コピーコンテナ２２２のｈｏｓｔ　ｐｏｒｔ６０３は、２００１６ポートが割り当てられている。コピーコンテナ２２２のｃｐｕ－ｓｈａｒｅｓ６０４には、５１２と設定されている。コピーコンテナ２２２のメモリ６０６には、２５６ＭＢが設定されており、即ち、２５６ＭＢまで利用可能であることを示す。コピーコンテナ２２２のメモリスワップ６０７には、２５６ＭＢが設定されており、即ち、スワップメモリを含むメモリが２５６ＭＢまで利用可能であることを示す。つまり、メモリ６０６との差分がないため、スワップメモリは利用しないものとしている。例えばこの値が１２８０ＭＢであれば、２５６ＭＢとの差分である１０２４ＭＢまでスワップファイルを利用できることを意味する。同様に、Ｓｅｎｄアプリケーション、ＰＤＬアプリケーション、Ｗｅｂブラウザアプリケーションについても、それぞれのコンテナ管理情報が当該コンテナ管理情報ＤＢ５０１上で管理される。このように、コンテナ管理情報６００には、コンテナごとに、ハードウェア資源（ホストポート、ＣＰＵの使用、メモリ、及びメモリスワップ）の割り当てに関する情報が含まれる

図５の説明に戻る。コンテナ制御部２２１がｄｏｃｋｅｒ　ｅｎｇｉｎｅ５１１に対して行うコンテナ制御には、コンテナ生成指示、コンテナ起動指示、及びコンテナ停止指示が含まれる。さらに、コンテナ制御には、コンテナ復帰指示、コンテナ設定変更指示、コンテナ終了指示、コンテナ破棄指示、及びコンテナ状態取得指示も含まれる。

【００５０】

コンテナ生成指示（ＣＣ＿ＣＯＮＴＡＩＮＥＲ＿ＣＭＤ＿ＣＲＥＡＴＥ、０ｘＣ０００）は、前述したＤｏｃｋｅｒｆｉｌｅ５１２と、コンテナ管理情報６００とに基づいて、コンテナを生成する指示である。コンテナ起動指示（ＣＣ＿ＣＯＮＴＡＩＮＥＲ＿ＣＭＤ＿ＳＴＡＲＴ、０ｘＣ１００）は、コンテナ上でアプリケーションを起動する指示である。コンテナ停止指示（ＣＣ＿ＣＯＮＴＡＩＮＥＲ＿ＣＭＤ＿ＰＡＵＳＥ、０ｘＣ２００））は、コンテナを起動状態から停止状態へ一時停止させる指示である。上記コンテナ停止指示は、例えば、現在ログインしているユーザがログアウトした場合や、対応する機能のジョブ終了後に同機能のジョブが投入されずに一定時間が経過した場合に行われる。さらに、コンテナ停止指示は、他のコンテナを起動する際にハードウェア資源が所定値未満となり、そのタイミングで使用頻度が低い場合や、致命的なエラーが発生した場合などにも行われる。コンテナ復帰指示（ＣＣ＿ＣＯＮＴＡＩＮＥＲ＿ＣＭＤ＿ＵＮＰＡＵＳＥ、０ｘＣ３００）は、一時停止したコンテナを再度活動状態にする指示である。また、装置が省電力モードやスリープモードへ移行した際に、少なくとも１つのコンテナを一時停止するようにしてもよい。コンテナ復帰指示は、基本的には停止状態のコンテナに対して機能実行の要求を受け付けたタイミングや装置の省電力モード等からの復帰時となる。コンテナ設定変更（ＣＣ＿ＣＯＮＴＡＩＮＥＲ＿ＣＭＤ＿ＵＰＤＡＴＥ、０ｘＣ７００）は、コンテナに割り振るＣＰＵリソースやメモリリソースなどのリソース設定を変更する指示である。コンテナ終了指示（ＣＣ＿ＣＯＮＴＡＩＮＥＲ＿ＣＭＤ＿ＳＴＯＰ、０ｘＣＤ００

）は、コンテナ上のアプリケーションを終了させる指示である。コンテナ破棄指示（ＣＣ＿ＣＯＮＴＡＩＮＥＲ＿ＣＭＤ＿ＲＥＭＯＶＥ、０ｘＣＥ００）は、コンテナを破棄する指示である。コンテナ破棄指示は、例えば常駐させないコンテナにおいてジョブを実行した後に、当該コンテナを破棄する場合に行われる。コンテナ状態取得指示（ＣＣ＿ＣＯＮＴＡＩＮＥＲ＿ＣＭＤ＿ＧＥＴ＿ＳＴＡＴＵＳ，０ｘＣＦ００）は、コンテナの状態を取得する指示である。

【００５１】

（アプリケーション制御コマンド）

図４に示すように、アプリケーションに対する制御コマンド識別子は、０ｘＡ０～０ｘＡＦから始まる。ここで、図７を参照して、本実施形態に係るデバイス・ジョブ制御部２１１によるアプリケーションに対するアプリケーション制御コマンドについて説明する。以下では、機器内外からデバイス・ジョブ制御部２１１に対し、制御コマンド識別子が０ｘＡ１１０である制御コマンドが送信された場合を例に説明する。

【００５２】

デバイス・ジョブ制御部２１１は、当該制御コマンド識別子のターゲットオプションを表現する下位バイトが０ｘ１０であることから、コピーアプリケーションに対する制御コマンドを受信したと判別する。そして、デバイス・ジョブ制御部２１１はコピージョブ仮受信部２１２に対して、当該制御コマンドを通知する。コピージョブ仮受信部２１２は、当該制御コマンド識別子が０ｘＡ１から始まることを検知し、アプリケーション実行指示を行う制御コマンドであることを判別し、コピージョブ本受信部２３１に対して当該制御コマンドを通知する。

【００５３】

なお、この際、コピージョブ仮受信部２１２は、コンテナ管理情報ＤＢ５０１のコピーコンテナ２２２に関する管理情報６１０を参照することで、コピーアプリケーションに割り当てられた制御コマンド通信用ポートが２００１６であることを認識する。そして、コピージョブ仮受信部２１２は、２００１６ポートに対して、制御コマンドを送信する。２００１６ポートに対する通信は、ｄｏｃｋｅｒ　ｅｎｇｉｎｅ５１１が受け付け、さらにコピーコンテナ２２２上に対して２０００１ポートへ送信する。コピージョブ本受信部２３１は２０００１ポートに送信された制御コマンドを読み取り、コピージョブ実行部２４１へ通知することで、コピージョブが実行される。それぞれのコンテナ２２２～２２５は、コンテナ管理情報ＤＢ５０１のコンテナ管理情報６００に記載されたリソース配分に従って動作をしている。

【００５４】

なお、本発明において、画像形成装置１００は、すべてのコンテナを同時に常時動作させておく必要はない。例えば、本実施形態では、ユーザが明確に利用する意思を示したタイミング、例えばＷｅｂブラウザＵＩアプリ部２０５を操作したタイミングで、Ｗｅｂブラウザコンテナ２２５を起動する。また、本実施形態では本体連動起動として常時動作させているアプリケーションについてもユーザ設定に従って常時動作させないようにしてもよい。その場合は、上記Ｗｅｂブラウザと同様にユーザが明確に利用する意思を示したタイミングで対応するコンテナが起動される。

【００５５】

図５の説明に戻る。各仮受信部２１２～２１５が各本受信部２３１～２３４に対して行うアプリケーション指示には、実行指示、一時停止指示、キャッシュメモリフラッシュ要求指示、及びキャンセル指示が含まれる。実行指示（ＣＣ＿ＡＰＰＬＩＣＡＴＩＯＮ＿ＣＭＤ＿ＥＸＥＣＵ‘ＴＥ、０ｘＡ１００）は、当該アプリケーションにおいてジョブを実行する指示である。一時停止指示（ＣＣ＿ＡＰＰＬＩＣＡＴＩＯＮ＿ＣＭＤ＿ＳＴＯＰ，０ｘＡ５００）は、当該アプリケーションで実行中のジョブを一時停止する指示である。キャッシュメモリフラッシュ要求指示（ＣＣ＿ＡＰＰＬＩＣＡＴＩＯＮ＿ＣＭＤ＿Ｃ＿ＦＬＵＳＨ，０ｘＡ７００）は、当該アプリケーション実行中に保持したメモリ上のキャッシュを破棄する指示である。これにより、アプリケーションが利用するメモリ使用量を削減

することができる。キャンセル指示（ＣＣ＿ＡＰＰＬＩＣＡＴＩＯＮ＿ＣＭＤ＿ＣＡＮＣＥＬ、０ｘＡＦ００）は、当該アプリケーションで実行中のジョブをキャンセルする指示である。

【００５６】

（デバイス制御コマンド）

図４に示すように、デバイスに対する制御コマンド識別子は、０ｘＤ０～０ｘＤＦから始まる。デバイス制御コマンドには、例えば、機器内外からデバイスの電力制御などの指示を受け付けるための制御コマンドを含むことができる。デバイス制御コマンド識別子には、スリープ復帰指示、スリープ突入指示、パネル点灯指示、パネル消灯指示、高速復帰電源断指示、及び電源切断指示がある。

【００５７】

スリープ突入指示（ＣＣ＿ＤＥＶＩＣＥ＿ＣＭＤ＿ＷＡＫＥＵＰ，０ｘＤ０００）は、消費電力を抑える省電力モードへデバイスの状態を移行する指示である。スリープ復帰指示（ＣＣ＿ＤＥＶＩＣＥ＿ＣＭＤ＿ＳＬＥＥＰ、０ｘＤ１００）は、省電力モードから通常モードへデバイスの状態を移行する指示である。パネル消灯指示（ＣＣ＿ＤＥＶＩＣＥ＿ＣＭＤ＿ＰＡＮＥＬＯＦＦ、０ｘＤ２００）は、操作部ユニット１３１に備えられた操作部パネルの画面表示を停止する指示である。パネル点灯指示（ＣＣ＿ＤＥＶＩＣＥ＿ＣＭＤ＿ＰＡＮＥＬＯＮ，０ｘＤ３００）は、操作部ユニット１３１に備えられた操作部パネルの画面表示を実行する指示である。高速復帰電源断指示（ＣＣ＿ＤＥＶＩＣＥ＿ＣＭＤ＿ＱＵＩＣＫＯＦＦ、０ｘＤ４００）は、通常モードから高速復帰電源断モードへデバイスの状態を移行する指示である。画像形成装置１００が高速復帰電源断モードに突入した場合、電源スイッチ１３０の操作によってのみ、画像形成装置１００は通常モードへ復帰できる。電源切断指示（ＣＣ＿ＤＥＶＩＣＥ＿ＣＭＤ＿ＰＯＷＥＲＯＦＦ，０ｘＤＦ００）は、デバイスの電源を切断する指示である。

【００５８】

なお、本実施形態に係る画像形成装置１００では、プリンタユニット１３２、スキャナユニット１３３、及びＦＡＸユニット１３４等の電力制御を細かに制御することで、画像形成装置１００全体の消費電力を抑える。例えば、プリンタユニット１３２に対して、待機中の消費電力量の異なる２種類のモードを備え、本体ファーム２００からの指示に従ってモードを使い分けるなどの工夫を行うこともできる。本実施形態では、デバイスのこれら電力制御は一般的な画像形成装置における電力制御と同様の処理となることから、詳細な説明については割愛する。

【００５９】

（汎用制御レスポンス識別子）

図４に示すように、汎用制御レスポンス識別子には、成功レスポンス識別子（ＣＣ＿ＲＥＳＰＯＮＳＥ＿ＯＫ、０ｘ００００）と、失敗レスポンス識別子が含まれる（ＣＣ＿ＲＥＳＰＯＮＳＥ＿ＮＧ，０ｘ０００１）。これらは、種々の制御コマンドに対する処理が成功したか失敗したかを通知するために用いられる。

【００６０】

（コンテナ状態制御レスポンス識別子）

図４に示すように、コンテナ状態制御レスポンス識別子は、コンテナ未存在状態制御レスポンス、コンテナ未起動状態制御レスポンス、コンテナ起動状態レスポンス、コンテナ停止状態レスポンスが含まれる。さらに、コンテナ状態制御レスポンス識別子には、コンテナ再起動中レスポンス、コンテナ終了中レスポンス、コンテナ削除中レスポンス、及び、コンテナ停止失敗中レスポンスが含まれる。

【００６１】

コンテナ未存在状態制御レスポンス（ＣＣ＿ＲＥＳＰＯＮＳＥ＿ＮＯＮＥＸＩＳＴＳ，０ｘ８０００）とは、対象となったアプリケーションを動かすためのコンテナが存在していない状態であることを示す。コンテナ未起動状態制御レスポンス（ＣＣ＿ＲＥＳＰＯＮＳＥ＿ＣＲＥＡＴＥＤ、０ｘ０００００）は、対象となったコンテナは存在しているが、起

動していない状態であることを示す。コンテナ起動状態レスポンス（ＣＣ＿ＲＥＳＰＯＮＳＥ＿ＲＵＮＮＩＮＧ，０ｘ０００１）は、対象となったコンテナが存在し、起動している状態であることを示す。コンテナ停止状態レスポンス（ＣＣ＿ＲＥＳＰＯＮＳＥ＿ＰＡＵＳＥＤ，０ｘ０００２）は、対象となったコンテナが存在し、起動しているが、停止指示を受け付けたことを示す。コンテナ再起動中レスポンス（ＣＣ＿ＲＥＳＰＯＮＳＥ＿ＲＥＳＴＡＲＴＩＮＧ，０ｘ０００３）は、コンテナに対する再起動要求を実行中であることを示す。コンテナ終了中レスポンス（ＣＣ＿ＲＥＳＰＯＮＳＥ＿ＥＸＩＴＥＤ，０ｘ０００４）は、コンテナが終了状態にあることを示す。コンテナ削除中レスポンス（ＣＣ＿ＲＥＳＰＯＮＳＥ＿ＲＥＭＯＶＩＮＧ，０ｘ０００５）は、コンテナに対する削除要求を実行中であることを示す。コンテナ停止失敗中レスポンス（ＣＣ＿ＲＥＳＰＯＮＳＥ＿ＤＥＡＤ、０ｘ０００６）は、対象となってコンテナが存在し、起動しているが、停止指示の要求に対して停止処理が失敗した状態であることを示す。

【００６２】

＜デバイス・ジョブ制御部の処理手順＞

次に、図８を参照して、本実施形態に係るデバイス・ジョブ制御部２１１の処理手順について説明する。以下で説明する処理は、例えばＣＰＵ１１１がＲＯＭ１１２や内蔵ストレージ１１４に格納された制御プログラムをＲＡＭ１１３に読み出して実行することにより実現される。

【００６３】

Ｓ８０１はループ処理を示し、デバイス・ジョブ制御部２１１は、全ての仮受信部２１２～２１５、及び、受信部２１６に対して、以下のＳ８０２乃至Ｓ８０５の処理を繰り返し実行する。Ｓ８０２で、デバイス・ジョブ制御部２１１は、それぞれの受信部に対して、受信部起動コマンドを送信する。続いて、Ｓ８０３で、デバイス・ジョブ制御部２１１は、それぞれの受信部からレスポンスを受信する。Ｓ８０４で、デバイス・ジョブ制御部２１１はレスポンスが失敗であるかを判断する。失敗である場合はＳ８０５に進み、デバイス・ジョブ制御部２１１は、致命エラーが発生したことを記録し、繰り返しの処理を継続する。一方、失敗でない場合はそのまま繰り返しの処理を継続する。以上のＳ８０２乃至Ｓ８０５の処理を繰り返すことで、デバイス・ジョブ制御部２１１は、全仮受信部２１２～２１５、及び、受信部２１６に対して起動指示を行う。全ての処理が終了するとＳ８０６に進む。

【００６４】

Ｓ８０６で、デバイス・ジョブ制御部２１１は、致命エラーが発生したか否かを判断し、致命エラーが発生している場合にはＳ８０８に進み、そうでない場合はＳ８０７に進む。Ｓ８０８で、デバイス・ジョブ制御部２１１は、操作部制御部２０１に対して、致命エラー発生に伴うエラー表示要求を通知し、処理を終了させる。一方、Ｓ８０７で、デバイス・ジョブ制御部２１１は、致命エラーが発生しなかった場合には、操作部制御部２０１に対して、起動指示を行う。操作部制御部２０１は、当該起動指示を受けて、ＵＩアプリ部２０２～２０６の起動を行い、Ｓ８３１に進む。なお、ＵＩアプリ部２０２～２０６は、プログラムの更新を容易にするためにも、操作部制御部２０１から呼び出される外部プログラムの形態で実現することが望ましい。

【００６５】

Ｓ８３１で、デバイス・ジョブ制御部２１１は、機器内外から送信されたコマンドを受信する。そして、Ｓ８３２で、デバイス・ジョブ制御部２１１は、受信したコマンドの種別に応じて、処理を分岐する。受信したコマンド種別がアプリ制御コマンドであれば、Ｓ８３３に進み、デバイス・ジョブ制御部２１１は当該制御コマンドからターゲットアプリケーションを特定する。続いて、Ｓ８３４で、デバイス・ジョブ制御部２１１は、ターゲットアプリケーションの仮受信部又は受信部に対して、当該制御コマンドを送信する。仮受信部又は受信部は後述する通り、当該コマンドに対する処理が行われ、制御レスポンスを送信する。Ｓ８３５で、デバイス・ジョブ制御部２１１は、ターゲットアプリケーションからレスポンスを受信する。そして、Ｓ８３６で、デバイス・ジョブ制御部２１１は、

Ｓ８３１で受信した制御コマンドのユーザ識別子に記録されている送信元に対して、当該受信したレスポンスを送信する。

【００６６】

一方、Ｓ８３２で受信したコマンド種別がデバイス制御コマンドと判断するとＳ８５１に進み、デバイス・ジョブ制御部２１１はコマンド種別に応じて処理を分岐する。コマンド種別がスリープ復帰コマンドであればＳ８５２に進み、デバイス・ジョブ制御部２１１は、スリープ復帰処理を実行する。同様に、Ｓ８５３乃至Ｓ８５６において、デバイス・ジョブ制御部２１１は、スリープ突入コマンド、パネル点灯コマンド、パネル消灯コマンド、又は高速復帰電源断コマンドに対して、それぞれの処理を実行する。その後、Ｓ８５７で、デバイス・ジョブ制御部２１１はこれらのコマンド実行の結果をレスポンスとして取得する。続いて、Ｓ８５８で、デバイス・ジョブ制御部２１１は、Ｓ８３１で受信した制御コマンドのユーザ識別子に記録されている送信元に対して、当該デバイス制御を行ったレスポンスを送信する。そして、また新たなコマンドを受信すべく、処理をＳ８３１に戻す。

【００６７】

また、Ｓ８５１でコマンド種別が電源断コマンドであると判断するとＳ８９１に進み、デバイス・ジョブ制御部２１１は、電源切断処理を実行し、処理を終了する。その結果、画像形成装置１００は完全な電源断状態となる。

【００６８】

以上の制御により、デバイス・ジョブ制御部２１１は、受信した制御コマンドに対して制御コマンドに含まれるターゲットに応じて、アプリケーションの仮受信部・受信部に対して、制御コマンドを転送したり、或いは、デバイスを制御したりできる。

【００６９】

＜仮受信部の処理手順＞

次に、図９を参照して、本実施形態に係る仮受信部２１２～２１５の処理手順について説明する。ここでは、コピージョブ仮受信部２１２を代表として説明をする。そして、他の仮受信部２１３～２１５については、コピージョブ仮受信部２１２と同様の振る舞いをするため、説明は割愛する。以下で説明する処理は、例えばＣＰＵ１１１がＲＯＭ１１２や内蔵ストレージ１１４に格納された制御プログラムをＲＡＭ１１３に読み出して実行することにより実現される。

【００７０】

Ｓ９０１で、コピージョブ仮受信部２１２は、デバイス・ジョブ制御部２１１から仮受信部起動コマンドを受信する。続いて、Ｓ９０２で、コピージョブ仮受信部２１２は、仮受信部の起動処理を実行する。例えば、仮受信部の起動処理では、事前にメモリを確保したり、関連するハードウェアの設定をしたりする。その後、Ｓ９０３で、コピージョブ仮受信部２１２は起動処理が成功したかを判断する。起動処理が成功した場合にはＳ９０４に進み、コピージョブ仮受信部２１２はデバイス・ジョブ制御部２１１へ成功レスポンスを通知し、以後アプリ制御コマンドに対する処理を繰り返すべく、処理をＳ９１１に進める。一方、起動処理が失敗した場合にはＳ９０５に進み、コピージョブ仮受信部２１２はデバイス・ジョブ制御部２１１へ失敗レスポンスを通知し、処理を終了する。

【００７１】

Ｓ９１１で、コピージョブ仮受信部２１２は、デバイス・ジョブ制御部２１１からアプリ制御コマンドを受信する。続いて、Ｓ９１２で、Ｓ９１１で受信したアプリ制御コマンドを実行するために、コピージョブ仮受信部２１２は、コンテナ制御部２２１に対して、コンテナ状態取得コマンドを送信する。さらに、Ｓ９１３で、コピージョブ仮受信部２１２は、コンテナ制御部２２１から、コンテナ状態制御レスポンスを取得する。

【００７２】

次に、Ｓ９１４で、コピージョブ仮受信部２１２は、受信したコンテナ状態に応じて処理を切り替える。コンテナ状態がコンテナ未起動状態（ＣＣ＿ＲＥＳＰＯＮＳＥ＿ＣＲＥＡＴＥＤ）だった場合はＳ９２１に進み、コピージョブ仮受信部２１２は、コンテナ制御

部２２１にコンテナ起動コマンド（ＣＣ＿ＣＯＮＴＡＩＮＥＲ＿ＣＭＤ＿ＳＴＡＲＴ）を送信する。続いて、Ｓ９２２で、コピージョブ仮受信部２１２は、コンテナ制御部２２１からコンテナ起動コマンドのレスポンスを受信する。以上により、当該アプリケーションを動かすためのコンテナはコンテナ起動状態（ＣＣ＿ＲＥＳＰＯＮＳＥ＿ＲＵＮＮＩＮＧ）へ移行する。そして、Ｓ９２９に進み、コピージョブ仮受信部２１２は一定時間待機した後、再度コンテナ状態時取得処理を繰り返すべく処理をＳ９１２に戻す。

【００７３】

Ｓ９１４でコンテナ状態がコンテナ停止状態（ＣＣ＿ＲＥＳＰＯＮＳＥ＿ＰＡＵＳＥＤ）だった場合はＳ９２３に進む。Ｓ９２３で、コピージョブ仮受信部２１２は、コンテナ制御部２２１に対して、コンテナ復帰コマンド（ＣＣ＿ＣＯＮＴＡＩＮＥＲ＿ＣＭＤ＿ＵＮＰＡＵＳＥ）を送信する。続いて、Ｓ９２４で、コピージョブ仮受信部２１２は、コンテナ制御部２２１からコンテナ復帰コマンドのレスポンスを受信する。以上により、当該アプリケーションを動かすためのコンテナはコンテナ起動状態（ＣＣ＿ＲＥＳＰＯＮＳＥ＿ＲＵＮＮＩＮＧ）へ移行する。そして、Ｓ９２９に進み、コピージョブ仮受信部２１２は一定時間待機した後、再度コンテナ状態時取得処理を繰り返すべく処理をＳ９１２に戻す。

【００７４】

Ｓ９１４でコンテナ状態がコンテナ稼働状態（ＣＣ＿ＲＥＳＰＯＮＳＥ＿ＲＵＮＮＩＮＧ）だった場合はＳ９４１に進む。Ｓ９４１で、コピージョブ仮受信部２１２は、コピージョブ本受信部２３１に対して、Ｓ９１１で受信したアプリ制御コマンドを送信する。続いて、Ｓ９４２で、コピージョブ仮受信部２１２は、コピージョブ本受信部２３１からアプリ制御コマンドに対するレスポンスを受信する。さらに、Ｓ９４３で、コピージョブ仮受信部２１２は、Ｓ９１１で受信したアプリ制御コマンドの送信元へＳ９４２で受信したレスポンスを送信し、処理をＳ９１１に戻す。

【００７５】

Ｓ９１４でコンテナ状態が、上記以外の状態だった場合はＳ９３１へ進み、コピージョブ仮受信部２１２は、操作部制御部２０１へ致命エラーを通知する。そして、Ｓ９３２で、コピージョブ仮受信部２１２は、Ｓ９１１で受信したアプリ制御コマンドの送信元へ失敗レスポンスを送信し、処理を終了する。

【００７６】

以上の制御により、コピージョブ仮受信部２１２は、コピーコンテナ２２２の状態にかかわらず、コピージョブに関するアプリ制御コマンドを常時受け付けることができる。さらに、コピージョブ仮受信部２１２は、コピーコンテナ２２２に対するコンテナ起動コマンドやコンテナ復帰コマンドをコンテナ制御部２２１へ送信することで、受信したアプリ制御コマンドを実行するためのコンテナ環境を準備することができる。

【００７７】

＜コンテナ制御部の処理手順＞

次に、図１０を参照して、本実施形態に係るコンテナ制御部２２１の処理手順について説明する。以下で説明する処理は、例えばＣＰＵ１１１がＲＯＭ１１２や内蔵ストレージ１１４に格納された制御プログラムをＲＡＭ１１３に読み出して実行することにより実現される。

【００７８】

Ｓ１００１はループ処理を示し、コンテナ制御部２２１は、管理をしている全コンテナ２２２～２２５を対象に、Ｓ１００２乃至Ｓ１００７、Ｓ１０１１乃至Ｓ１０１４、及びＳ１０２１の処理を繰り返す。Ｓ１００２で、コンテナ制御部２２１は、ｄｏｃｋｅｒ　ｅｎｇｉｎｅ５１１に対して、コンテナ状態取得コマンドを実行する。続いてＳ１００３で、コンテナ制御部２２１は、ｄｏｃｋｅｒ　ｅｎｇｉｎｅ５１１から状態取得コマンドのレスポンスを取得する。そして、Ｓ１００４で、コンテナ制御部２２１は、取得したレスポンスによって処理を切り替える。

【００７９】

Ｓ１００４で取得したレスポンスが、コンテナが未存在である場合はＳ１００５に進み、コンテナ制御部２２１は、ｄｏｃｋｅｒ　ｅｎｇｉｎｅ５１１に対して、コンテナ生成コマンドを実行する。続いて、Ｓ１００６で、コンテナ制御部２２１は、ｄｏｃｋｅｒ　ｅｎｇｉｎｅ５１１からコンテナ生成コマンドのレスポンスを取得する。そして、Ｓ１００７で、コンテナ制御部２２１は、取得したレスポンスによって処理を切り替える。取得したレスポンスがコンテナ生成に失敗した場合はＳ１０２１に進み、コンテナ制御部２２１は致命エラー発生を記録して、繰り返しの処理を継続する。

【００８０】

一方、Ｓ１００４でコンテナの状態取得コマンドのレスポンスがコンテナ未存在ではない、或いは、Ｓ１００７でコンテナ生成コマンドのレスポンスが成功だった場合はＳ１０１１に進む。Ｓ１０１１で、コンテナ制御部２２１は、当該コンテナが本体連動起動か否かによって処理を切り替える。本体連動起動となっている場合はＳ１０１２に進み、コンテナ制御部２２１は、ｄｏｃｋｅｒ　ｅｎｇｉｎｅ５１１に対してコンテナ起動コマンドを実行する。続いて、Ｓ１０１３で、コンテナ制御部２２１は、ｄｏｃｋｅｒ　ｅｎｇｉｎｅ５１１からコンテナ起動コマンドのレスポンスを取得する。そして、Ｓ１０１４で、コンテナ制御部２２１は、取得したレスポンスによって、処理を切り替える。ここで、コンテナ起動コマンドが失敗した場合はＳ１０２１に進み、コンテナ制御部２２１は致命エラー発生を記録し、繰り返しの処理を継続する。一方、Ｓ１０１１で本体連動起動になっていない場合、或いは、Ｓ１０１４でコンテナ起動コマンドが失敗ではない場合は、そのまま繰り返しの処理を継続する。

【００８１】

以上の処理を繰り返すことで、コンテナ制御部２２１が管理している全コンテナ２２２～２２５が生成される。そして、コンテナ管理情報ＤＢ５０１のコンテナ管理情報６００上で本体連動起動が指定されているコンテナは、コンテナ起動状態（即ち、ＣＣ＿ＲＥＳＰＯＮＳＥ＿ＲＵＮＮＩＮＧ、０ｘ０００１と同等）となる。そして、本体連動起動が指定されていないコンテナは、コンテナ未起動状態（即ち、ＣＣ＿ＲＥＳＰＯＮＳＥ＿ＣＲＥＡＴＥＤ、０ｘ０００００と同等）となる。繰り返し処理が終了すると、Ｓ１０４１に進む。

【００８２】

Ｓ１０４１で、コンテナ制御部２２１は、管理をしている全コンテナ２２２～２２５を対象としたコンテナ起動処理において、Ｓ１０２１での致命エラーが記録されたか否かによって処理を切り替える。致命エラーが記録されている場合はＳ１０４２へ進み、コンテナ制御部２２１は、操作部制御部２０１に対して、致命エラー表示要求を通知し、処理を終了する。

【００８３】

一方、致命エラーが記録されていない場合はＳ１０５０に進み、コンテナ制御部２２１は、コンテナ制御コマンドの受信処理を開始し、機内外から、コンテナ制御コマンドを受信する。続いて、Ｓ１０５１で、コンテナ制御部２２１は、受信したコンテナ制御コマンドに応じて、処理を切り替える。

【００８４】

Ｓ１０５１で受信したコンテナ制御コマンドがコンテナ生成コマンドである場合にはＳ１０５２に進み、コンテナ制御部２２１は、ｄｏｃｋｅｒ　ｅｎｇｉｎｅ５１１に対して、コンテナ生成コマンドを実行する。受信したコンテナ制御コマンドがコンテナ起動コマンドである場合にはＳ１０５３に進み、コンテナ制御部２２１は、ｄｏｃｋｅｒ　ｅｎｇｉｎｅ５１１に対して、コンテナ起動コマンドを実行する。受信したコンテナ制御コマンドがコンテナ停止コマンドである場合にはＳ１０５４に進み、コンテナ制御部２２１は、ｄｏｃｋｅｒ　ｅｎｇｉｎｅ５１１に対して、コンテナ停止コマンドを実行する。受信したコンテナ制御コマンドがコンテナ復帰コマンドである場合にはＳ１０５５に進み、コンテナ制御部２２１は、ｄｏｃｋｅｒ　ｅｎｇｉｎｅ５１１に対して、コンテナ復帰コマンドを実行する。受信したコンテナ制御コマンドがコンテナ設定変更コマンドである場合に

はＳ１０５６に進み、コンテナ制御部２２１は、ｄｏｃｋｅｒ　ｅｎｇｉｎｅ５１１に対して、コンテナ設定変更コマンドを実行する。受信したコンテナ制御コマンドがコンテナ終了コマンドである場合にはＳ１０５７に進み、コンテナ制御部２２１は、ｄｏｃｋｅｒ　ｅｎｇｉｎｅ５１１に対して、コンテナ復帰コマンドを実行する。受信したコンテナ制御コマンドがコンテナ破棄コマンドである場合にはＳ１０５８に進み、コンテナ制御部２２１は、ｄｏｃｋｅｒ　ｅｎｇｉｎｅ５１１に対して、コンテナ破棄コマンドを実行する。各コマンドを実行するとそれぞれＳ１０６１に進む。

【００８５】

Ｓ１０６１で、コンテナ制御部２２１は、ｄｏｃｋｅｒ　ｅｎｇｉｎｅ５１１で実行した各種コマンドに対するレスポンスを受信する。続いて、Ｓ１０６２で、コンテナ制御部２２１は、受信したレスポンスによって処理を切り替える。コマンドのレスポンスが成功だった場合にはＳ１０６３に進み、コンテナ制御部２２１は、コンテナ制御コマンドに対するレスポンスを成功レスポンスとし、Ｓ１０９１に進む。一方、コマンドのレスポンスが失敗だった場合にはＳ１０６４に進み、コンテナ制御部２２１は、コンテナ制御コマンドに対するレスポンスを失敗レスポンスとし、Ｓ１０９１に進む。

【００８６】

また、Ｓ１０５１で受信したコンテナ制御コマンドがコンテナ状態取得コマンドである場合にはＳ１０７１に進み、コンテナ制御部２２１は、ｄｏｃｋｅｒ　ｅｎｇｉｎｅ５１１に対して、コンテナ存在確認コマンドを実行する。続いて、Ｓ１０７２で、コンテナ制御部２２１は、ｄｏｃｋｅｒ　ｅｎｇｉｎｅ５１１から上記Ｓ１０７１のコンテナ存在確認コマンドのレスポンスを取得する。そして、Ｓ１０７３で、コンテナ制御部２２１は、コンテナが存在するかによって処理を切り替える。当該コンテナが存在する場合は、Ｓ１０７５に進み、コンテナ制御部２２１は、ｄｏｃｋｅｒ　ｅｎｇｉｎｅ５１１に対して、コンテナ状態取得コマンドを実行する。続いて、Ｓ１０７６で、コンテナ制御部２２１は、ｄｏｃｋｅｒ　ｅｎｇｉｎｅ５１１からコンテナ状態取得コマンドのレスポンスを取得する。そして、Ｓ１０７７で、コンテナ制御部２２１は、コンテナ制御コマンドに対するレスポンスを、コンテナ状態取得コマンドのレスポンスに基づいて設定し、処理をＳ１０９１に進める。例えば、ｄｏｃｋｅｒ　ｅｎｇｉｎｅ５１１上で当該コンテナが起動状態（ＲＵＮＮＩＮＧ）であれば、コマンド制御コマンドに対するレスポンスは、コンテナ起動状態（ＣＣ＿ＲＥＳＰＯＮＳＥ＿ＲＵＮＮＩＮＧ，０ｘ０００１）となる。一方、Ｓ１０７３でコンテナが存在しない場合にはＳ１０７４に進み、コンテナ制御部２２１は、コンテナ制御レスポンスにコンテナ無しを設定し、処理をＳ１０９１に進める。

【００８７】

Ｓ１０９１で、コンテナ制御部２２１は、Ｓ１０５０で受信した制御コマンドの送信元へ、以上の処理で決定したコンテナ制御コマンドのレスポンスを送信し、処理をＳ１０５０に戻す。

【００８８】

以上の制御により、コンテナ制御部２２１は、機内で流通するコンテナ制御コマンドに基づき、コンテナを制御するためのｄｏｃｋｅｒ　ｅｎｇｉｎｅ５１１に対して適切なコマンドを発行し、各コンテナを制御することができる。当該構成を用いることにより、コンテナ制御部２２１に対してコンテナ制御コマンドを送信し、コンテナ制御コマンドのレスポンスを受信する各アプリケーションは、コンテナを実現するライブラリの種別を意識する必要がなくなる。例えば、本実施例では、一般的なコンテナ管理ライブラリｄｏｃｋｅｒを用いて説明をしているが、それ以外のコンテナ管理ライブラリを用いてシステムを構築することもできる。各コンテナ管理ライブラリごとに、アプリケーションが発行するべきコンテナ制御コマンドを使い分けるのは非常に手間である。そのため、コンテナ制御部２２１のようにコンテナ制御全般を受け持つブリッジモジュールを導入することで、コンテナライブラリ毎の振る舞いの違いを吸収することもできる。

【００８９】

＜まとめ＞

ある機能を実現するアプリケーションの一般的なデザインパターンとして、モデル／ビュー／コントローラからなるＭＶＣが広く知られている。ビューはモデルが示す情報に基づいて表示画面を作成するモジュールである。コントローラは、ユーザからの操作を受け付けて、モデルへ通知するモジュールである。モデルはユーザによって操作された内容に基づき、ビューを更新するモジュールである。ＭＶＣモデルはユーザインタフェースを伴うソフトウェアにおいて、マルチプラットフォームでの再利用性を高めたりするために採用されることも少なくないデザインパターンである。

【００９０】

しかし、従来のＭＶＣの考え方だけでは、コンテナ上で動作するアプリケーションの規模は非常に大きくなり、リソースの効率的な活用をするには扱いにくい。具体的には、ＭＶＣすべてのモジュールをコンテナ上で動かすと、操作要求に応じてコンテナを稼働するのに時間を要し、また、必要となるリソースも大きくなるという課題がある。これは、従来の考え方は、あるアプリケーションを起動状態とすることは、ユーザに対して全機能を提供可能にすることと同義としていたためである。

【００９１】

本発明では、当該ＭＶＣの考え方を発展させ、ビューとコントローラに相当する部分はコンテナの外に設け、モデルに相当する部分をコンテナ内で管理する、組み込みアプリケーション用の管理手法を提案している。これにより、あるアプリケーションにおいて、ユーザからの操作要求を受け付ける部分と、ユーザからの操作要求を実行する部分とに分割できる。そして、部分ごとに起動するか否かを切り替えることで、リソースの効果的な活用を実現することができる。例えば、ビューとコントローラに相当する部分は常時起動させることで、ユーザからの操作を本体ファームにおいてリアルタイムに応答できる。また、コンテナ上で必要に応じてモデルに相当するアプリケーション実行部分を起動することで、複数のアプリケーションを搭載していても、システム全体で要求されるリソースを削減することができる。

【００９２】

上述の構成を実現すべく、本実施形態に係る１以上の機能を提供する画像形成装置は、ユーザインタフェースを介して、それぞれが対応する機能の要求を受け付ける１以上の第１モジュールと、１以上の第１モジュールのそれぞれに対応する機能を実行する１以上の第２モジュールとを備える。また、本画像形成装置は、１以上の第１モジュールが受け付けた要求を、対応する第２モジュールへ通知する第１制御部と、当該通知に従って、対応する第２モジュールを制御する第２制御部とを備える。さらに、本画像形成装置によれば、１以上の第１モジュールは、画像形成装置の起動時から常時起動され、１以上の第２モジュールは、オペレーティングシステムからは独立した実行環境のコンテナとして生成され、第２制御部からの指示により起動状態が制御される。これにより、本実施形態によれば、組み込みデバイスにコンテナ技術を好適に適用して、ハードウェア資源を有効に活用するとともに、ユーザの操作に対する応答性を担保することができる。

【００９３】

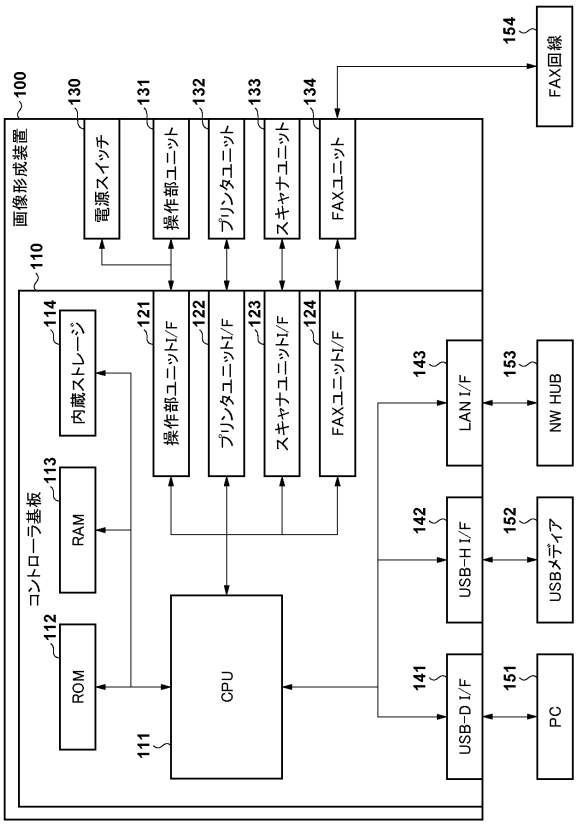
＜変形例＞

本発明は上記実施形態に限らず様々な変形が可能である。本発明によれば、仮受信部２１２～２１５が制御コマンドを本受信部へ通知する以外の処理や機能を実現してもよい。例えば、仮受信部２１２～２１５は、それぞれのアプリケーションが利用するプリンタユニット制御部２７１やスキャナユニット制御部２７２に対して通信を行い、ジョブ受け付け可否（実行可能であるか否か）を判断する機能を有していてもよい。この場合、スキャナユニット制御部２７２がＳｅｎｄジョブ実行部２４２によって利用されている状態であっても、ＰＤＬジョブ実行部２４３はプリンタユニット制御部２７１を利用して印字処理を実行することができる。しかし、コピージョブ実行部２４１は、スキャナユニット制御部２７２が利用中である場合には、ジョブ実行の予約はできるが、即時実行はできない。これらのジョブ受け付け可否を判断する機能は、それぞれの実行部２４１～２４４に備える構成のほかに、仮受信部２１２～２１５が備える構成をとってもよい。

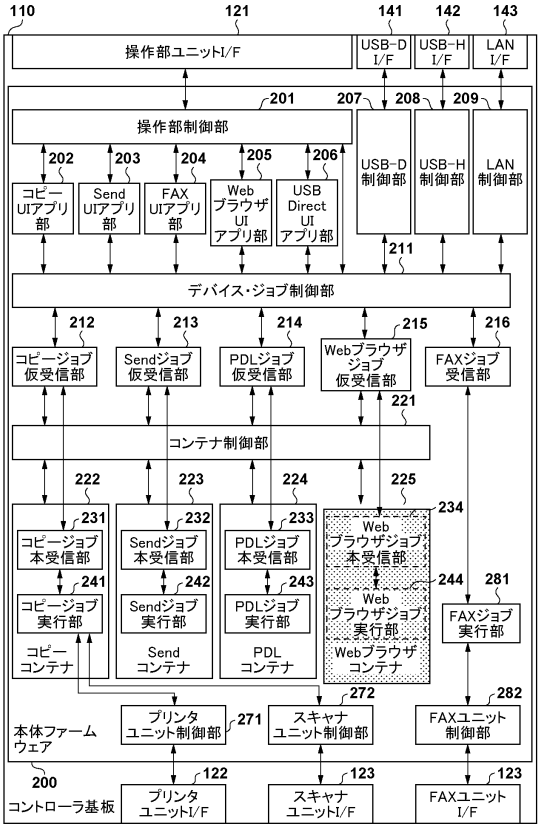
【００９４】
　また、それぞれのジョブ実行部２４１～２４４が備える機能の一部を、仮受信部２１２～２１５が備えてもよい。例えば、ＰＤＬジョブ実行部２４３は、ＬＡＮ制御部２０９から通知された印刷データをＲＡＭ１１３や内蔵ストレージ１１４上に一時スプールする機能を有する場合がある。しかし、コンテナ制御部２２１によってＰＤＬコンテナ２２４が起動状態にならないと、データ受信を開始できず、印刷開始がそれだけ遅延する可能性がある。そこで、ＰＤＬジョブ仮受信部２１４は、コンテナ制御部２２１がＰＤＬコンテナ２２４の起動／復帰処理を実行している間に、印刷データをＲＡＭ１１３や内蔵ストレージ１１４上に一時的にスプールしてもよい。そしてＰＤＬジョブ実行部２４３は、ジョブ実行開始時にＰＤＬジョブ仮受信部２１４が既に受信した印刷データを含めて処理を実行する。これにより、コンテナの起動や復帰を待つことなく、画像形成装置１００は印刷データのスプールを開始することができ、処理遅延を軽減することができる。
【００９５】
　＜その他の実施形態＞
　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。
【００９６】
　発明は上記実施形態に制限されるものではなく、発明の精神及び範囲から離脱することなく、様々な変更及び変形が可能である。従って、発明の範囲を公にするために請求項を添付する。
【符号の説明】
【００９７】
　１１１：ＣＰＵ、２１１：デバイス・ジョブ制御部、２２１：コンテナ制御部、５１１：Ｄｏｃｋｅｒ　ｅｎｇｉｎｅ

【図面】

【図１】



【図２】



【図３】

（ネットワークバイトオーダー）

	+0h	+1h	+2h	+3h	+4h	+5h	+6h	+7h	+8h	+9h	+Ah	+Bh	+Ch	+Dh	+Eh	+Fh
0h	制御コマンド バージョン (=0xC0CAFF01)				制御コマンド識別子 制御レスポンス識別子				ユーザ識別子							
10h	制御コマンド／制御レスポンス固有データ長															
20h	制御コマンド／制御レスポンス固有データボディ部															
30h																
40h																
50h	.. ..															

【図４】

```
// 制御コマンド識別子
enum CONTROL_COMMAND{
        // ターゲットオプション
        CC_TARGET_COPY              =0x10, // コピーアプリ
        CC_TARGET_SEND              =0x20, // Sendアプリ
        CC_TARGET_PDL               =0x30, // PDLアプリ
        CC_TARGET_WEBBROW           =0x40, // ウェブブラウザアプリ
        CC_TARGET_FAX               =0x50, // FAXアプリ
        CC_TARGET_USER_START        =0x80, // ユーザーアプリ(開始)
        CC_TARGET_USER_END          =0xFE, // ユーザーアプリ(終了)
        CC_TARGET_DEVICE            =0xFF, // デバイス

        // 受信部制御コマンド
        CC_RECEPTOR_CMD_START       =0xF000, // 受信部起動

        // アプリ制御コマンド
        CC_APPLICATION_CMD_EXECUTE  =0xA100, // 実行
        CC_APPLICATION_CMD_STOP     =0xA500, // 一時停止
        CC_APPLICATION_CMD_C_FLUSH  =0xA700, // キャッシュメモリフラッシュ要求
        CC_APPLICATION_CMD_CANCEL   =0xAF00, // キャンセル

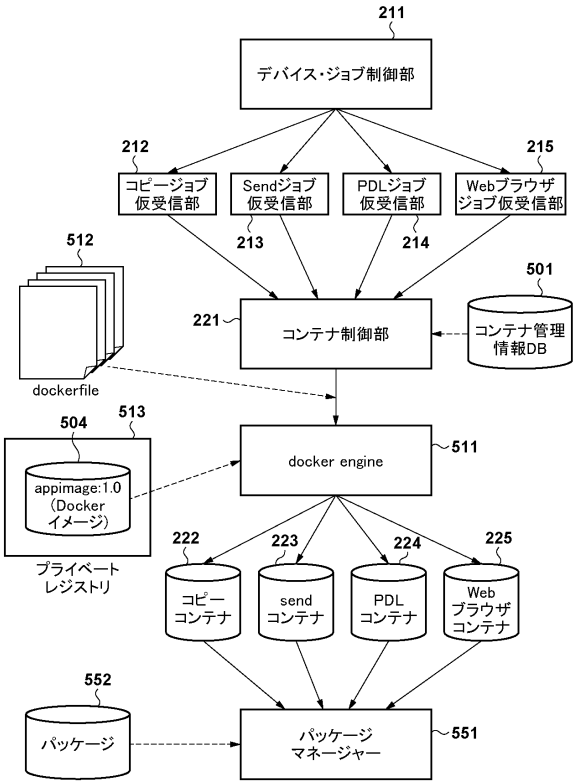
        // コンテナ制御コマンド
        CC_CONTAINER_CMD_CREATE      =0xC000, // コンテナ生成
        CC_CONTAINER_CMD_START      =0xC100, // コンテナ起動
        CC_CONTAINER_CMD_PAUSE      =0xC200, // コンテナ停止
        CC_CONTAINER_CMD_UNPAUSE    =0xC300, // コンテナ復帰
        CC_CONTAINER_CMD_UPDATE     =0xC700, // コンテナ設定変更
        CC_CONTAINER_CMD_STOP       =0xCD00, // コンテナ終了
        CC_CONTAINER_CMD_REMOVE     =0xCE00, // コンテナ破棄
        CC_OONTAINER_CMD_GET_STATUS =0xCF00, // コンテナ状態取得

        // デバイス制御コマンド
        CC_DEVICE_CMD_WAKEUP        =0xD000, // スリープ復帰
        CC_DEVICE_CMD_SLEEP         =0xD100, // スリープ突入
        CC_DEVICE_CMD_PANELOFF      =0xD200, // パネル点灯
        CC_DEVICE_CMD_PANELON       =0xD300, // パネル消灯
        CC_DEVICE_CMD_QUICKOFF      =0xD400, // 高速復帰電源断
        CC_DEVICE_CMD_POWEROFF      =0xDF00, // 電源切断
};

//汎用制御レスポンス識別子I
enum CC_RESPONSE{
        CC_RESPONSE_OK              =0x0000, // 成功
        CC_RESPONSE_NG              =0x0001, // 失敗
};

//コンテナ状態制御レスポンス識別子
enum CC_STATUS{
        CC_RESPONSE_NONEXISTS       =0x8000, // コンテナ未存在状態
        CC_RESPONSE_CREATED         =0x0000, // コンテナ未起動状態
        CC_RESPONSE_RUNNING         =0x0001, // コンテナ起動状態
        CC_RESPONSE_PAUSED          =0x0002, // コンテナ停止状態
        CC_RESPONSE_RESTARTING      =0x0003, // コンテナ再起動中
        CC_RESPONSE_EXITED          =0x0004, // コンテナ終了中
        CC_RESPONSE_REMOVING        =0x0005, // コンテナ削除中
        CC_RESPONSE_DEAD            =0x0006, // コンテナ停止失敗中
};
```

【図５】



【図６】

512

```
1. # Dockerfile for copyserver

2. # コンテナ生成時
3. # 1. Dockerイメージをプライベートレジストリから取得
4. FROM appimage:1.0

5. # 2. ワークディレクトリ作成
6. RUN mkdir -p /var/copy/

7. # 3. 作業ディレクトリに移動
8. WORKDIR /var/copy/

9. # 4. コピーコンテナアプリケーションをインストール
10.RUN [ “pkg-manager”, “install”, “copyserver” ]

11.# 5. コンテナ上でジョブ情報を受け取るポートを開放
12.EXPOSE 20001

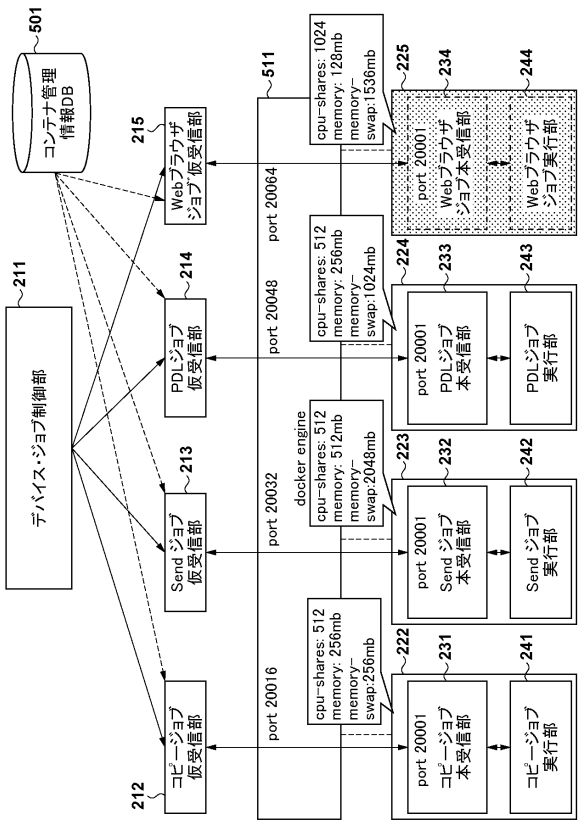
13.# コンテナ動作時
14.# 1. 下記プログラムを実行
15.CMD [ “/apps/copyserver” ]

16.# コンテナ終了時
17.# 1. SIGTERMを送信し、プログラムへ通知する
18.STOPSIGNAL SIGTERM
```

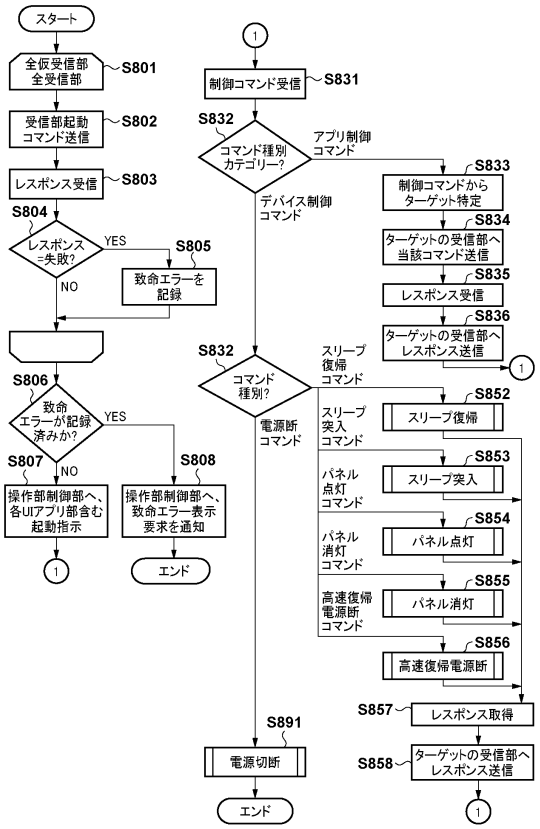
601 602 603 604 605 606 607 600 610

No.	name	本体関連起動	host port	cpu-shares	memory（メモリ）	memory-swap（メモリ+SWAP）	…
1	copyserver	する	20016	512	256mb	256mb	…
2	sendserver	する	20032	512	512mb	2048mb	…
3	pdlserver	する	20048	512	256mb	1024mb	…
4	webbrowsserver	しない	20064	1024	128mb	1536mb	…

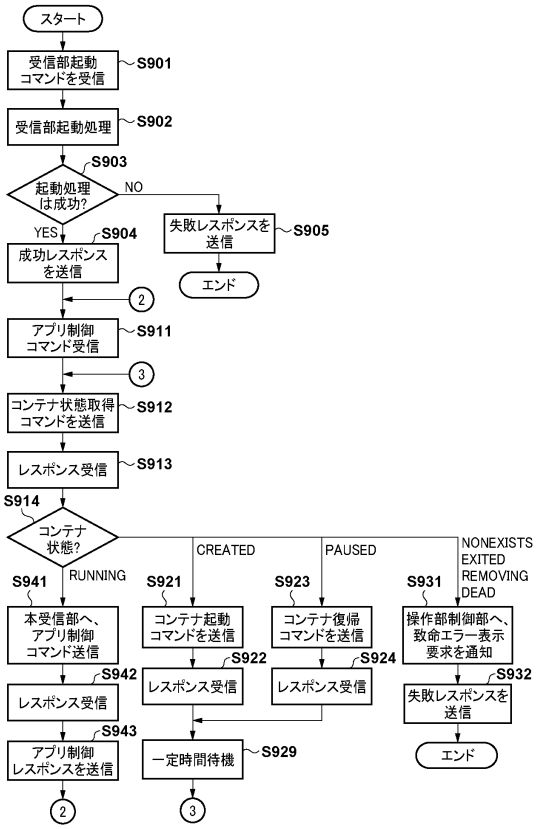
【図７】



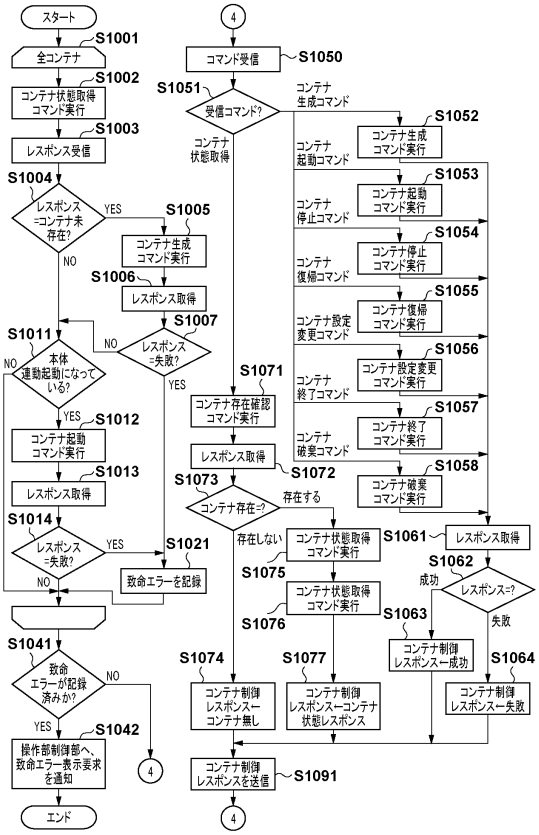
【図８】



【図９】


スタート
受信部起動コマンドを受信
S901
受信部起動処理
S902
起動処理は成功?
S903
NO
YES
失敗レスポンスを送信
S905
成功レスポンスを送信
S904
エンド
2
アプリ制御コマンド受信
S911
3
コンテナ状態取得コマンドを送信
S912
レスポンス受信
S913
S914
コンテナ状態?
RUNNING
CREATED
PAUSED
NONEXISTS
EXITED
REMOVING
DEAD
S941
S921
S923
S931
本受信部へ、アプリ制御コマンド送信
コンテナ起動コマンドを送信
コンテナ復帰コマンドを送信
操作部制御部へ、致命エラー表示要求を通知
S942
S922
S924
S932
レスポンス受信
レスポンス受信
失敗レスポンスを送信
S943
一定時間待機
S929
アプリ制御レスポンスを送信
エンド
2
3

【図１０】


スタート
S1001
全コンテナ
S1002
コンテナ状態取得コマンド実行
S1003
レスポンス受信
S1004
レスポンス=コンテナ未存在?
YES
NO
S1005
コンテナ生成コマンド実行
S1006
レスポンス取得
S1007
レスポンス=失敗?
S1011
本体連動起動になっている?
NO
YES
S1012
コンテナ起動コマンド実行
S1013
レスポンス取得
S1014
レスポンス=失敗?
S1021
致命エラーを記録
S1041
致命エラーが記録済みか?
S1042
操作部制御部へ、致命エラー表示要求を通知
エンド
4
コマンド受信
S1050
S1051
受信コマンド?
コンテナ生成コマンド
コンテナ起動コマンド
コンテナ停止コマンド
コンテナ復帰コマンド
コンテナ設定変更コマンド
コンテナ終了コマンド
コンテナ破棄コマンド
コンテナ状態取得
S1052
コンテナ生成コマンド実行
S1053
コンテナ起動コマンド実行
S1054
コンテナ停止コマンド実行
S1055
コンテナ復帰コマンド実行
S1056
コンテナ設定変更コマンド実行
S1057
コンテナ終了コマンド実行
S1058
コンテナ破棄コマンド実行
S1071
コンテナ存在確認コマンド実行
S1072
レスポンス取得
S1073
コンテナ存在=?
存在する
存在しない
S1074
コンテナ制御レスポンス-コンテナ無し
S1075
コンテナ状態取得コマンド実行
S1076
コンテナ制御レスポンス-コンテナ状態レスポンス
S1077
S1061
レスポンス取得
S1062
レスポンス=?
成功
失敗
S1063
コンテナ制御レスポンス-成功
S1064
コンテナ制御レスポンス-失敗
コンテナ制御レスポンスを送信
S1091
4

フロントページの続き

(56)参考文献　特開２０１９－１７５３９９（ＪＰ，Ａ）
特開２０１８－１２９００３（ＪＰ，Ａ）
特開２０１９－１５３０７９（ＪＰ，Ａ）
特開２０１９－１０１８６６（ＪＰ，Ａ）
特開２０１９－２１９８６６（ＪＰ，Ａ）

(58)調査した分野　(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　９／４４－９／４５５
Ｇ０６Ｆ　３／１２