(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第3817518号
(P3817518)
(45)発行日　平成18年9月6日(2006.9.6)　(24)登録日　平成18年6月16日(2006.6.16)

(51)Int.Cl.		F I		
H04Q 9/00	(2006.01)	H04Q	9/00	301B
G06K 19/00	(2006.01)	H04Q	9/00	331Z
		G06K	19/00	Q

請求項の数 23　(全 100 頁)

(21)出願番号	特願2002-527988 (P2002-527988)
(86)(22)出願日	平成13年9月12日 (2001.9.12)
(65)公表番号	特表2004-508784 (P2004-508784A)
(43)公表日	平成16年3月18日 (2004.3.18)
(86)国際出願番号	PCT/AU2001/001145
(87)国際公開番号	WO2002/023411
(87)国際公開日	平成14年3月21日 (2002.3.21)
審査請求日	平成15年5月12日 (2003.5.12)
(31)優先権主張番号	PR 0073
(32)優先日	平成12年9月12日 (2000.9.12)
(33)優先権主張国	オーストラリア(AU)
(31)優先権主張番号	PR 5593
(32)優先日	平成13年6月8日 (2001.6.8)
(33)優先権主張国	オーストラリア(AU)

(73)特許権者　000001007
キヤノン株式会社
東京都大田区下丸子３丁目３０番２号
(74)代理人　100076428
弁理士　大塚　康徳
(74)代理人　100112508
弁理士　高柳　司郎
(74)代理人　100115071
弁理士　大塚　康弘
(74)代理人　100116894
弁理士　木村　秀二

最終頁に続く

(54)【発明の名称】カード利用サービスへのアクセスのためのシステム

(57)【特許請求の範囲】

【請求項１】

受け入れのためのレセプタクルを有するカード読取り装置へと挿入されるように構成され、印が形成された基板を具備するユーザインタフェースカードに対応したサービスをユーザに提供するサービス提供装置であって、

前記カード読取り装置より、前記印の１つを特定する第１のデータを受信し、

ここで、前記第１のデータは前記インターフェースカードのメモリから読み込んだデータである；

前記インターフェースカードを特定する第２のデータを受信する中央処理装置を有し、

ここで、前記第２のデータは、前記インターフェースカードの挿入時から、前記カード読取り装置の前記レセプタクルから前記インターフェースカードを取り除くまでの間、前記カード読取り装置に対する操作が行われる度に、前記サービス提供装置に転送される；

前記サービス提供装置は、前記第１のデータを、前記第２のデータによって特定されるサービスに対する指示情報としてサービスを提供するように構成される

ことを特徴とするサービス提供装置。

【請求項２】

前記第１のデータは、前記カード読取り装置に設けられた実質的に透明な感圧膜上のユーザ押下位置を定義しする情報データであり、

ここで、前記感圧膜は前記インターフェースカードに重なるように構成される；

前記情報データは前記カードが前記読取り装置へと挿入されていないときは、ユーザによる前記感圧膜の押下に応じて前記読取り装置から受信される
ことを特徴とする請求項１記載のサービス提供装置。
【請求項３】
前記中央処理装置は、現在のフロントアプリケーションに関連づけられたデータとは異なる前記第２のデータを受信した場合に応じて、前記フロントアプリケーションから、複数の更なるアプリケーションのうちの１つへの変更を制御するように構成されることを特徴とする請求項１記載のサービス提供装置。
【請求項４】
前記中央処理装置は、不正なカードが前記カード読取り装置へと挿入されるとユーザに警告する請求項１記載のサービス提供装置。
【請求項５】
前記中央処理装置は、前記データにより定義されるメモリアドレスを利用して前記メモリアドレスに格納されるデータをアクセスするように構成される請求項１記載のサービス提供装置。
【請求項６】
前記メモリアドレスはＵＲＬであることを特徴とする請求項５記載のサービス提供装置。
【請求項７】
受け入れのためのレセプタクルを有するカード読取り装置へと挿入されるように構成され、印が形成された基板を具備するインタフェースカードに対応したサービスをユーザに提供するサービス提供装置であって、
前記カード読取り装置より、前記印の１つを特定する第１のデータを受信する手段と、
ここで、前記第１のデータは、前記カードのメモリから読み込まれる；
前記インターフェースカードを識別する第２のデータを受信する手段とを備え、
ここで、前記第２のデータは、前記インターフェースカードの挿入時から、前記カード読取り装置の前記レセプタクルから前記インターフェースカードを取り除くまでの間、前記カード読取り装置に対する操作が行われる度に、前記カード読取り装置から前記サービス提供装置に転送される；
前記サービス提供装置は、前記第１のデータを、前記第２のデータによって特定されるサービスに対する指示情報としてサービスを提供する
ことを特徴とするサービス提供装置。
【請求項８】
受け入れのためのレセプタクルを有するカード読取り装置へと挿入されるように構成され、印が形成された基板を具備するインタフェースカードに対応したサービスをユーザに提供する方法であって、
前記カード読取り装置から、前記印の１つを特定する第１のデータを受信する工程と、
ここで、前記第１のデータは、前記カードのメモリから読み込まれる；
前記カード読取り装置から、前記インターフェースカードを特定する第２のデータを受信する工程と、
ここで、前記第２のデータは、前記インターフェースカードの挿入時から、前記カード読取り装置の前記レセプタクルから前記インターフェースカードを取り除くまでの間、前記カード読取り装置に対する操作が行われる度に、前記カード読取り装置から前記サービス提供装置に転送される；
前記第１のデータを、前記第２のデータによって特定されるサービスに対する指示情報としてサービスを提供する工程と
を備えることを特徴とする方法。
【請求項９】
コンピュータが読み込み実行することで、受け入れのためのレセプタクルを有するカード読取り装置へと挿入されるように構成され、印が形成された基板を具備するインタフェ

ースカードに対応したサービスをユーザに提供するサービス提供装置として機能させるプログラムであって、
前記カード読取り装置から、前記印の１つを特定する第１のデータを受信する工程と、
ここで、前記第１のデータは、前記カードのメモリから読み込まれる；
前記カード読取り装置から、前記インターフェースカードを特定する第２のデータを受信する工程と、
ここで、前記第２のデータは、前記インターフェースカードの挿入時から、前記カード読取り装置の前記レセプタクルから前記インターフェースカードを取り除くまでの間、前記カード読取り装置に対する操作が行われる度に、前記カード読取り装置から前記サービス提供装置に転送される；
前記第１のデータを、前記第２のデータによって特定されるサービスに対する指示情報としてサービスを提供する工程
として機能させることを特徴とするプログラム。
【請求項１０】
前記サービス提供装置中のコンピュータ可読な媒体に格納されることを特徴とする請求項９記載のプログラム。
【請求項１１】
前記第２のデータは、１つの前記印の選択に応じて前記サービス提供装置に送信されることを特徴とする請求項１に記載のサービス提供装置。
【請求項１２】
前記第２のデータは、１つの前記印の選択が解除されることに応じて前記サービス提供装置に送信されることを特徴とする請求項１に記載のサービス提供装置。
【請求項１３】
前記第２のデータは、前記インターフェースカードが前記カード読取り装置に挿入されることに応じて、前記サービス提供装置に送信されることを特徴とする請求項１に記載のサービス提供装置。
【請求項１４】
前記第２のデータは、前記インターフェースカードが前記カード読取り装置から取り除かれることに応じて、前記サービス提供装置に送信されることを特徴とする請求項１に記載のサービス提供装置。
【請求項１５】
前記第２のデータは、前記選択位置の移動に応じて前記サービス提供装置に送信されることを特徴とする請求項１に記載のサービス提供装置。
【請求項１６】
前記第２のデータはサービス識別子であることを特徴とする請求項１に記載のサービス提供装置。
【請求項１７】
前記識別子は、アプリケーションによる利用のため、ベンダーによって設定されることを特徴とする請求項１６に記載のサービス提供装置。
【請求項１８】
前記サービス識別子は、中央機関によってベンダーに割り当てられることを特徴とする請求項１７に記載のサービス提供装置。
【請求項１９】
第２のデータは、擬似乱数であることを特徴とする請求項１に記載のサービス提供装置。
【請求項２０】
第２のデータは、前記カードが前記レセプタクルに挿入する毎に増加することを特徴とする請求項１に記載のサービス提供装置。
【請求項２１】
第２のデータは、前記インターフェースカードにメモリ無いに格納されていることを特

徴とする請求項１に記載のサービス提供装置。
【請求項２２】
前記サービス提供装置はセットトップボックスであることを特徴とする請求項１に記載のサービス提供装置。
【請求項２３】
前記サービス提供装置はパーソナルコンピュータであることを特徴とする請求項１に記載のサービス提供装置。
【発明の詳細な説明】
【０００１】
本発明は、リモートリーダ装置と共に使用される制御テンプレート又はスマートカードに関し、特に、サービスを提供するカードインタフェースシステムに関する。また、本発明はカードインタフェースシステムに対するコンピュータプログラムを記録したコンピュータ可読な媒体を含むコンピュータプログラム製品に関する。
【０００２】
様々な種類の制御パッドが知られており、かなり多岐に渡る分野で使用されている。通常、このようなパッドは、ユーザが適切な圧力を印加すると信号を発生して関連する制御回路に供給する１つ以上のキー、ボタン、又は感圧領域を含む。
【０００３】
しかし、従来の制御パッドは、キー、ボタン、又は感圧領域の単一構成のみを考慮しているという点で多少制限がある。どの分野でも標準のレイアウトが存在することはまれであり、ユーザは使用する各制御パッドに関して新規のレイアウトについて学ぶことを強いられることが多い。例えば、多くの現金自動預払機（「ＡＴＭ」）及び販売時点電子資金移動（「ＥＦＴＰＯＳ」）装置は、データ入力での要求事項が比較的類似しているにも関わらず、異なるレイアウトを使用している。このため、ユーザは制御パッドごとに押下する必要のあるボタンの位置を判定しなければならず、混乱を招く可能性がある。このような制御パッドは、ユーザの関心、更には、ユーザの使用能力を超える多くのオプションを提供することが多いので問題は悪化することになる。
【０００４】
コンピュータのキーボードなどのためのオーバレイテンプレートが知られている。しかし、これらのテンプレートは設計上柔軟性に乏しく、オーバレイを使用するたびにキーボードが関連付けられているシステムを正確に構成するをユーザに要求している。
【０００５】
既知のシステムの１つは、機器のリモート制御のために設計されたスマートカード読取り装置を伴う。この装置により、例えば、テレビメーカはカードを製造し、このカードをリモート制御筐体及びテレビ受像機と共に供給することができる。顧客は、カードと共に筐体をテレビ受像機用のリモート制御装置として使用することができるようになる。テレビメーカ又はラジオメーカは、自社の製品のために固有のリモート制御装置を製造する必要はないが、リモート制御筐体を固有のカードと共に使用することができる。しかし、上述の概念には、カード上に格納され、関連する装置を制御するのに使用される制御データ（ＰＬＡＹコマンド、ＲＥＣＯＲＤコマンド、ＲＥＷＩＮＤコマンドなど）が装置のメーカによるものであり、そのため、適用が制限されるという欠点がある。
【０００６】
別の既知のシステムは、ビデオ装置、オーディオ装置などをリモート制御するのに使用される「リモートコマンダ（remote commander）」として知られる操作カード読取り装置を伴う。この既知のシステムの操作カードは、リモートコマンダがどのモードで操作しているか及びリモートコマンダからどの制御データが送信されるかを識別するためのカード識別メカニズムを含む。操作カード識別メカニズムは、カードの側面に形成される電極／ノッチ及び操作カード内に格納される識別情報のいずれかの形態をとることができる。操作カード識別メカニズムは、リモートコマンダが識別メカニズムの構成によってビデオテープレコーダ及びテレビ受像機のいずれかのためのコマンドを送信できるように構成するこ

とができる。この既知のシステムにも、ビデオテープレコーダ又はテレビを制御するのに使用される制御データ（ＰＬＡＹコマンド、ＲＥＣＯＲＤコマンド、ＲＥＷＩＮＤコマンドなど）が装置のメーカによるものであり、そのため、適用が制限されるという欠点がある。更に、操作カード識別メカニズムは、ユーザが制御対象装置の変更を希望するたびに構成される必要があり、制御対象のビデオ装置、オーディオ装置などとのインタラクションに使用できないように操作カードに限定される。

【０００７】

更に別の既知のスマートカードシステムは、テレビのチャンネルから情報を受信するための光学部品及びリモートサービスプロバイダ上で実行されているアプリケーションとのリアルタイム双方向通信を提供するためのモデムを伴う。この既知のスマートカードシステムは、既存のホームショッピングアプリケーションなどのリモートサービストランザクションに使用される。この既知のシステムによると、ホームショッピングプログラム情報、商品名、商品の明細、商品価格、商品ＣＭ、及びプログラミングの再実行回数を含む情報をスマートカードへとダウンロードすることができる。スマートカードは、スマートカードのモデムと共にアクセス情報を使用してホームショッピングプログラムの自動サービスコンピュータに自動的にダイヤルし、注文を行なう。しかし、商品購入にスマートカードを使用するたびにアクセス情報をダウンロードしなければならず、商品名及び商品明細により指定される商品を購入する際にしか使用することができないため、このシステムの適用も制限される。

【０００８】

上述のシステムは全てが柔軟性に欠けており、それぞれの適用に制限がある。これらのシステムは、全てが事前に実行されるアプリケーションと共に使用され、メーカによる指定のもの以外のアプリケーションとの間にはインタラクションがない。

【０００９】

本発明の目的は、既存の構成の１つ以上の欠点をほぼ克服するか、あるいは、少なくとも改善することである。

【００１０】

本発明の一面によると、受け入れのためのレセプタクル（receptacle）を有するカード読取り装置へと挿入されるように構成され、印（indicia）が形成された基板を具備するユーザインタフェースカードのユーザにサービスを提供するサービス提供装置であって、
前記カード上に格納されるサービス識別子とユーザにより選択される印に関連するデータとを前記読取り装置から受信する中央処理装置を具備し、前記中央処理装置は前記サービス識別子により識別されるサービスを提供するように構成されることを特徴とするサービス提供装置が提供される。

【００１１】

本発明の別の面によると、受け入れのためのレセプタクルを有するカード読取り装置へと挿入されるように構成され、印が形成された基板を具備するインタフェースカードのユーザにサービスを提供するサービス提供装置であって、
前記カード内に格納されるサービス識別子とユーザにより選択される印に関連するデータとを前記読取り装置から受信する手段と、
前記サービス識別子により識別されるサービスをユーザに提供する手段とを備えるサービス提供装置が提供される。

【００１２】

本発明の更に別の面によると、受け入れのためのレセプタクルを有するカード読取り装置へと挿入されるように構成され、印が形成された基板を具備するインタフェースカードのユーザにサービスを提供する方法であって、
前記カード内に格納されるサービス識別子とユーザにより選択される印に関連するデータとを前記読取り装置から受信する工程と、
前記サービス識別子により識別されるサービスを提供する工程とを備える方法が提供される。

【００１３】
本発明の更に別の面によると、受け入れのためのレセプタクルを有するカード読取り装置へと挿入されるように構成され、印が形成された基板を具備するインタフェースカードのユーザにサービスを提供するサービス提供装置により実行されるプログラムであって、
前記カード内に格納されるサービス識別子とユーザにより選択される印に関連するデータとを前記読取り装置から受信するコードと、
前記サービス識別子により識別されるサービスを提供するコードとを備えるプログラムが提供される。
【００１４】
本発明の更に別の面によると、機器を制御するために表面に形成される少なくとも１つの印とデータを格納するメモリとを有する制御テンプレートに対する読取り装置と共に使用されるサービス提供装置であって、前記カードが前記読取り装置へと挿入されるように構成されるサービス提供装置であって、
少なくともサービス識別子とユーザにより選択される印と関連付けられる更なるデータとを前記読取り装置から受信する中央処理装置を具備し、前記中央処理装置は前記更なるデータを受信すると前記機器を介して前記サービス識別子と関連付けられるサービスを提供するように構成されるサービス提供装置が提供される。
【００１５】
本発明の更に別の面によると、機器を制御するために表面に形成される少なくとも１つの印とデータを格納するメモリとを有する制御テンプレートに対する読取り装置と共に使用されるセットトップボックスであって、前記カードが前記読取り装置へと挿入されるように構成されるセットトップボックスであって、
少なくともサービス識別子とユーザにより選択される印と関連付けられる更なるデータとを具備するデータを前記読取り装置から受信する中央処理装置を具備し、前記中央処理装置は前記更なるデータを受信すると前記機器を介して前記サービス識別子と関連付けられるサービスを提供するように構成されるセットトップボックスが提供される。
【００１６】
本発明の更に別の面によると、機器を制御するために表面に形成される少なくとも１つの印とデータを格納するメモリとを有する制御テンプレートに対する読取り装置と共に使用されるコンピュータであって、前記カードが前記読取り装置へと挿入されるように構成されるコンピュータであって、
少なくともサービス識別子とユーザにより選択される印と関連付けられる更なるデータとを前記読取り装置から受信する中央処理装置を具備し、前記中央処理装置は前記更なるデータを受信すると前記機器を介して前記サービス識別子と関連付けられるサービスを提供するように構成されるコンピュータが提供される。
【００１７】
本発明の更に別の面によると、基板とその上に形成される印とを具備するインタフェースカードを受け入れるレセプタクルを有するカード読取り装置を使用するカードユーザにサービスを提供するサービス提供装置であって、
前記カードに格納されるサービス識別子とユーザにより選択される印に関連するデータとを前記読取り装置から受信する中央処理装置を具備し、前記中央処理装置はユーザにより押下される印に従って前記サービス識別子により識別されるサービスをユーザに提供するサービス提供装置が提供される。
【００１８】
本発明の更に別の面によると、基板とその上に形成される印とを具備する取外し可能なインタフェースカードに重なるように構成されるほぼ透明な感圧膜を有するカード読取り装置を使用するカードユーザにサービスを提供する装置であって、
サービス識別子と前記感圧膜上の押下されるユーザ押下座標とを前記読取り装置から受信し、前記座標を前記印と関連付けられるデータと照合して前記サービス提供装置から前記座標と一致するデータを読む中央処理装置であって、ユーザにより押下される印に従って

前記サービス識別子により識別されるサービスをユーザに提供する中央処理装置を具備するサービス提供装置が提供される。

【００１９】

本発明の更に別の面によると、基板とその上に形成される印とを具備する取外し可能なインタフェースカードに重なるように構成されるほぼ透明な感圧膜を有するカード読取り装置を使用するカードユーザにサービスを提供する装置であって、

前記装置がサービス識別子と前記感圧膜上の押下されるユーザ押下座標とを前記読取り装置から受信する場合に、前記座標を前記印と関連付けられるデータと照合して前記サービス提供装置から前記座標と一致するデータを読み、前記装置が前記印と関連付けられるデータを前記読取り装置から受信する場合にデータを処理する中央処理装置であって、ユーザにより押下される印に従って前記サービス識別子により識別されるサービスをユーザに提供する中央処理装置を具備するサービス提供装置が提供される。

【００２０】

本発明の更に別の面によると、基板とその上に形成される印とを具備する取外し可能なインタフェースカードに重なるように構成されるほぼ透明な感圧膜を有するカード読取り装置を使用するカードユーザにサービスを提供する装置であって、

前記感圧膜上のユーザ押下の接触座標を読取り装置から受信し、前記受信された座標に基づいてディスプレイに表示されるカーソルを移動する中央処理装置を具備するサービス提供装置が提供される。

【００２１】

本発明の更に別の面によると、インタフェースカードを受け入れるレセプタクルを有するカード読取り装置を使用するカードユーザにサービスを提供するサービス提供装置であって、

前記読取り装置におけるカード挿入の現在のセッションを識別し、カードが前記読取り装置へと挿入されるたびに増分される数値であるセッション識別子を前記読取り装置から受信し、前記受信されたセッション識別子を先に受信されたセッション識別子と比較することによって前記挿入されたカードの有効性を判定する中央処理装置を具備するサービス提供装置が提供される。

【００２２】

本発明の更に別の面によると、カスタマイズ可能なユーザインタフェースシステムに対するソフトウェアアーキテクチャであって、

ユーザインタフェース信号を前記システムの全体にわたって解釈可能な形態へと変換するインタフェースモジュールと、

前記インタフェース信号を受信し、前記アーキテクチャのコンポーネント間で前記信号及び更なる信号を送受信するように動作可能なイベントマネージャと、

ユーザインタフェースの活動を識別し、前記ユーザインタフェースに関連する動作と関連付けられると共に前記ユーザインタフェースに関連する１つ以上のアプリケーションの各々の開始及び終了を前記ユーザインタフェースのために管理するランチャモジュールの動作を促すマスタランチャとを具備するアーキテクチャが提供される。

【００２３】

本発明の更に別の面によると、ソフトウェアアーキテクチャを使用してカスタマイズ可能なユーザインタフェースシステム内で複数のアプリケーションを起動する方法であって、

ユーザインタフェース信号を前記システムの全体にわたって解釈可能な形態へと変換し、

前記インタフェース信号を受信し、前記アーキテクチャのコンポーネント間で前記信号及び更なる信号を送受信し、

ユーザインタフェースの活動を識別し、前記ユーザインタフェースに関連する動作と関連付けられると共に前記ユーザインタフェースに関連する１つ以上のアプリケーションの各々の開始及び終了を前記ユーザインタフェースのために管理するランチャモジュールの動作を促すことを特徴とする方法が提供される。

【００２４】

本発明の更に別の面によると、カスタマイズ可能なユーザインタフェースシステムに対するソフトウェアアーキテクチャであって、
ユーザインタフェース信号を前記システムの全体にわたって解釈可能な形態へと変換するインタフェースモジュールと、
前記インタフェース信号を受信し、前記アーキテクチャのコンポーネント間で前記信号及び更なる信号を送受信するように動作可能なイベントマネージャと、
ユーザインタフェースの活動を識別し、前記ユーザインタフェースに関連する動作と関連付けられると共に前記ユーザインタフェースに関連し各々が前記ユーザインタフェースと関連付けられる対応のサービス識別子を利用してアクセス可能な１つ以上のアプリケーションの各々の開始及び終了を前記ユーザインタフェースのために管理するランチャモジュールの動作を促すマスタランチャとを具備するアーキテクチャが提供される。

【００２５】

本発明の更に別の面によると、ソフトウェアアーキテクチャを使用してカスタマイズ可能なユーザインタフェースシステム内で複数のアプリケーションを起動するプログラムであって、
ユーザインタフェース信号を前記システムの全体にわたって解釈可能な形態へと変換するコードと、
前記インタフェース信号を受信し、前記アーキテクチャのコンポーネント間で前記信号及び更なる信号を送受信するコードと、
ユーザインタフェースの活動を識別し、前記ユーザインタフェースに関連する動作と関連付けられると共に前記ユーザインタフェースに関連する１つ以上のアプリケーションの各々の開始及び終了を前記ユーザインタフェースのために管理するランチャモジュールの動作を促すコードとを具備するプログラムが提供される。

【００２６】

本発明の更に別の面によると、カードユーザにサービスを提供するサービス提供装置であって、各々がイベントデータに対応するアイコンが表面に形成されたインタフェースカードを受け入れるレセプタクルを有するカード読取り装置と共に使用されるように構成されるサービス提供装置であって、
ユーザにより選択されるアイコンに関連する前記イベントデータを前記カード読取り装置から受信し、前記インタフェースカード内に格納されるサービス識別子に対応するアプリケーションを実行してユーザにサービスを提供する中央処理装置を具備するサービス提供装置が提供される。

【００２７】

本発明の更に別の面によると、各々がイベントデータに対応するアイコンが表面に形成されたインタフェースカードを受け入れるレセプタクルを有するカード読取り装置を利用してカードユーザにサービスを提供する方法であって、
ユーザにより選択されるアイコンに関連する前記イベントデータを前記カード読取り装置から受信する工程と、
前記インタフェースカード内に格納されるサービス識別子に対応するアプリケーションを実行して前記ユーザにサービスを提供する工程とから成る方法が提供される。

【００２８】

本発明の更に別の面によると、各々がイベントデータに対応するアイコンが表面に形成されたインタフェースカードを受け入れるレセプタクルを有するカード読取り装置を利用してカードユーザにサービスを提供するプログラムであって、
ユーザにより選択されるアイコンに関連する前記イベントデータを前記カード読取り装置から受信するコードと、
前記インタフェースカード内に格納されるサービス識別子に対応するアプリケーションを実行して前記ユーザにサービスを提供するコードとを具備するプログラムが提供される。

【００２９】

本発明の更に別の面によると、受け入れのためのレセプタクルを有するカード読取り装置

へと挿入されるように構成され、印が形成された基板を具備するインタフェースカードのユーザにサービスを提供するサービス提供装置であって、
前記カード上に格納される識別子とユーザにより選択される印に関連するデータとを前記読取り装置から受信し、前記受信された識別子に対応するアプリケーション位置を取得する中央処理装置を具備するサービス提供装置が提供される。
【００３０】
本発明の更に別の面によると、受け入れのためのレセプタクルを有するカード読取り装置へと挿入されるように構成され、印が形成された基板を具備するインタフェースカードのユーザにサービスを提供するサービス提供装置により実行されるプログラムであって、
前記カード上に格納される識別子とユーザにより選択される印に関連するデータとを前記読取り装置から受信するコードと、
前記受信された識別子に対応するアプリケーション位置を取得するコードとを具備するプログラムが提供される。
【００３１】
本発明のその他の面も開示される。
【００３２】
【本発明の実施の形態】
図面を参照しながら本発明の１つ以上の実施形態を説明する。
【００３３】
添付のいずれか１つ以上の図面において、同じ符号を有するステップ及び／又は特徴に言及する場合、特に明記されていない限り、以下の記述においては、これらのステップ及び／又は特徴は、同じ機能又は動作を有するものとする。
【００３４】
以下に開示される各実施形態は、主に、リモート制御システム、現金自動預払機、ビデオゲームコントローラ、及びネットワークアクセスと共に使用されるべく開発されたものであり、これらの適用例及びその他の適用例を参照して説明する。しかし、本発明がこれらの使用分野に限定されないことは明らかであろう。
【００３５】
説明を簡単にする目的で、以下の説明では、第１．０節（section）から第１３．０節に分割する。各節は関連する項（subsectioin）を有する。
【００３６】
１．０　カードインタフェースシステムの概要
図１は、カードレセプタクル４及び表示領域６を定義する筐体２を有するリモートリーダ１を示す。データ読取り手段が、露出した電気接点７及び関連する制御回路（不図示）の形態で設けられる。また、リモートリーダ１は、表示領域６を覆うタッチパネル８を形成するほぼ透明な感圧膜の形態のセンサ手段を含む。ここで開示されるリモートリーダ１は、タッチパネル８を形成するほぼ透明な感圧膜を用いて説明されるが、代替の技術をほぼ透明なタッチパネルとして使用できることが当業者には明らかであろう。例えば、タッチパネルは電気抵抗の高いものにすることも、感温性のものにすることもできる。リモートリーダ１は、ユーザインタフェースカードと共に使用するように構成される。図１から図３に示すカードにおいて、ユーザインタフェースカードは電子スマートカード１０Ａの形態をとる。スマートカード１０Ａは、４方向コントローラ２０、上面１６に印刷された「ジャンプボタン」２２、「キックボタン」２４、「開始ボタン」、及び「終了ボタン」の形態の種々の制御印１４を有する積層基板１２を含む。プロモーション素材又は教示的な素材などの他の非制御印を制御印の横に印刷することができる。例えば、図２に示すように、広告素材２６をスマートカード１０Ａの前面又は裏面２７に印刷することができる。
【００３７】
図３に示すように、スマートカード１０Ａは制御印と関連付けられるデータに対してオンボードメモリチップ１９の形態の記憶手段を含む。また、スマートカード１０Ａは、オンボードメモリチップ１９に接続されると共にリモートリーダ１上の露出した接点７に対応

する電気データ接点１８を含む。

【００３８】

図３に示すように、上面１６は、制御印１４が印刷された粘着ラベル６０により形成されても良い。ここで、制御印１４は「終了ボタン」及び方向コントローラ２０の右矢印「ボタン」に対応する。ラベル６０は積層基板１２に貼付される。一般家庭のユーザは、Canon, Inc製のカラーBUBBLEJET（登録商標）プリンタなどのプリンタを使用することによって、特定のスマートカード１０Ａと共に使用するための適切なラベルを印刷することができる。また、制御印１４を積層基板に直接印刷することも、各制御印に別々の粘着ラベルを使用することもできる。

【００３９】

使用の際、スマートカード１０Ａはカードレセプタクル４へと挿入されるので、感圧タッチパネル８はスマートカード１０Ａの上面１６を覆うようになる。このとき、制御印は透明な感圧タッチパネル８を通して表示領域６内に見えることになる。

【００４０】

リーダ１の露出した接点７及び関連回路は、スマートカード１０Ａの制御テンプレートレセプタクル４への挿入時に自動的に、あるいは、リモートリーダ１からの信号に応答して選択的に、メモリチップ１９から制御印１４と関連付けられた格納データを読むように構成される。信号は、例えば、露出した接点７及びデータ接点１８を介してスマートカード１０Ａに送信することができる。

【００４１】

制御印１４と関連付けられたデータが一度読み取られると、ユーザは感圧タッチパネル８の制御印１４に重なる各領域を押下することができる。感圧タッチパネル８にかかる圧力を検知し、格納データを参照することによって、リモートリーダ１はユーザが選択したのがどの制御印１４であるかを推定することができる。例えば、ユーザが感圧タッチパネル８の「キックボタン」２４の付近に圧力をかける場合、リモートリーダ１は圧力が印加された位置を推定し、格納データを参照し、「キック」ボタン２４が選択されたと判定するように構成される。この情報を使用して、関連するビデオゲームコンソール（従来の構成のもの、不図示）などの外部装置を制御することができる。以上の説明から、制御印１４は実際にはボタンでないことが認識されるであろう。制御印１４は、タッチパネル８のマッピングデータ及び機能との対応する結び付きによって、リモート制御装置と従来から関連付けられてきたボタンをエミュレートするように動作するユーザ選択可能な機能である。

【００４２】

１つの有利な実現例において、リモートリーダ１は、ユーザにより選択される印に関連する情報を送信するための赤外線（ＩＲ）送信機又は無線周波数（ＲＦ）送信機などの送信機（従来の構成のもの、不図示）を含む。図１に示すように、リモートリーダ１はＩＲ発光ダイオード（ＬＥＤ）２５を有するＩＲ送信機を組み込む。制御印１４のうちの１つを選択すると、リモートリーダ１は選択された印に関連する情報をリモートコンソール（図１では示されていない）に送信させる。リモートコンソールでは、対応するＩＲ受信機又はＲＦ受信機により、リーダ１のユーザがプレーするゲームなどの何らかの機能を制御する際に使用される情報を検知・復号化することができる。

【００４３】

直接ハード配線を含む任意の適切な送信方法を使用してリモートリーダ１からリモートコンソールへと情報を送信することができる。更に、先に説明したのと逆の方向に送信するために、リモートコンソール自体が送信機を組み込み、リモートリーダ１が受信機を組み込むこともできる。リモートコンソールからリモートリーダ１への送信は、例えば、ハンドシェーキングデータ、セットアップ情報、又はリモートコンソールからリモートリーダ１に送信することが望まれる任意の形態の情報を含むことができる。

【００４４】

図４に戻ると、制御カード１０Ｂが示される。制御カード１０Ｂは、制御印（不図示）を

有する積層基板１２を含む。制御カード１０Ｂにおいて、記憶手段は制御カード１０Ｂの裏面２７の縁部２８に沿って形成される磁気ストリップ２９の形態をとる。制御印と関連付けられる格納データは、従来のように磁気ストリップ２９に格納されても良い。この構成の対応するリーダ（不図示）は、対応する制御テンプレートレセプタクルへの入口部又はその付近に位置する磁気読取りヘッドを含む。制御カード１０Ｂがカードレセプタクルへと差し込まれるとき、格納データは磁気読取りヘッドにより磁気ストリップ２９から自動的に読み取られる。リーダ１は図１のカード１０Ａに対応する方法で操作されても良い。

【００４５】

図５は制御カード１０Ｃの形態の別のカードを示す。制御カード１０Ｃにおいて、記憶手段は機械読取り可能な印の形態をとる。図５のカード１０Ｃにおいて、機械読取り可能な印は、カード１０Ｃの裏面２７の縁部３８に沿って形成されるバーコード３６の形態をとる。格納データは符号化されているのが好ましく、図示する位置に印刷される。図５のカード１０Ｃの対応するコントローラ（不図示）は、関連する制御テンプレートレセプタクルへの入口部又はその付近に位置する光学読取りヘッドを含むことになる。カード１０Ｃがカードレセプタクルへと差し込まれるとき、格納データは光学読取りヘッドによりバーコード３６から自動的に読み取られる。また、バーコードは、カード１０Ｃを挿入する直前にリーダと関連するバーコードリーダを使用して走査することも、カード１０Ｃが完全に挿入されてから内部のバーコードリーダスキャナにより走査することもできる。カード１０Ｃは、その後図１のカード１０Ａに対応する方法で操作されても良い。バーコードの位置、向き、及び符号化は、特定のアプリケーションに適合するように変更可能であることが認識されるであろう。更に、エンボス加工の機械読取り可能な図形、印刷された英数字、パンチングやその他の方法で形成された切抜き、光学的な印又は光磁気の印、２次元バーコードを含むその他の任意の形態の機械読取り可能な印を使用することができる。

【００４６】

図６（ａ）は、カードインタフェースシステム６００Ａのハードウェアアーキテクチャを示す。システム６００Ａにおいて、リモートリーダ１は通信ケーブル３を介してパーソナルコンピュータシステム１００に配線接続される。あるいは、配線接続の代わりに無線周波数送受信機又はＩＲ送受信機１０６を使用して、リモートリーダ１と通信を行なうことができる。パーソナルコンピュータシステム１００は画面１０１及びコンピュータモジュール１０２を含む。パーソナルコンピュータシステム１００については、図７を参照して以下で更に詳細に説明する。キーボード１０４及びマウス２０３も設けられる。

【００４７】

システム６００Ａは、上述のスマートカード１０Ａと同様の構成のスマートカード１０Ｄを含む。スマートカード１０Ｄはプログラム可能であり、サードパーティにより作製又はカスタマイズすることができる。この場合、サードパーティは、カード１０Ｄ及び／又はカードリーダ１のメーカ以外の業者であっても良い。サードパーティは、スマートカード１０Ｄ自体の最終ユーザであっても、メーカとユーザとの間の仲介業者であっても良い。図６（ａ）のシステム６００Ａによると、スマートカード１０Ｄは１回の接触操作でコンピュータ１００と通信を行ない、インターネットなどのネットワーク２２０を介してサービスを取得するようにプログラム・カスタマイズすることができる。コンピュータ１００は、特定のプロトコルに従ってリモートリーダ１から通信ケーブル３を介して送信される信号を解釈するように動作する。このプロトコルについては詳細に後述する。コンピュータ１００は、触れられた制御印に従って選択された機能を実行し、ネットワーク２２０を介してデータを送受信するように構成することができる。このように、コンピュータ１００は、リモートサーバコンピュータ１５０、１５２上に格納されたアプリケーション及び／又はデータへのアクセスを許可し、表示装置１０１での適切な再現を可能にすることができる。

【００４８】

図６（ｂ）は、カードインタフェースシステム６００Ｂのハードウェアアーキテクチャを

示す。システム６００Ｂにおいて、出力インタフェースに接続するセットトップボックス６０１においてローカルにサービスを取得するためにリモートリーダ１をプログラムすることができる。出力インタフェースは、ここではデジタルテレビ受信機などの音声映像出力装置１１６の形態をとる。セットトップボックス６０１は、リモートリーダ１から特定のプロトコル（詳細に後述）に従って受信される信号１１２を解釈するように動作する。この信号１１２は、電気信号であっても、無線周波数信号であっても、あるいは赤外線（ＩＲ）信号であっても良い。セットトップボックス６０１は、触れられた制御印に従って選択された機能を実行し、出力装置１１６での適切な再現を可能にするように構成することができる。また、セットトップボックス６０１は、信号１１２を通信に適した形態に変換し、コンピュータ１００への適切な送信を行なうように構成することができる。このため、コンピュータ１００は、制御印に従って選択された機能を実行し、セットトップボックス６０１にデータを提供することで出力装置１１６での適切な再現を可能にすることができる。セットトップボックス６０１については、図４２を参照して以下で詳細に説明する。

【００４９】

システム６００Ｂの１つの適用例において、スマートカード１０Ｄはサービスをリモートで又はローカルに取得するためにプログラムすることができる。例えば、スマートカード１０Ｄは、リモートサーバコンピュータ１５０、１５２上に格納されたアプリケーション及び／又はデータをネットワーク２２０を介して取得し、これをセットトップボックス６０１へとロードするようにプログラムすることができる。後者の場合、カードはセットトップボックス６０１上のロードされたアプリケーションからサービスを取得するようにプログラムすることもできる。

【００５０】

特に明記する場合を除き、以降、システム６００Ａ及び６００Ｂをシステム６００と総称的に呼ぶ。更に、特に明記する場合を除き、以降、スマートカード１０Ａ、１０Ｂ、１０Ｃ、及び１０Ｄを総称的にスマートカード１０と呼ぶ。

【００５１】

図７は、システム６００の汎用コンピュータシステム１００を示す。このシステムを使用してカードインタフェースシステムを実行させると共に、スマートカード１０をプログラムするためのソフトウェアアプリケーションを実行させることができる。コンピュータシステム１００は、コンピュータモジュール１０２、キーボード１０４及びマウス２０３などの入力装置、並びにプリンタ（不図示）及び表示装置１０１を含む出力装置を具備する。変復調（モデム）送受信装置２１６は、コンピュータモジュール１０２が、例えば、電話回線２２１又はその他の機能媒体を介して接続可能な通信ネットワーク２２０と通信を行なうのに使用される。モデム２１６は、インターネット及び構内通信網（ＬＡＮ）又は広域通信網（ＷＡＮ）などのその他のネットワークシステムへのアクセス権を取得するのに使用することができる。

【００５２】

コンピュータモジュール１０２は、通常、少なくとも１つの中央処理装置（ＣＰＵ）２０５、例えば、半導体ランダムアクセスメモリ（ＲＡＭ）及び読出し専用メモリ（ＲＯＭ）から形成されるメモリユニット２０６、ビデオインタフェース２０７とキーボード１０４及びマウス２０３用のＩ／Ｏインタフェース２１３とを含む入出力（Ｉ／Ｏ）インタフェース、書込み装置２１５、並びにモデム２１６用のインタフェース２０８を含む。記憶装置２０９が設けられるが、この記憶装置２０９は、通常、ハードディスクドライブ２１０及びフロッピーディスクドライブ２１１を含む。磁気テープドライブ（不図示）を使用しても良い。ＣＤ－ＲＯＭドライブ２１２は、通常、不揮発性のデータソースとして設けられる。コンピュータモジュール２０１の構成要素２０５から２１３は、通常、相互接続バス２０４を介して、コンピュータシステム１０２の動作モードが、当業者には既知の従来の動作モードになるように通信を行なう。上述の構成を実施可能なコンピュータの例としては、ＩＢＭのコンピュータ及びその互換機、Sun Sparcstation、又はそれを進化させた

同様のコンピュータシステムがある。
【００５３】
通常、システム６００のソフトウェアプログラムはハードディスクドライブ２１０に常駐し、ＣＰＵ２０５による実行の際に読み出されて制御される。ソフトウェアアプリケーションプログラムとネットワーク２２０から取り込まれるデータとの中間記憶装置は、半導体メモリ２０６を使用し、場合によってはハードディスクドライブ２１０と協働させるように使用して達成されても良い。アプリケーションプログラムは、ＣＤ－ＲＯＭ又はフロッピーディスク上に符号化された形でユーザに供給され、対応するドライブ２１２又は２１１を介して読み取られる場合もあれば、ネットワーク２２０からモデム装置２１６を介してユーザが読み取る場合もある。更に、ソフトウェアは、磁気テープ、ＲＯＭ又は集積回路、光磁気ディスク、コンピュータモジュール１０２と別の装置との間の無線又は赤外線の伝送チャネル、スマートカード及びコンピュータのＰＣＭＣＩＡカードなどのコンピュータ可読なカード、並びに電子メール送受信及びウェブサイトなどに記録された情報などを含むインターネット又はイントラネットを含む他のコンピュータ可読な媒体からコンピュータシステム１０２へとロードすることもできる。以上の例は、関連するコンピュータ可読な媒体の一例に過ぎない。添付の請求の範囲により定義される本発明の趣旨から逸脱することなく、その他のコンピュータ可読な媒体も実施することができる。
【００５４】
スマートカード１０は、コンピュータモジュール１０２のＩ／Ｏインタフェース２１３に接続される書込み装置２１５を用いてプログラムすることができる。書込み装置２１５は、スマートカード１０上のメモリにデータを書き込む機能をもつことができる。また、書込み装置２１５は、スマートカード１０の上面に図形を印刷する機能を有するのが好ましい。書込み装置２１５は、スマートカード１０上のメモリからデータを読み取る機能ももつことができる。まず、ユーザはスマートカード１０を書込み装置２１５へと挿入する。ユーザが汎用コンピュータ１０２のキーボード１０４を介して所要のデータを入力すると、ソフトウェアアプリケーションが書込み装置２１５を介してこのデータをスマートカードのメモリに書き込む。格納データがバーコードの使用などの光学的な復号化用に符号化されている場合、書込み装置は符号化されたデータをスマートカード１０に印刷することができる。
【００５５】
図４２は、システム６００のセットトップボックス６０１を示す。このセットトップボックス６０１を使用してリモートリーダ１から受信された信号１１２を解釈することができる。一部の実現例では、セットトップボックス６０１は実質的にコンピュータモジュール１０２の規模を変更したものとなる。セットトップボックス６０１は、通常、少なくとも１つのＣＰＵユニット４３０５、例えば、半導体ランダムアクセスメモリ（ＲＡＭ）及び読出し専用メモリ（ＲＯＭ）から形成されるメモリユニット４３０６、並びにデジタルテレビ１１６用のＩ／Ｏインタフェース４３１３、信号１１２を送受信するためのＩＲ送受信機を有するＩ／Ｏインタフェース４３０８、ネットワーク２２０に接続するためのＩ／Ｏインタフェース４３１７とを少なくとも含む入出力（Ｉ／Ｏ）インタフェースを含む。セットトップボックス６０１の構成要素４３０５、４３０６、４３１３、４３１５、及び４３１７は、通常、相互接続バス４３０４を介して、動作モードが従来の動作モードになるように通信を行なう。リモートリーダ１又はネットワーク２２０から受信されるデータの中間記憶装置は、半導体メモリ４３０６を使用して達成されても良い。また、セットトップボックスは記憶装置２０９と同様の記憶装置（不図示）を含むことができる。
【００５６】
カードインタフェースシステム６００については、以下の段落で詳細に説明する。
【００５７】
２．０　カードインタフェースシステムソフトウェアアーキテクチャ
２．１　ソフトウェアアーキテクチャのレイアウト
システム６００により表されるハードウェアアーキテクチャに対するソフトウェアアーキ

テクチャ２００がほぼ図８に示される。アーキテクチャ２００は、幾つかの別個のプロセスコンポーネント及び１つのプロセスクラスへと分割することができる。別個のプロセスには、口語で「Ｉ／Ｏデーモン」３００とも呼ばれるＩ／Ｏインタフェース３００、イベントマネージャ３０１、ディスプレイマネージャ３０６、（アプリケーション）ランチャ３０３、及びディレクトリサービス３１１が含まれる。プロセスクラスは１つ以上のアプリケーション３０４により形成される。ここで説明するアーキテクチャ２００には、セットトップボックス６０１により通常形成されるスマートカードリモート接続ごとに、１つのＩ／Ｏデーモン３００、１つのイベントマネージャ３０１、１つのディスプレイマネージャ３０６、及び１つのランチャ３０３と、各ランチャ３０３を稼動させているコンピュータ１００（例えば、サーバコンピュータ１５０、１５２）ごとに１つのマスタランチャ（不図示）と、全てのシステムに対して少なくとも１つのディレクトリサービス３１１とが存在する。ランチャ３０３は、サービスの名前又は位置あるいはそのサービスに対して使用されるアプリケーション３０４の名前又は位置を示すリソースロケータ（ＵＲＬなど）へとサービスデータを変換するためにディレクトリサービス３１１に問合せを行なう。

【００５８】

アーキテクチャ２００は、図８の破線により示されるように、６つの別個の部分１０１、３０７、３０９、３１２、３１３、及び６０１へと物理的に分離することができる。各部分は物理的に別々の計算装置上で実行させることができる。システム６００の各部分間の通信は、伝送制御プロトコル／インターネットプロトコル（ＴＣＰ／ＩＰ）ストリームを使用して実行される。また、各部分１０１、３０７、３０９、３１２、３１３、及び６０１は、同じマシン上で実行させることもできる。

【００５９】

図６（ａ）のシステム６００Ａにおいて、プロセスコンポーネント３００、３０１、３０３、３０４、及び３０６の全てはコンピュータ１００上で実行させることができる。イベントマネージャ３０１、ランチャ３０３、及びディスプレイマネージャ３０６は、コンピュータ１００のハードディスク２０９に格納され、ＣＰＵ２０５による実行時に読み出して制御することができる１つの実行可能なプログラムへと統合されるのが好ましい。ディレクトリサービス３１１は、同じコンピュータ１００又はネットワーク２２０を介してコンピュータ１００に接続される別のコンピュータ（サーバ１５０など）上で実行される。

【００６０】

図６（ｂ）のシステム６００Ｂにおいて、コンポーネント３００から３０４及び３０６の全てはセットトップボックス６０１から実行させることができる。この例において、コンポーネント３００から３０４及び３０６はセットトップボックス６０１のメモリ４３０６に格納することができ、ＣＰＵ４３０５による実行時に読み出して制御することができる。ディレクトリサービス３１１は、コンピュータ１００上で実行させることができると共にコンピュータ１００のメモリ２０６に格納することができる。ディレクトリサービス３１１はＣＰＵ２０５による実行時に読み出して制御することができる。また、ディレクトリサービス３１１をセットトップボックス６０１上で実行させたり、あるいは、その機能をランチャ３０３により実行させたりすることもできる。

【００６１】

セットトップボックス６０１がシステム６００をローカルに実行させるのに十分な能力をもたない場合、Ｉ／Ｏデーモン３００のみをセットトップボックス６０１上で実行させれば良く、アーキテクチャ２００のその他の部分（すなわち、プロセスコンポーネント３０１、３０３、３０４、３０６、及び３１１）は他のサービス（１５０、１５２）上においてリモートで実行させることができる。他のサービスには、ネットワーク２２０を介してアクセスすることができる。この例では、Ｉ／Ｏデーモン３００はセットトップボックス６０１のメモリ４３０６に格納することができ、ＣＰＵ４３０５による実行の際に読み出して制御することができる。このようなシステムの機能部分も図８に示すように分割することができる。

【００６２】

２．１．１　Ｉ／Ｏデーモン
Ｉ／Ｏデーモン３００は、リモートリーダ１から受信されたデータグラムをＴＣＰ／ＩＰストリームへと変換するプロセスコンポーネントである。このＴＣＰ／ＩＰストリームは、イベントマネージャ３０１へと送信することができると同時にイベントマネージャ３０１から受信することもできる（例えば、双方向プロトコルを使用するとき）。リモートリーダ１は任意の適切なデータ形式を使用することができる。Ｉ／Ｏデーモン３００はリモートリーダ１のデータ形式の変更に影響されないのが好ましく、リモートリーダ１の複数の構成に対処することができる。システム６００の１つの有利な実現例では、Ｉ／Ｏデーモン３００はイベントマネージャ３０１へと統合される。

【００６３】
システム６００Ａにおいて、ユーザがコンピュータ１００を起動することによってスマートカードシステム６００を開始し、イベントマネージャ３０１が開始されているとき、Ｉ／Ｏデーモン３００が開始される。また、ユーザがセットトップボックス６０１の電源を入れることによってシステム６００を開始するとき、Ｉ／Ｏデーモン３００が開始される。

【００６４】
Ｉ／Ｏデーモン３００については第９．０節を参照して以下で更に詳細に説明する。

【００６５】
２．１．２　イベントマネージャ
イベントマネージャ３０１は、全ての通信がイベントマネージャ３０１を経由するという点でアーキテクチャ２００の中心を形成する。イベントマネージャ３０１は、リモートリーダ１により生成され、Ｉ／Ｏデーモン３００により中継される全てのイベントを収集するように構成される。これらのイベントは、種々のプロセスコンポーネント３００から３０４及び３０６並びに実行中の各アプリケーションへと再分配される。また、イベントマネージャ３０１は、イベントが有効なヘッダ、正しいデータ長を有することをチェックするように構成されるが、通常、イベントが正しい形式であることをチェックするようには構成されていない。ここでの「イベント」は、Ｉ／Ｏデーモン３００、ランチャ３０３、又はアプリケーション３０４からの単一のデータトランザクションを表す。

【００６６】
異なるシステム間でのプロトコルの変更にはイベントマネージャ３０１が対処する。可能な場合には、現在実行中のアプリケーション３０４により理解されるデータ形式に適合するようにイベントを書き換えることができる。書き換えが可能でない場合、イベントマネージャ３０１は送信元のアプリケーション３０４にエラーを報告する。例えば、複数枚のスマートカードを稼働させるシステムにおいて、異なるデータ形式が使用されている場合、イベントマネージャ３０１は混乱の発生を最小限に抑えることを保証するのが好ましい。

【００６７】
イベントマネージャ３０１は、表示画面又はその他の出力装置１１６にはその存在を示さない。しかし、イベントマネージャ３０１は、どのアプリケーションが現在必要とされており（すなわち、「フロント」アプリケーション）、現在画面１０１に表示されるべきであるかについてディスプレイマネージャ３０６に指示するように構成することができる。イベントマネージャ３０１は、ランチャ３０３からアプリケーション３０４へと渡されるメッセージからこの情報を推定する。これについては、第１０．０節を参照して以下で詳細に説明する。

【００６８】
イベントマネージャ３０１は、Ｉ／Ｏデーモン接続の受信の有無を常に聞き取るように構成することも、あるいは、システム６００を開始させることもできる。使用される方法はシステム６００の全体的な構成によって決まる。イベントマネージャ３０１がシステム６００を開始させることができるか、あるいは、セットトップボックス６０１がＩ／Ｏデーモン３００の接続の受信を使用してシステム６００を開始させることができる。イベント

マネージャ３０１については、第７．０節を参照して以下で更に詳細に説明する。

【００６９】

２．１．３　マスタランチャ

シン（thin）クライアントコンピュータが利用され、各々が１つのセットトップボックスを担当する複数のランチャ３０３が実行されている場合、イベントマネージャ３０１と直接通信を行なうマスタランチャ（不図示）を使用することができる。マスタランチャは、２つ以上のイベントマネージャ３０１がシステム６００上で実行されている場合に、各イベントマネージャ３０１に対応するランチャ３０３を開始するのに使用される。まず、Ｉ／Ｏデーモン３００がイベントマネージャ３０１に接続するとき、マスタランチャがイベントマネージャ３０１に対して第１のプロセスを開始するよう、イベントマネージャ３０１が要求する。この第１のプロセスは、通常、任意のスマートカードアプリケーション３０４に対するランチャ３０３である。また、マスタランチャは、イベントマネージャ３０１が要求するときにはアプリケーション３０４のランチャ３０３を停止し、イベントマネージャ３０１にランチャ３０３が終了したことを通知するように構成することができる。

【００７０】

関連するスマートカードアプリケーション３０４を実行中の物理的に別々のサーバ（例えば、１５０、１５２）ごとに１つのマスタランチャが実行されるのが好ましい。この１つのマスタランチャは特定のサーバ上でランチャを要求する全てのイベントマネージャに対する要求を扱う。図７に示すようにコンピュータ１００上で実行されるときには、マスタランチャはシステム６００のその他の部分より前又はそれと同時に動作を開始する。この例では、マスタランチャは最初に開始される。

【００７１】

マスタランチャは、例えば、関連するランチャがイベントマネージャ３０１と同じコンピュータ上で実行されているときには、イベントマネージャ３０１へと統合することができる。

【００７２】

２．１．４　ランチャ／第１のアプリケーション

システム６００の１つの有利な実現例では、スマートカード１０のリモートリーダ１への挿入により開始される第１のプロセスはランチャ３０３である。特定のシステムでは特定のアプリケーションが開始されるであろう。例えば、自動現金預払機はバンキングアプリケーションを開始することができる。別の例としては、特定の１組のアプリケーションのみを開始する限定されたランチャの使用が含まれる。ランチャ３０３は、特定のイベントマネージャ３０１に対して他のアプリケーションを開始するアプリケーションである。ランチャ３０３はアプリケーションを開始／終了させることができると共に、各セッションを開始／終了させることができる。また、ランチャ３０３はイベントマネージャ３０１にいつアプリケーションが開始／終了するかを通知し、アプリケーション３０４にいつフォーカスを受け取る／失うか、あるいは、いつ終了する必要があるかを通知する。これに関して、複数のアプリケーション３０４が同時に動作している場合、現在画面に表示されているアプリケーション３０４がフォーカスを有し、「フロントアプリケーション」として知られるアプリケーションである。別のアプリケーションが優先権を得ようとすると、ランチャ３０３はフロントアプリケーションにフォーカスが失われることを通知するので、現在のアプリケーションは当座のタスクを完了することができる。また、ランチャ３０３は新規のアプリケーション３０４にフォーカスが得られ、まもなくアプリケーション３０４が変更状態に入ることを通知する。ランチャ３０３はアプリケーションを強制的に終了するようにも構成される。

【００７３】

ランチャ３０３は、リモートリーダ１が生成する「カードなし」、「バッテリの電力低下」、及び「不良カード」などのイベントを受信しても良い。また、ランチャ３０３は、現在フロントアプリケーションでない各アプリケーション用のイベントも受信し、これらのイベントを正確に解釈するように動作する。

【００７４】
ランチャ３０３は、イベントマネージャ３０１によりその開始を要求する要求が生成されるときにのみ開始されるのが好ましい。また、ランチャ３０３は、イベントマネージャ３０１により終了するように指示したり、強制的に終了したりすることもできる。
【００７５】
ランチャ３０３はディレクトリサービス３１１と通信を行なう必要がある唯一のプロセスコンポーネントであるのが好ましい。新規のアプリケーション３０４を開始する必要があるとき、ランチャ３０３はサービスデータを有するディレクトリサービス３１１に問合せを行なう。ディレクトリサービス３１１は、アプリケーション３０４の位置及び新規のアプリケーション３０４と関連付けられるサービスデータを戻す。サービスデータは、ここでは、ＥＭ＿ＧＡＩＮＩＮＧ＿ＦＯＣＵＳイベントと呼ばれるイベント中の初期化データとして新規のアプリケーション３０４に送信される。アプリケーション位置は実行対象のアプリケーション３０４の位置を指定する。これは、ローカルコンピュータを用いた実現例ではローカルなものであっても良く、あるいは、ネットワーク化されていても良い。アプリケーション位置が空である場合、ランチャ３０３はサービスデータに基づいてどのアプリケーションを開始すべきかを決定しなければならない。
【００７６】
ランチャ３０３は、システム６００の動作中はほぼ常に実行されているであろうブラウザコントローラなどの任意のアプリケーションを開始するように構成することもできる。このようなアプリケーションは永続的アプリケーションと呼ばれる。アプリケーションは、対応するスマートカード１０上での最初のユーザ選択に応答して、あるいは、アプリケーション３０４のうちの別の１つの要求に応じてランチャ３０３により開始することができる。
【００７７】
システム６００の一部の実現例では、ランチャ３０３はイベントマネージャ３０１へと統合することができる。
【００７８】
ランチャ３０３については、第１０．０節を参照して以下で更に詳細に説明する。
【００７９】
２．１．５　ディスプレイマネージャ
ディスプレイマネージャ３０６は、どのスマートカードアプリケーション３０４が現在表示画面１０１上に出力を表示することができるかを選択する。ディスプレイマネージャ３０６には、ランチャ３０３が送信元であるＥＭ＿ＧＡＩＮＩＮＧ＿ＦＯＣＵＳイベントにより表示できるのがどのアプリケーション３０４であるのかが通知される。このイベントはディスプレイマネージャ３０６に直接送信することもできるが、イベントマネージャ３０１がディスプレイマネージャ３０６及び予定される受信者にイベントのコピーを送信することもできる。
【００８０】
一般的に、出力を表示するように試みるべき唯一のアプリケーション３０４はフロントアプリケーションである。ディスプレイマネージャ３０６は、表示の制御権を有する各アプリケーション間での切り替え中に安定した出力を提供することができる。ディスプレイマネージャ３０６は、フロントアプリケーションとしての各アプリケーションの切り替え中には推定のデータを使用する必要があるかもしれない。
【００８１】
アーキテクチャ２００は、ディスプレイマネージャ３０６が必要とされない、あるいは、ディスプレイマネージャ３０６の役割がアーキテクチャ２００の別の部分３０１又は３０３により想定されても良いように構成することができる。
【００８２】
２．１．６　ディレクトリサービス
ディレクトリサービス３１１は、スマートカード１０上に格納されるサービス識別子をサ

ービスの位置又はサービスと関連付けられるアプリケーションの位置を示すリソースロケータ（ＵＲＬなど）へと変換するように構成される。また、ディレクトリサービス３１１はオプションのサービスデータを変換するようにも構成される。ディレクトリサービス３１１は、特定のカード１０と関連付けられたランチャ３０３がリソースロケータを用いて何をすべきか、例えば、関連するアプリケーション３０４をダウンロード・実行するか、あるいは、リソースロケータをブラウザアプリケーションへとロードするかを決定できるようにする。

【００８３】

２．１．７　アプリケーション

特定のスマートカード１０と関連付けられたアプリケーション３０４は、対応するカード上の最初のボタン押下に応じて、スマートカード１０と関連付けられたランチャ３０３により開始することができる。各アプリケーション３０４は１つ以上のサービスグループのメンバであることもできる。このサービスグループについては本明細書で後述する。アプリケーション３０４は、任意のサービスグループの一部とならないように指定することができ、この場合、アプリケーションは他のアプリケーションと共に実行されることはない。アプリケーションは、実行中になればサービスグループの一部となることができ、そのアプリケーションが現在フロントアプリケーションであるときにはサービスグループから外れることができる。

【００８４】

一部のアプリケーションは、システム６００が始動されると開始することができ、これらのアプリケーション、例えば、ブラウザ制御アプリケーション又はメディアプレーイングアプリケーションは常に実行させることができる。これらの永続的アプリケーションはシステム固有であっても、より汎用的に適用可能なものであっても良い。

【００８５】

図９は、上述のプロセスコンポーネント３０１から３０６を含むカードインタフェースシステムの概略ブロック図表現である。図９のシステムにおいて、リモートリーダ１はＩＲリンク及びそれを制御するためのＩ／Ｏデーモン３００を介してコンピュータ１００と通信を行なう。更に、コンピュータ１００は、ここではウェブサーバ４１０に対するインターネット４００により表される通信ネットワークと通信を行なうように構成される。この例において、スマートカード１０及びリモートリーダ１を利用してアクセス可能なアプリケーション３０４の一部は、様々なスマートカード１０と関連付けられるウェブページ４０６であることもできる。ウェブライブラリ４０７は、スマートカード１０と共に使用されるウェブページに含まれる可能性がある関数（例えば、Ｊａｖａｓｃｒｉｐｔ関数）及びクラス（Ｊａｖａクラス）を含む。ウェブページ４０６は、ウェブブラウザ４０３と呼ばれる実行中のアプリケーションを用いてアクセスすることができる。図９のシステムにおいて、イベントマネージャ３０１はリモートリーダ１からイベントを受信するように構成される。イベントはランチャ３０３に送信される。ランチャ３０３はウェブブラウザ４０３を制御するブラウザコントローラ４０２にメッセージを送信するように構成することができる。アプリケーションセッション又はブラウザセッションを開始するプロセスについては以下で詳細に説明する。ランチャ３０３は、ファイルサーバ４１１から実行中のアプリケーションと共にアプリケーション４０８もダウンロードするように構成することができる。ファイルサーバ４１１はインターネット４００を介してコンピュータ１００に接続される。

【００８６】

３．０　リーダ

リモートリーダ１は、スマートカード１０とインタフェースをとってカスタマイズ可能なユーザインタフェースを提供するハンドヘルド・バッテリ駆動のユニットであるのが好ましい。上述のように、リモートリーダ１は、デジタルＴＶ、セットトップボックス、コンピュータ、又はケーブルＴＶ機器と共に使用されることを意図しており、家庭環境におけるオンライン顧客サービスに対する簡単で直感的なインタフェースを提供する。

【００８７】
図４３及び４４は上述のリーダ１と同様のリーダ４４０１を示す。リーダ４４０１はカード１０の読取りのために構成される。リーダ４４０１はカードレセプタクル４４０４及び表示領域４４０６を組み込む筐体４４０２により形成される。カードレセプタクル４４０４は、図１に示されるスマートカード１０を挿入可能なアクセス開口部４４１０を含む。
【００８８】
表示領域４４０６の上側境界は、上述の膜８と同様のほぼ透明な感圧膜４４０８の形態のセンサ手段により定義される。膜４４０８の下方に配置されるのは、スマートカード１０の相補的な接点に接触するように構成される露出した電気接点４４０７の構成の形態で設けられるデータ読取り手段である。
【００８９】
図４５に示すように、カード１０はアクセス開口部４４１０を介してリーダ４４０１へと挿入される。リーダ４４０１の構成により、ユーザはリーダ４４０１を一方の手に保持し、他方の手でスマートカード１０をリーダ４４０１へと簡単に挿入することができる。スマートカード１０がリーダ４４０１に完全に挿入されると、感圧膜４４０８はスマートカード１０の上面１６を完全に覆うようになる。表示領域４４０６は、上面１６が透明な感圧膜４４０８を通して表示領域４４０６内に事実上完全に見えるようにカード１０の上面１６とほぼ同じ寸法を有するのが好ましい。
【００９０】
図４６は、カードが完全に挿入された後にリーダ４４０１を操作するユーザを示す。
【００９１】
図４７（ａ）から図４７（ｃ）において、筐体４４０２は、膜４４０８を包囲する最上部４８２７と、最上部４８２７との接続部４８２９から膜４４０８の横方向の中心の下方にあり且つこれに近接する位置４８１１まで延出するベース部４８０５とにより定義される実質的に二部構成の外部ケーシングから形成される。ベース部４８０５は、赤外線（ＩＲ）透過材から形成される対向端面４８１５を含むので、ＩＲ通信情報をリーダ４４０１により送ることができる。
【００９２】
位置４８１１は、ベース部４８０５とカード支持面４８０７との接続点を定義する。カード支持面４８０７は、接点４４０７が面４８０７と最上部４８２７との間で膜４４０８を挟持する内部接合部４８３５の方に位置する平面を通って延出する。アクセス開口部４４１０は、図４７（ａ）に示すように、位置４８１１と筐体４４０２の周縁部４８３６との間の空間により実質的に定義される。
【００９３】
接点４４０７は、プリント回路基板（ＰＣＢ）４８０１上に設置されるコネクタブロック４８３７から延出する。ＰＣＢ４８０１は、２つのマウンティング４８１７及び４８１９によりベース部４８０５と支持面４８０７との間に配置される。ＰＣＢ４８０１のコネクタブロック４８３７に対向する側には、コネクタ４４０７及びタッチセンシティブ膜４４０８に電気的に接続され、膜４４０８の押下に応じてカード１０からデータを読み取るように構成された電子回路（不図示）が配置される。ＰＣＢ４８０１からは、制御対象の装置（セットトップボックス６０１など）との通信用のＩＲウィンドウとして機能する端部４８１５に隣接して配置される赤外線発光ダイオード（ＬＥＤ）４８００も取り付けられている。
【００９４】
図４７（ｂ）は、スマートカード１０がアクセス開口部４４１０を介してレセプタクル４４０４に部分的に挿入された図４７（ａ）に類似する図を示す。図４７（ｂ）において明らかなように、支持面４８０７は、接点４４０７の平面から接合部４８１１に向かって下方に延出する一体的に形成された曲線状の輪郭４８４０を有する。この構成により、図４７（ｂ）に示すように、スマートカード１０が最初はレセプタクル４４０４の平面に対して傾斜するように、リーダ４４０１はスマートカード１０を受け入れる。支持面４８０７

の曲線状の輪郭部４８４０の構成により、ユーザの手の力を受けるとスマートカード１０は完全な挿入位置にまで導かれる。具体的には、カード１０が更に挿入されると、支持面４８０７の湾曲がカード１０を接点４４０７及びレセプタクル４４０４の平面へと導く。

【００９５】

図４７（ｃ）は、スマートカード１０がレセプタクル４４０４に完全に挿入された図４７（ａ）に類似する図を示す。このとき、カード１０は、レセプタクル４４０４及びスマートカード１０のデータ接点（不図示）のうちの関連する１つに接触する接点４４０７の平面に位置する。スマートカード１０は感圧膜４４０８により覆われる。更に、接点４４０７はカード１０に対向する力を提供するように動作するばね接点であって、膜４４０８と関連付けられるのが好ましい。この構成は、適切な衝突収納(neat interfernce fit)によりカード１０をレセプタクル内に保持するのに十分である。

【００９６】

以下の説明では、リーダ１への言及は、図１のリーダ１又は図４３から図４７（ｃ）のリーダ４４０１として実現されるリーダへの言及として解釈される。

【００９７】

図１０は、リモートリーダ１の内部構成を更に詳細に示す概略ブロック図である。リモートリーダ１は、リモートリーダ１を制御し、リモートリーダ１と、例えば、セットトップボックス６０１との間の通信を調整し、マッピング情報を格納するためのマイクロコントローラ４４を含む。マイクロコントローラ４４は、ランダムアクセスメモリ（ＲＡＭ）４７及びフラッシュ（ＲＯＭ）メモリ４６を含む。また、マイクロコントローラ４４は中央処理装置（ＣＰＵ）４５を含む。マイクロコントローラ４４はクロックソース４８とマイクロコントローラ４４内のイベントのタイミングを調整するためのクロックコントローラ４３とに接続される。ＣＰＵ４５には５Ｖバッテリ５３からの電力が供給され、ＣＰＵ４５の動作は電源制御装置５０により制御される。マイクロコントローラ４４は、カードエントリ状態について及び「ボタン」押下に対して可聴のフィードバックを与えるためのビーパ５１にも接続される。

【００９８】

赤外線（ＩＲ）通信は、マイクロコントローラ４４に接続される２つの回路、すなわち、ＩＲ送信用のＩＲ送信機（送信機）４９及びＩＲ受信用のＩＲ受信機（受信機）４０を使用して実現されるのが好ましい。

【００９９】

リモートリーダ１の感圧タッチパネル８は、タッチパネルインタフェース４１を介してマイクロコントローラ４４と通信を行なう。スマートカードインタフェース４２は電気接点７に接続する。

【０１００】

システム内プログラミングインタフェース５２もマイクロコントローラ４４に接続される。これにより、マイクロコントローラフラッシュメモリ４６を介してファームウェアを用いてマイクロコントローラ４４をプログラミングできるようになる。ファームウェアについては、第６．０節を参照して本文書で詳細に後述する。

【０１０１】

ここでは、リモートリーダ１の内部構成を更に詳細に説明する。

【０１０２】

３．１　低電力モードの寿命

電源制御装置５０は、低電力「スリープ」モード及びアクティブモードの２つの電力モードを提供するように動作可能である。低電力モードの寿命は、バッテリ５３の寿命を年単位で表したものである。リモートリーダ１が機能しておらず、低電力モードにあるとき、寿命は２年間よりも長くなる可能性がある。

【０１０３】

リーダ１がスリープモードにあるときにユーザがタッチパネル８を押下する場合、リモートリーダ１はスリープモードから抜け出て、ＣＰＵ４５は接触座標を計算すると共に赤外

線送信によりシリアルメッセージを送信する。バッテリ５３は１００，０００回を超えるボタン押下の電流供給要求に対して使用可能状態を維持するのが好ましい。

【０１０４】

３．２　耐用寿命

耐用寿命は、リモートリーダ１が使用可能状態を維持することが予想される期間として定義され、バッテリの交換は含まない。耐用寿命は平均故障間隔（ＭＴＢＦ）の数値に関連し、通常、加速寿命試験を使用することで統計的に得られる。従って、リモートリーダ１の耐用寿命は５年よりも長くなる可能性がある。

【０１０５】

３．３　マイクロコントローラ

リモートリーダ１のマイクロコントローラ４４は、４０９６バイトのフラッシュメモリ４６及び１２８バイトのランダムアクセスメモリ４７と共に８ビット中央ＣＰＵを有する。マイクロコントローラ４４は３から５ボルトの供給電力で動作するのが好ましく、フレキシブルオンボードタイマ、割込み源、８ビットＡ／Ｄコンバータ（ＡＤＣ）、クロックウオッチドッグ、及び低電圧リセット回路を有する。また、マイクロコントローラ４４は大電流出力ピンを有し、わずかな外部接続のみで回路中にプログラムできるのが好ましい。

【０１０６】

３．４　クロックソース

リモートリーダ１用のメインクロックソース４８は、内蔵の平衡コンデンサを有する３ピン４．９１ＭＨｚセラミック共振子であるのが好ましい。周波数許容範囲は０．３％である。この許容範囲は水晶ほど良くはないが、シリアル通信に適しており、水晶よりも小型で安価である。

【０１０７】

３．５　ビーパ（Beeper)

ビーパ５１はリモートリーダ１の中に含まれており、カードエントリ状態について及びボタン押下に対して可聴のフィードバックを与える。ビーパ５１は圧電セラミックディスク型であるのが好ましい。

【０１０８】

３．６　赤外線通信

上述のように、赤外線（ＩＲ）通信は、ＩＲ送信用のＩＲ送信機４９及びＩＲ受信用のＩＲ受信機４０の２つの回路を使用して実現されるのが好ましい。この２つの回路４０及び４９は、リモートリーダ１内のプリント回路基板（例えば、図４７のＰＣＢ４８０１）上で組み合わされるのが好ましい。プリント回路基板４８０１は、４方向フラットプリントケーブルによりマイクロコントローラ４４に接続することができる。送信時に必要なサージ電流を提供するために、ＰＣＢ４８０１には大型のバルク減結合コンデンサが必要とされる。

【０１０９】

３．７．１　赤外線送信

ＩＲ送信は、ＩＲ送信機４９の一部を形成する赤外線発光ダイオード（ＬＥＤ）（例えば、図４７（ａ）のＬＥＤ４８００）により行なわれるのが好ましい。

【０１１０】

３．７．２　赤外線受信

ＩＲ受信機４０は、赤外線フィルタ、ＰＩＮダイオード、増幅器、及び弁別回路と共に単一のデバイスへと統合されるのが好ましい。受信されるシリアル情報は、このデバイスからマイクロコントローラ４４の入力ポートへと直接進む。このポートは、データの受信時に割込みを発生し、迅速な格納及び入力信号の処理が可能になるようにプログラムすることができる。

【０１１１】

３．８　ＣＰＵ／メモリカードインタフェース

リモートリーダ１は、国際標準化機構（ＩＳＯ）標準７８１６－３及びＩＳＯ７８１０に

より定義されるスマートカード１０をサポートできるのが好ましい。３Ｖメモリカード及び５Ｖメモリカードと同様に、Ｔ＝０プロトコル及びＴ＝１プロトコルを有する３ＶＣＰＵカード及び５ＶＣＰＵカード（すなわち、内蔵のマイクロプロセッサを有するカード）もサポート可能である。

【０１１２】

カード１０とマイクロコントローラ４４とを接触させるのに使用される電気接点７は、８つのワイピング接点及び１つの「カードイン」スイッチを有する表面実装型コネクタであるのが好ましい。ＩＳＯの基準によると、以下の信号が提供されなければならない：

・ピン１－ＶＣＣ－供給電圧、
・ピン２－ＲＳＴ－リセット信号、カードへの２進出力、
・ピン３－ＣＬＫ－クロック信号、カードへの２進出力、
・ピン４－ＲＦＵ－未使用、常時未接続、
・ピン５－ＧＮＤ－グランド、
・ピン６－ＶＰＰ－プログラミング電圧、不要、ＧＮＤ、ＶＣＣへの連結又は開放、
・ピン７－Ｉ／Ｏ－データＩ／Ｏ、双方向信号、及び
・ピン８－ＲＦＵ－未使用、常時未接続

ＲＳＴピン及びＩ／Ｏピンはマイクロコントローラ４４に直接接続されるのが好ましい。電源を除く全てのピンには、静電気放電の問題を回避するために、シリーズターミネーションダイオード及び過度電圧抑制器ダイオードが備わっている。

【０１１３】

３．９　ＣＰＵカード電源

上述のように、マイクロコントローラ４４が動作するには３Ｖ～５Ｖの電源が必要である。５Ｖ電源は、調整５ＶチャージポンプＤＣ－ＤＣコンバータチップの形態の電源制御装置５０を用いてバッテリ５３として動作する３Ｖコイン型リチウム電池から生成することができる。

【０１１４】

３．１０　タッチセンシティブインタフェース

上述のように、リモートリーダ１の感圧タッチパネル８は、タッチパネルインタフェース４１を介してマイクロコントローラ４４と通信を行なう。タッチパネルインタフェース４１はタッチパネル８上の接触の位置に従ってアナログ信号を提供する。このアナログ信号はマイクロコントローラ４４に送信される。

【０１１５】

接触座標の計算には、下側及び左側のタッチパネル８の接点（不図示）がマイクロコントローラ４４上のＡ／Ｄコンバータの入力へと接続されることが必要である。

【０１１６】

タッチパネル８への接触は、好ましくは、リモートリーダ１をスリープモードからウェイクアップするのに使用することができる。左側の画面接点から図示されるスリープ中のウェイクアップポートまで抵抗接続することでこの機能が提供される。尚、システム内プログラミング中に最大８ボルトが割込み要求ピン（ＩＲＱ）と呼ばれるマイクロコントローラ４４上のピンに印加される可能性があるので、デバイスの損傷を防止するためにクランピングダイオードをこのピンに取り付ける必要がある。この例では、タッチパネル８の押下を検知するのに必要なバイアスを実際に提供するのはＩＲＱピン上の内部プルアップである。

【０１１７】

３．１１　バッテリ

上述のように、リモートリーダ１はバッテリ５３を使用する。５Ｖコイン型リチウム電池をバッテリ５３として使用し、リモートリーダ１の全回路に電力を供給することができる。

【０１１８】

３．１２　システム内プログラミング

マイクロコントローラはシステム内プログラミング（ＩＳＰ）オプションをサポートする。リモートリーダ１においては、システム内プログラミングインタフェース５２が使用され、ファームウェアを用いたマイクロコントローラフラッシュＲＯＭメモリ４６のプログラミングなどのマイクロコントローラ４４のプログラミングが実行される。

【０１１９】

３．１３　プリント回路基板及び相互接続

リモートリーダ１は、リーダ４４０１の１つのＰＣＢ４８０１の代りに、以下に示すような２つのプリント回路基板（ＰＣＢ）を含むことができる：

（ｉ）赤外線ダイオードを保持し、ＦＥＴ及び受信機を駆動する赤外線（ＩＲ）ＰＣＢ、及び

（ii）上述の他の全ての構成要素４０から５３を保持するメインＰＣＢ（例えば、図４７（ａ）のＰＣＢ４８０１）

上述の双方のＰＣＢボードは、標準のＦＲ４、１．６ｍｍＰＣＢ材料を使用する両面設計であるのが好ましい。完成品のＰＣＢの厚さは重要であるので、メインＰＣＢは表面実装型部品を利用するのが好ましく、部品は最大約３ｍｍの高さに制限されるのが好ましい。

【０１２０】

ＩＲ　ＰＣＢはスルーホール部分を使用することができるが、部品の高さには厳しい制限が課せられるのが好ましい。２つのＰＣＢの相互接続は、カスタムデザイン４方向プリントケーブル（ＦＣＡ）を介して行なわれる。これは、タッチパネル８とインタフェースをとるのに使用される同一部品の表面実装型ＦＣＡコネクタを介して２つのＰＣＢとインタフェースをとる。

【０１２１】

３．１４　低電力モード

リモートリーダ１が短時間の間使用されない場合、バッテリの寿命を浪費しないために事前にプログラムされたファームウェアが装置を低電力モードにするのが好ましい。低電力モードでは、全ての電流を消費する構成要素への供給電力が停止され、マイクロコントローラ４４の各ポートは安全なスリープ状態に設定され、クロック４８は停止される。この状態で、リモートリーダ１の電流消費は５μＡ未満である。ＰチャネルＦＥＴを使用して電流を消費する構成要素への電力供給を制御することができる。

【０１２２】

低電力モードからリモートリーダ１をウェイクアップするには以下に示すような３つの代替の好適な方法がある：

・タッチパネル８への接触、
・カードレセプタクル４へのカードの挿入、及び
・バッテリ５３の取外し及び再挿入

カード挿入ウェイクアップにより、装置が低電力モードにあるか否かに関わらず、リモートリーダ１はカードが挿入されるときには常にビープ音を発することができる。接触／カード挿入ウェイクアップは、マイクロコントローラ４４上に図示されるようにＩＲＱピンにより処理される。新規の接触又はカード挿入のみがマイクロコントローラをウェイクアップするように、このピンは「エッジトリガ」に設定されることが重要である。ＩＲＱの感度が「レベル」トリガに設定される場合、例えば、リモートリーダ１が荷物の中に詰められているときにタッチパネル８が偶然押下されたままの状態になると、リモートリーダ１は低電力モードに入ることができない。

【０１２３】

３．１５　割込み及びリセット

リモートリーダ１用のマイクロコントローラ４４のファームウェアは、２つの外部割込み原因及び１つの内部割込み原因を使用する。低電力モードのウェイクアップの場合、外部割込みはＩＲＱピンから発生する。内部割込みはタイマのオーバーフローにより引き起こされ、種々の外部インタフェースの時間を調整するのに使用される。これらの割込みは事前にプログラムされたファームウェア手順により行なわれる。

【０１２４】
マイクロコントローラには以下に示すように４つの可能なリセット原因が存在する：
・２．４Ｖでの低供給電力リセット、
・不正なファームウェア演算コードリセット、
・ファームウェアがループに入り込んだ場合のコンピュータ正常動作（ＣＯＰ）リセット、及び
・システム内プログラミング（ＩＳＰ）が開始されるときにリセットピンに対して強制されるＩＳＰリセット
【０１２５】
４．０　カードデータ形式
上述のカード１０に対するデータ形式を以下の段落で説明する。図４に関連して説明した制御カード１０Ｂのようなメモリカードの場合、後述する形式に従ったデータはカードに直接コピーされることになる。上述のＣＰＵカードの場合、後述する形式に従ったデータは、ファイルとしてカードのＣＰＵのファイルシステムへとロードすることができる。
【０１２６】
上述のカード１０は、種々のカード特性及びカード上に印刷された任意のユーザインタフェース印を記述するデータ構造を格納するのが好ましい。また、カード１０はカード、ベンダ、及びサービスについての情報などの属性を指定するグローバル特性を含むことができる。ユーザインタフェースオブジェクトが存在する場合には、カード１０の表面の領域と関連付けるべきデータを指定する。
【０１２７】
ここで説明するユーザインタフェースは、所定の領域又はカード１０の表面に直接転写されたアイコン表現をコマンド又はアドレス（例えば、Uniform Resource Locator（ＵＲＬ））と関連付けるマッピングデータを表す。マッピングデータは、一般的に、カード１０上のユーザインタフェース要素（例えば、所定の領域）のサイズ及び位置を定義する座標を含む。ここで、ユーザインタフェース要素という用語は、通常、カード１０上の印を指すが、ユーザインタフェースオブジェクトという用語は、通常、特定の印に関連するデータを指す。しかし、これらの用語は以下の説明において区別なく使用される。
【０１２８】
ユーザインタフェースオブジェクトはカード１０に直接格納されるのが好ましい。また、ユーザインタフェースオブジェクトはカード１０自体ではなく、システム６００に格納することもできる。例えば、カード１０はオンカードメモリを介して固有の識別子であるバーコード又は磁気ストリップを格納することができる。この固有の識別子は、ほぼ同様のユーザインタフェース要素及びレイアウトを有するカード１０に固有のものである。固有の識別子は、ユーザの押下の結果タッチパネル８から判定された座標と共にリーダ１によりシステム６００のコンピュータ１００又はセットトップボックス６０１に送信することができる。コンピュータ１００、セットトップボックス６０１、又はサーバ１５０上に格納されたユーザインタフェースオブジェクトを有するシステム６００は、カード１０上のユーザインタフェース要素により表される所望の機能を提供するために、判定された座標からカード１０及びユーザ押下と関連付けられたサービスに関連する対応するコマンド、アドレス、又はデータへのマッピングを実行することができる。この例では、上述のユーザにより選択された印に関連するデータは、所望の印に重なるタッチパネル８の部分をユーザが押下した結果リーダ１により判定される座標の形態をとる。
【０１２９】
上述のカード（例えば、１０）において、カード１０により格納されるデータはカードヘッダを含み、このカードヘッダには以下の各節で説明する０個以上のオブジェクトが続く。
【０１３０】
４．１　カードヘッダ
図１１は、スマートカード１０に格納されたカードヘッダ１１００のデータ構造を示す。

ヘッダ１１００は、各々が４バイトのデータを表す複数の列１１０１を含む。データは「ビッグエンディアン」形式であるのが好ましい。完全なヘッダは２０バイト長であり、以下のフィールドを含む（詳細は図１２に記載）：
（ｉ）カードを有効なメモリカードとして指定する定数を含むマジックナンバーフィールド１１０２（例えば、このマジックナンバーフィールド１１０２は、特定の製品に帰属するプロプライエタリカードが使用されていることを確認又は検証するのに使用することができる）、
（ii）カードレイアウトの下位のバージョンと互換性のあるリーダにより読むことができないレイアウトの変更を指定する各バージョンの増加分を含むバージョンフィールド１１０３、
（iii）予約フィールド１１０４（このフィールドは今後の使用のために予約されている）、
（iv）カード用のフラグを含むフラグフィールド１１０５（図１３参照）、
（v）サービス１１０６及びサービス固有１１０７フィールドの２つのフィールドを含む区別用識別子フィールド１１１０（サービスフィールド１１０６は対応するカードのサービスを識別し、サービス固有フィールド１１０７はオプションとしてサービス固有値を含む）、
（vi）ヘッダに続くオブジェクトの数を表す数値を含むオブジェクト数フィールド１１０８（このフィールドは０に設定することもできる）、及び
（vii）チェックサム自体を除くカード上の全てのデータのカードチェックサムを含むチェックサムフィールド１１０９
【０１３１】
図１２は、図１１を参照して説明した種々の（数値）フィールドの内容の記述を提供する。特に、区別用ＩＤ番号フィールド１１１０は８バイトの区別用識別子を具備する。区別用識別子は、データの単位である２つの部分、すなわち、サービス識別子及びサービス固有識別子を含む。区別用識別子は、サービス識別子が区別用識別子の値全体のうちの５バイトを占め、サービス固有識別子が３バイトを占めるように構成されるのが好ましい。
【０１３２】
フィールド１１０６に含まれるサービス識別子は、サービス又はベンダを区別する。すなわち、例えば、サービスはカードユーザにそのサービスを提供するアプリケーションと関連付けることができる。これは、複数のアプリケーションを提供することによってカードユーザに複数のサービスを提供することができるベンダとは異なる。
【０１３３】
サービス識別子は使用するアプリケーション又はアプリケーションの位置（例えば、ＵＲＬ）を識別するための識別子となることができる。また、総称カードがシステム６００Ａ又は６００Ｂに追加されても良く、これらのカードはサービス識別子の特殊な使用例である。総称カードは、既に実行中の現在のアプリケーションへの入力を提供するのに使用可能な特殊なサービス識別子を有するカードである。サービスの特殊値０ｘ００００００００００１は「総称サービス識別子」として知られ、「総称カード」上で使用される。総称カードは既に実行中のフロントアプリケーションにデータを送信するのに使用することができる。これらのカードは、例えば、テキスト入力を任意のアプリケーションに送信するのに使用可能なキーパッド又は個人的な詳細情報を任意のアプリケーションに送信するのに使用されても良い個人情報付きのカードの代わりに使用される。
【０１３４】
フィールド１１０７に含まれるサービス固有識別子は、特定のサービスのベンダが、その特定サービスと関連付けられる所定の機能を提供するのに任意で使用することができる。サービス固有識別子の使用は、システム６００上で実行中のアプリケーション３０４によってほぼ決まる。例えば、サービス識別子及びサービス固有識別子は、カード１０に対する固有の識別子として使用することができる。この固有の識別子は、特定のサービスと関連付けられる特定の機能へのアクセスを獲得又は拒絶し、ログファイル中に固有サービス

識別子を再現してその値を有する特定のカード１０がサービスにアクセスするのに使用されたことを確認又は検証できるようにし、データベース中の対応する値に合わせることができる固有の識別子を提供してサービスのユーザに関する情報（名前、住所、クレジットカード番号など）を取得するために使用することができる。

【０１３５】

サービス固有識別子の使用の他の例として、カード１０の配布のメカニズム又は様式（郵便、バスターミナルの売店、列車内での配布など）に関する情報の提供を含むことができる。更に、サービス固有識別子は、サービスがアクセスされたときにシステム６００にどのデータがロードされるべきかを識別することができる。

【０１３６】

以上の説明はサービス固有識別子の可能な使用法又は適用例を網羅したリストではなく、可能な適用例の少数のサンプルに過ぎない。フィールド１１０７のサービス固有識別子にはその他にも多くの適用例がある。

【０１３７】

４．１．１　カードフラグ

図１１のヘッダ１１００のフラグフィールド１１０５は以下のような３つのフラグを含んでも良い：

（i）ビープ停止、

（ii）移動イベントなし、及び

（iii）イベント座標なし

図１３は上述の各フラグの記述を示す。上述のフラグは、各フラグの記述により定義されるように、リモートリーダ１においてスマートカード１０が実行可能な機能に影響を及ぼす。図１３で参照されたユーザインタフェース要素の一例はカード１０上の「ボタン」である。ユーザインタフェース要素は本文書で更に詳細に後述する。

【０１３８】

４．２　オブジェクト

図５７に示すように、図１１のカードヘッダ１１００の直後に、特定のカード１０の各オブジェクトを定義し且つカード１０上に格納されたデータの一部を形成する０個以上のオブジェクト構造５７１３を続けることができる。各オブジェクト構造５７１３は以下のような４つのフィールドを具備する：

（i）種別フィールド５７０１、

（ii）オブジェクトフラグフィールド５７０３、

（iii）長さフィールド５７０５、及び

（iv）データフィールド５７０７

データフィールド５７０７の構造は後述するオブジェクト種別によって決まる。

【０１３９】

図１４は、オブジェクト構造５７１３の各フィールド５７０１、５７０３、５７０５、及び５７０７の記述を示す。オブジェクト構造５７１３のフラグフィールド５７０３は、非アクティブフラグを含むのが好ましい。図１５は非アクティブフラグの記述を示す。

【０１４０】

上述のカード１０Ａ、１０Ｂ、１０Ｃ、及び１０Ｄには以下に示すように５つのオブジェクト種別が提供されるのが好ましい：

（i）ユーザインタフェースオブジェクト（すなわち、カード１０上のボタンを定義するデータ）、

（ii）カードデータ、

（iii）固定長データ、

（iv）リーダ挿入、

（v）操作なし、及び

（vi）操作なし（１バイト）

図１６は、上述のオブジェクト種別（i）から（vi）の各々の記述を示している。

【０１４１】
４．２．１　ユーザインタフェースオブジェクト
各ユーザインタフェースオブジェクトは、カード１０上の矩形領域とユーザがパネル８のカード１０の対応する矩形領域の上にある領域に接触するときに送信される幾つかの関連データとを定義する。座標マッピングシステムの起点は、スマートカード１０がＩＳＯ標準メモリスマートカードであり、チップ接点１８がビューアの反対方向且つカード１０の底部の方向を向くように縦向きに保持されたときのカード１０の左上である。このカードの向きを使用しないリーダ１の場合、正しい「ボタン」押下を通知するように四隅の点の値がリーダ１によって調整されなければならない。
【０１４２】
ユーザインタフェース（要素）オブジェクト構造は、以下に示すように６つのフィールドを有するのが好ましい：
（ｉ）フラグフィールド、
（ｉｉ）Ｘ１フィールド、
（ｉｉｉ）Ｙ１フィールド、
（ｉｖ）Ｘ２フィールド、
（ｖ）Ｙ２フィールド、及び
（ｖｉ）例えば、ＵＲＬ、コマンド、キャラクタ又は名前に対するユーザインタフェース要素と関連付けられるデータを通常含むデータフィールド
【０１４３】
図１７は、先に説明したユーザインタフェースオブジェクト構造に対する上述の各フィールドの記述を示す。感圧タッチパネル８の押下は、以下のような場合に特定のユーザインタフェースオブジェクトの内側であると定義される：
（ｉ）押下位置のＸ値が関連するユーザインタフェースオブジェクトのＸ１値以上であり、その特定のユーザインタフェースオブジェクトのＸ２値未満である場合、及び
（ｉｉ）押下位置の押下Ｙ値が特定のユーザインタフェース要素のＹ１値以上であり、Ｙ２値未満である場合
【０１４４】
ユーザインタフェース要素は重複しても良い。押下が２つ以上のユーザインタフェース要素の範囲内である場合、送信されるオブジェクトはＺ順により判定される。カード上のユーザインタフェース要素の順序は、その特定のカード上の全てのユーザインタフェース要素に対するＺ配列を定義する。先頭のユーザインタフェース要素は特定のカード１０に対する第１のユーザインタフェース要素である。最後尾のユーザインタフェース要素はそのカード１０に対する最終のユーザインタフェース要素である。これにより、非矩形の領域の定義も考慮される。例えば、「Ｌ」型のユーザインタフェース要素を定義するには、第１のユーザインタフェースオブジェクトがデータフィールドに０バイトを伴って定義され、第２のユーザインタフェースオブジェクトが第１のユーザインタフェースオブジェクトの左下にユーザインタフェースオブジェクトに重なるように定義されるであろう。
【０１４５】
押下の位置は「フィンゲル（fingel）」において通知される。この「フィンゲル」はフィンガー要素（画素を表す「ピクセル」に類似）を表す。フィンゲルの高さはＩＳＯメモリスマートカードの長さの１／２５６であり、幅はＩＳＯメモリスマートカードの幅の１／１２８であると定義される。各要素と関連付けられる挙動は１つ以上のフラグを用いて変更されても良い。各ユーザインタフェース要素は以下に示すように４つの関連するフラグを有するのが好ましい：
（ｉ）ビープイネーブル反転
（ｉｉ）オートリピート
（ｉｉｉ）押下時データ送信禁止
（ｉｖ）リリース時データ送信禁止
図１８は、各ユーザインタフェース要素フラグの記述を示す。

【０１４６】
４．２．２　カードデータ
カードデータオブジェクトは、特定のカードに固有のデータを格納するのに使用される。このオブジェクトに対するデータレイアウトは固定の形式をもたない。カード１０がリーダ１へと挿入されるとき、カードデータオブジェクトの内容がＩＮＳＥＲＴメッセージの一部としてリーダ１から送信される。
【０１４７】
４．２．３　固定長データ
固定長データオブジェクトは、例えば、コンピュータ１００により書き込むことが可能なカード上の固定長ブロックを定義するのに使用される。
【０１４８】
４．２．４　リーダ挿入
リーダ挿入オブジェクトは、特定のカードが挿入されたときにリモートリーダ１に対する命令を格納するのに使用される。これは、例えば、ＩＲコマンドの特定の構成を使用して特定のセットトップボックス又はＴＶなどと送受信できるようにするべくリーダ１に指示するのに使用することができる。
【０１４９】
４．２．５　操作なし
操作なしオブジェクトは、特定のカード上の他のオブジェクト間の未使用のセクションを埋めるのに使用される。操作なしオブジェクトに格納されるあらゆるデータはリモートリーダ１によって無視される。カード１０の端の未使用の空間は操作なしオブジェクトを用いて埋める必要がない。
【０１５０】
４．２．６　操作なし（１バイト）
操作なし（１バイト）オブジェクトは、完全なオブジェクト構造には小さすぎるオブジェクト間の隙間を埋めるのに使用される。これらのオブジェクトは合計しても１バイト長である。
【０１５１】
５．０　リーダプロトコル
リモートリーダ１は、例えば、リモートリーダ１とセットトップボックス６０１又はコンピュータ１００との間の単方向通信及び双方向通信の双方をサポートするデータグラムプロトコルを使用する。リモートリーダ１とユーザとのインタラクションの結果、リモートリーダ１から受信するメッセージに使用される形式は、リモートリーダ１に送信される形式とは異なる形式である。
【０１５２】
５．１　メッセージ種別
リモートリーダ１により送信することが可能なメッセージイベント種別は少なくとも７つある。これらのイベントは以下の通りである：
・ＩＮＳＥＲＴ：カード１０がリモートリーダ１へと挿入され、カード１０が検証されると、リモートリーダ１によりＩＮＳＥＲＴイベントが生成され、関連するメッセージが送信される。このメッセージはカード１０の存在を受信機（例えば、セットトップボックス６０１）に通知するものである。ＩＮＳＥＲＴメッセージは特定の区別用識別子を含むのが好ましく、最初のインタラクションが起こるまで待たずにカード１０の挿入直後にアプリケーションを開始又はフェッチできるようにする。ＩＮＳＥＲＴメッセージは、この種のオブジェクトがカード１０上に存在する場合、リーダ１へと挿入されるカード１０からのカードデータオブジェクトの内容を含むのが好ましい。
【０１５３】
・ＲＥＭＯＶＥ：カード１０がリモートリーダ１から抜き取られると、対応するＲＥＭＯＶＥイベントが生成され、ＲＥＭＯＶＥメッセージがリモートリーダ１と関連付けられた特定の受信機に送信される。ＩＮＳＥＲＴメッセージと同様に、関連する区別用識別子が

メッセージと共に送信される。抜き取られたカード１０からは区別用識別子を読むことができないので、区別用識別子はリモートリーダ１のメモリ４７に格納される。他の全てのメッセージが区別用識別子を必要とするが、区別用識別子が必要となるたびにカード１０から区別用識別子を読むのでは遅すぎる可能性があるので、これは有用な最適化である。システム６００は処理を制御する際にＩＮＳＥＲＴメッセージ及びＲＥＭＯＶＥメッセージに依存しない。受信したメッセージが予期されるものでない場合、システム６００は欠落したメッセージがあることを推定するように構成される。例えば、アプリケーションが２つの連続するＩＮＳＥＲＴメッセージを検知した場合、現在の構成では一度に２枚のカードを挿入することはできないので、第１のＩＮＳＥＲＴメッセージのカードと関連付けられたＲＥＭＯＶＥメッセージが欠落したと想定することができる。アプリケーションは、第２のＩＮＳＥＲＴメッセージを処理する前に必要なあらゆる動作を行なうことができる。

【０１５４】

・メッセージの欠落が起こりうる別の例は、有線式のリーダに対してハンドヘルドの赤外線接続のリーダ１が使用される場合である。ユーザはカードの挿入又は抜き取り時にリーダを受信機に直接向けないことが多い。この問題は、連続するＰＲＥＳＳ／ＲＥＬＥＡＳＥの対において区別用識別子が異なることに基づいてＩＮＳＥＲＴ操作又はＲＥＭＯＶＥ操作を推定するシステム６００により解消することができる。

【０１５５】

・ＢＡＤ ＣＡＲＤ：無効なカードが挿入される場合、リモートリーダ１はＢＡＤ ＣＡＲＤイベントを生成し、ＢＡＤ ＣＡＲＤメッセージを送信するように構成されるのが好ましい。このメッセージにより、関連する受信機はユーザに対して無効なカードであることを警告すべく何らかの動作を行なうことができる。

【０１５６】

・ＰＲＥＳＳ：リモートリーダ１により接触が検知されるとき、ＰＲＥＳＳイベントが生成され、ＰＲＥＳＳメッセージが関連する受信機に送信される。ＰＲＥＳＳメッセージは、関連するカードの詳細、押下の位置、及びその特定の位置においてユーザインタフェース要素と関連付けられたデータを含む。その位置に対して定義されるユーザインタフェース要素が存在しない場合（カード１０上に定義されるユーザインタフェース要素が全く存在しない場合を含む）、関連するカードの詳細及び押下の位置を含むＰＲＥＳＳメッセージが送信される。ＰＲＥＳＳイベントが生成されたときにリモートリーダ１内にカードが存在しない場合、特殊な「ＮＯ＿ＣＡＲＤ」識別子（すなわち、８バイトの０－０ｘ００）及び押下の位置を含むＰＲＥＳＳメッセージが送信される。

【０１５７】

・ＲＥＬＥＡＳＥ：ＲＥＬＥＡＳＥイベントはＰＲＥＳＳイベントを補うものであり、システム６００のアプリケーションプログラムにＰＲＥＳＳが解除されたことを通知するためにＲＥＬＥＡＳＥメッセージを送信することができる。全てのＰＲＥＳＳイベントは対応するＲＥＬＥＡＳＥイベントを有するのが好ましい。リーダは複数回の押下を登録したり、あるいは、ＰＲＥＳＳメッセージとＲＥＬＥＡＳＥメッセージとの間に発生する可能性があるその他のイベントを提供したりすることができる。

【０１５８】

・ＭＯＶＥ：ＰＲＥＳＳイベントの処理後に接触位置がある量だけ変化する場合、指（又はカードに接触するのに使用されるもの）が移動中であると想定される。ＭＯＶＥ ＥＶＥＮＴが生成され、接触が解除されるまでＭＯＶＥメッセージが送信される。接触位置が静止したままであるときには、ＭＯＶＥイベントは直前のＭＯＶＥメッセージを再送信することによってオートリピートする。接触が解除され、対応するＲＥＬＥＡＳＥメッセージが送信されると、送信の繰り返しは終了する。ＰＲＥＳＳイベント及びＲＥＬＥＡＳＥイベントとは異なり、ＭＯＶＥイベントと関係するユーザインタフェースオブジェクトは存在しない。

【０１５９】

・ＬＯＷ ＢＡＴＴ：リモートリーダ１中のバッテリ５３の電力が低下しているとき、ＬＯＷ ＢＡＴＴイベントが生成され、ＬＯＷ ＢＡＴＴメッセージが送信される。このメッセージは、ユーザとのインタラクションの後にメッセージがシステム６００のその他の部分により受信される可能性を高めるために送信される。ＬＯＷ ＢＡＴＴメッセージの送信は、リモートリーダ１が低電力状態に入ることを妨害しない。

【０１６０】

５．２　データ形式

システム６００において使用されるリーダプロトコルの好適なデータ形式は、固定サイズのヘッダの後に０バイト以上の長さを有することが可能な可変長のデータフィールドが続き、更にその後に８ビットのチェックサム及び補数が続く形式である。

【０１６１】

５．２．１　メッセージヘッダ

メッセージヘッダは固定長であることが好ましく、リモートリーダ１から送信される全てのメッセージの前に付加される。帯域幅に制約がある可能性があるので、メッセージヘッダは可能な限り小さくする必要がある。図１９は、リモートリーダ１から送信されるメッセージヘッダの形式を示す。

【０１６２】

ベンダが特定のサービスを登録するとき、スマートカード識別機関によりサービス識別子及びサービス固有識別子をそのベンダに割り当てることができる。サービス及びサービス固有識別子は任意のカードからの全てのメッセージに対して同じである。サービス固有識別子はアプリケーションと共に使用するためにベンダにより設定されるのが好ましい。リーダ識別子も各メッセージのヘッダ中に存在する。この識別子は、例えば、マルチプレーヤゲームにおいてユーザを区別するためにアプリケーション３０４により使用することができる。

【０１６３】

図２０は、以上説明したメッセージイベント種別を列挙するテーブルを示している。

【０１６４】

５．２．２　単純なメッセージ

幾つかのメッセージ種別は、上述のメッセージヘッダ並びにそれに続くメッセージチェックサムバイト及びその補数のみから成るという点で単純であると考えられる。例えば、ＢＡＤＣＡＲＤメッセージ、ＬＯＷ＿ＢＡＴＴメッセージ、及びＲＥＭＯＶＥメッセージは単純なメッセージである。

【０１６５】

図２１は、単純なメッセージの形式を示す。

【０１６６】

５．２．３　ＭＯＶＥメッセージ

ＭＯＶＥメッセージは、上述のメッセージヘッダ及びそれに続くリモートリーダ１のタッチパネル８上の接触位置の座標を定義する２つのフィールドから形成される。図２２はＭＯＶＥメッセージの形式を示す。

【０１６７】

５．２．４　ＰＲＥＳＳメッセージ及びＲＥＬＥＡＳＥメッセージ

図２３は、ＰＲＥＳＳメッセージ及びＲＥＬＥＡＳＥメッセージの形式を示す。ＰＲＥＳＳメッセージ及びＲＥＬＥＡＳＥメッセージは、ＭＯＶＥメッセージと同様にメッセージヘッダ及び接触座標を含む。加えて、接触位置がカード上で定義されるユーザインタフェース要素と一致する場合、ＰＲＥＳＳメッセージ及びＲＥＬＥＡＳＥメッセージはユーザインタフェース要素と関連付けられるデータを送信する。このデータは可変長であり、実際のサイズは対応するカード１０により定義される。接触位置がカード上で定義されるユーザインタフェース要素と一致しない場合（カード上で定義されるユーザインタフェース要素がない場合を含む）、ユーザインタフェース要素と関連付けられる０バイトのデータが送信される。リーダ１内にカード１０がない場合、サービス識別子は全てが０（すなわ

ち、０ｘ００）に設定され、ユーザインタフェース要素と関連付けられる０バイトのデータが送信される。ユーザインタフェース要素と関連付けられたデータは、通常、カード上で定義されるユーザインタフェース要素と関連付けられたデータに対応するが、カード１０又はリーダ１での処理により変更又は生成されても良い。

【０１６８】

図２４は、システム６００内の上述のメッセージの流れを示すデータフロー図である。図２４に示すように、カードヘッダ１１００及びオブジェクト構造５７１３は、リモートリーダ１のＣＰＵ４５により読み取られる。ＣＰＵ４５は対応するＩＮＳＥＲＴメッセージ、ＲＥＭＯＶＥメッセージ、ＰＲＥＳＳメッセージ、ＲＥＬＥＡＳＥメッセージ、ＭＯＶＥメッセージ、ＢＡＤＣＡＲＤメッセージ、又はＬＯＷ ＢＡＴメッセージをＩ／Ｏデーモン３００を介してイベントマネージャ３０１に送信する。以下で更に詳細に後述するが、イベントマネージャ３０１は２１種類のコアメッセージを有し、これらのメッセージはＭＬ３０２、ランチャ３０３、及びアプリケーション３０４との間で送受信される。

【０１６９】

５．２．５　ＩＮＳＥＲＴメッセージ

ＩＮＳＥＲＴメッセージは上述のメッセージヘッダ及び挿入されたカード１０からのカードデータオブジェクトの内容から形成される。図２１ＡはＩＮＳＥＲＴメッセージの形式を示す。

【０１７０】

６．０　リーダファームウェア

６．１　概要

マイクロコントローラ４４は内蔵の不揮発性メモリ４６を有し、この不揮発性メモリ４６は詳細に後述するファームウェアを用いてプログラムすることができる。マイクロコントローラ４４及び周辺ハードウェア（コンピュータ１００など）と協働するファームウェアは、リモートリーダ１の機能的要求を指図することができる。

【０１７１】

６．２　コード種別

リモートリーダ１のユーザに対する費用を最小限に抑えるためには、リモートリーダ１上のメモリを最小限にするのが好ましい。従って、リモートリーダ１用に作成されるアプリケーションプログラム（すなわち、ファームウェア）は可能な限り小規模且つ高速にしなければならない。

【０１７２】

６．３　リソース制約条件

マイクロコントローラ４４は以下のような特徴を有する：

６．３．１　不揮発性メモリ

フラッシュメモリ４６は４０９６バイトのフラッシュＲＯＭを有するように構成され、ファームウェア格納用に利用することができる。フラッシュＲＯＭは再プログラム可能であるが、大量生産の場合、マスクＲＯＭ部品を利用することができる。

【０１７３】

６．３．２　ランダムアクセスメモリ（ＲＡＭ）

ＲＡＭ４７はファームウェアにより使用される１２８バイトのＲＡＭとして構成される。

【０１７４】

６．４　割込み

リモートリーダ１は、マイクロコントローラ４４がサポートする数多くの割込み原因のうちの２つを使用する。これらの割込みは以下に示すように説明することができる：

６．４．１　受信データ割込み

赤外線（ＩＲ）シリアルデータ受信機は、一般的に、入力データを受信すると立下り端を生成する。このデータは可能な限り迅速にサンプリング・バッファリングされる必要がある。マイクロコントローラ４４の１つのポートは、立下り端で割込みを開始することができる入力タイミングキャプチャピンの役割を兼ねる。

【０１７５】
６．４．２　タイマオーバフロー割込み
マイクロコントローラ４４は、オーバーフローしたときに割込みを発生するようにプログラム可能な自走１６ビットタイマを有する。４．９１ＭＨｚクロックソース及びプリスケール係数６４の場合、３．４１秒ごとの割込みになる。割込みサービスルーチンは、好ましくは、約１分の休止の後に低電力モードでのサスペンド状態を引き起こすカウンタを増分する。
【０１７６】
６．５　リセット
マイクロコントローラ４４は５つのリセットソースをサポートするが、リモートリーダ１は全てのリセットリソースを使用するように構成されるのが好ましい。これらのリセットリソースは以下のように説明することができる：
６．５．１　パワーオンリセット（ＰＯＲ）
ＰＯＲリセットは新規のバッテリがリモートリーダ１に取り付けられると開始される。マイクロコントローラ４４はパワーオン状態を検知しリセットを生成する回路を含む。
【０１７７】
６．５．２　低電圧禁止（ＬＶＩ）リセット
ＬＶＩリセットは、供給電圧が２．４ボルトを割ったことをマイクロコントローラ４４内の回路（不図示）が検知すると開始される。この種のリセットが発生するとリセット状態レジスタ（ＲＳＲ）中のフラグが設定され、初期化ルーチンはバッテリ５３が切れかけていると推定することができる。例えば、赤外線データの送信中、データがパルス状に送信されると共に赤外線ＬＥＤは大電流を消費する。バッテリ５３が切れると、送信中に供給電圧は２．４ボルトの閾値を割ってＬＶＩリセットを引き起こす可能性がある。リセット後、バッテリ５３の電圧は回復し、ＬＶＩリセットは次の大電流ドレインまで発生しない。これにより、ユーザによるバッテリ５３の交換を促すことができるように、リモートリーダ１にはバッテリ５３の消耗を関連するセットトップボックス又はリモート機器にフラグを立てて通知する機会が与えられる。
【０１７８】
６．５．３　コンピュータ正常動作（ＣＯＰ）リセット
ＣＯＰリセットはマイクロコントローラ４４が特定の動作を異常に長い時間実行したままでいる場合、マイクロコントローラ４４をリセットするように構成される。ＣＯＰ回路はオーバーフローする許可が得られる場合にリセットを生成するカウンタの形態をとる。ＣＯＰレジスタはＣＯＰリセットを回避するために所定の時間間隔で書き込まれなければならない。
【０１７９】
６．５．４　不正アドレス／演算コードリセット
不正アドレス／演算コードリセットは、マイクロコントローラ４４が所定の範囲外のアドレス及び所定の条件に合致しない演算コードに遭遇する場合に生成される。このリセットは停止することができず、コードデバッギング中にのみ出現するべきである。
【０１８０】
６．５．５　ハードウェアリセット
ハードウェアリセットは通常動作中にマイクロコントローラ４４の「リセット」ピンをローに駆動することによって生成される。また、マイクロコントローラ４４が低電力モードにある場合、割込み要求（ＩＲＱ）ピンの立下り端もハードウェアリセットを生成する。このリセットはファームウェアにおいてマイクロコントローラ４４を低電力モードからウェイクアップするのに使用されるメカニズムである。ＩＲＱピンがこの機能に適しているが、それは、リセットピンのようにレベルを感知するのではなく、端のみを感知するように構成することができるからである。
【０１８１】
６．６　メモリカード／ＣＰＵカードインタフェース

ファームウェアは、集積回路プロトコル（例えば、Ｉ２Ｃプロトコル）を使用するメモリカード周辺装置のみをサポートするのが好ましい。また、ファームウェアはＣＰＵカード形式をサポートすることができる。

【０１８２】

６．７　電力消費

ファームウェアはバッテリ５３の寿命を伸ばすのに重要な役割を果たす。マイクロコントローラ４４が実行する全ての動作は、可能な限り迅速にして電力の消費を可能な限り抑えるように最適化される。リモートリーダ１がある時間（例えば、１分）非アクティブ状態であると、マイクロコントローラ４４は低電力モードでサスペンドし、バッテリの寿命を更に伸ばす。低電力モードは通常動作モードの約１／１０００の電流を消費するので、このモードでの効率的なサスペンドが非常に望ましい。ファームウェアは、低電力モード中マイクロコントローラ４４のポートの状態を制御する。

【０１８３】

６．８　デバイスプログラミング

マイクロコントローラ４４は、マイクロコントローラ４４内の内蔵モニタによりサポートされるシステム内プログラム（ＩＳＰ）機能を使用してプログラムすることができる。モニタコードは、通常、メーカにより工場で設定されており、変更することができない。

【０１８４】

特定のハードウェアに対するマイクロコントローラ４４のプログラミングは、インサーキットシミュレータ（ＩＣＳ）キット及びモニタモードダウンロードケーブルを使用して実行することができる。このケーブルは、マイクロコントローラ４４のＶＣＣピン、ＧＮＤピン、ＲＳＴピン、ＩＲＱピン、及びＰＲＢ０ピンを使用する。プログラムするソースコードは、例えば、ＩＣＳハードウェアに対するコンピュータシリアルポート及びマイクロコントローラ４４のピンに対するダウンロードケーブルを介してＷｉｎｄｏｗｓ（登録商標）９５開発環境から供給することができる。このプログラミング方法はファームウェア開発／検査には理想的であるが、大量生産の場合は変更しても良い。マイクロコンピュータにおいてはモニタモードプログラミングモデルが好ましく、生産用の埋込み型プログラミングジグを使用することができる。生産ＩＳＰを考慮して信号をプログラムするための検査ポイントを提供することができる。ファームウェアがマイクロコントローラ４４へとマスクプログラムされる場合、デバイスプログラミングは必要とされない。

【０１８５】

６．９　ファームウェアプログラミングシーケンス

ローカルコンピュータ１００に接続されて動作中のリモートリーダ１を参照して、ファームウェアのプログラミングを説明する。

【０１８６】

６．９．１　メインループ

図２５は、ソフトウェアアーキテクチャ２００を組み込むシステム６００のリモートリーダ１により実行される読取り方法２５００を示すフローチャートである。方法２５００は、上述のように、リセットイベントの発生後に開始される。方法２５００はＣＰＵ４５により実行される。図２５の方法は「一定ペースのループ」状に構成される。すなわち、方法２５００は１０ミリ秒の遅延を生成するルーチンによりペース設定される。この遅延は、割込みの処理に十分な待ち時間を提供しながら必要なルーチンに適切なサービスを与える。

【０１８７】

第１のステップ２６００において、初期化ルーチンがＣＰＵ４５により実行される。初期化ルーチンは構成レジスタを初期化するために実行されるものであり、図２６のフローチャートを参照して後述する。方法２５００は次のステップ２５０１へと続く。ステップ２５０１において、コンピュータ正常動作（ＣＯＰ）レジスタがクリアされるが、これは、ファームウェアが循環ループに入り込んでいないことを示す。次のステップ２７００において、特定のスマートカード１０の存在及び有効性の変化をチェックするためにＣＰＵ４

５によりチェックカードプロセスが実行される。チェックカードプロセスについては、図２７を参照して以下で詳細に説明する。方法２５００は次のステップ２８００へと続く。ステップ２８００において、ユーザによるタッチパネル８の接触をチェックするためにＣＰＵ４５により走査タッチパネルプロセスが実行される。次のステップ２９００において、ＣＰＵ４５により１０ミリ秒待機ルーチンが実行され、方法２５００はステップ２５０１へと戻る。

【０１８８】

６．９．１　初期化プロセス

上述の５つのソースのうちのいずれか１つによるリセット後、全ての構成レジスタは正しい初期化を必要とする。ＬＶＩリセットが受信された場合、「バッテリ切れ可能性」フラグが設定される。図２６は、ソフトウェアアーキテクチャ２００を組み込むシステム６００を初期化する方法２６００を示すフローチャートである。方法２６００はＣＰＵ４５により実行されるものであり、ステップ２６０１で開始される。ステップ２６０１において、全てのレジスタが所定のデフォルト状態へと初期化される。次のステップ２６０２において、ＣＰＵ４５によりチェックが実行され、リセットがＬＶＩリセットであるか否かが判定される。ステップ２６０２においてＬＶＩリセットでない場合、方法２６００は終了する。ＬＶＩリセットの場合、方法２６００はステップ２６０３へと進む。ステップ２６０３において、バッテリ切れ可能性フラグが設定され、方法２６００は終了する。

【０１８９】

６．９．２　チェックカードプロセス

図２７は、ソフトウェアアーキテクチャ２００を組み込むシステム６００のカード１０をチェックする方法２７００を示すフローチャートである。上述のように、方法２７００はリモートリーダ１においてスマートカード１０の存在及び有効性の変化をチェックし、それに従って応答する。方法２７００はＣＰＵ４５により実行されるものであり、ステップ７０１で開始される。ステップ７０１において、スマートカード１０がリモートリーダ１へと挿入される場合、方法２７００はステップ７０２へと進む。ステップ７０２において、カード１０が新規のカードである（すなわち、それ以前の状態ではリーダ１にカードが存在しなかった）場合、方法２７００はステップ７０３へと進む。新規のカードでない場合、方法２７００は終了する。次のステップ７０３において、「マジックナンバー」フィールド及び「チェックサム」フィールドがカード１０のメモリ１９に格納されたカードヘッダから読み取られ、適正さがチェックされる。「マジックナンバー」及び「チェックサム」が適正である場合、方法２７００はステップ７０４へと進む。方法２７００はステップ７０４へと続く。ステップ７０４において、区別用識別子がカードヘッダから読み取られ、「移動イベントなし」フラグ及び「イベント座標なし」フラグが設定される。カードデータが存在する場合には、それもこのステップ７０４においてカードから読み取られる。次のステップ７０５において、カードデータが存在する場合はそのカードデータも含むＩＮＳＥＲＴメッセージがコンピュータ１００に送信され、ＣＰＵ２０５により処理される。ステップ７０６において、「ビープ」が鳴って方法２７００は終了する。

【０１９０】

ステップ７０３において「マジックナンバー」フィールド及び「チェックサム」フィールドが適正でない（すなわち、カード１０が有効でない）場合、方法２７００はステップ７１０へと進む。ステップ７１０において、ビープ停止フラグ、移動イベントなしフラグ、及びイベント座標フラグが設定される。次のステップ７１１において、ＢＡＤ ＣＡＲＤメッセージがコンピュータ１００に送信され、ＣＰＵ２０５により処理される。ステップ７１２において、「ブープ(boop)」が鳴って方法２７００は終了する。

【０１９１】

ステップ７０１において、スマートカード１０がリモートリーダ１に挿入されていない場合、方法２７００はステップ７０７へと進む。ステップ７０７において、これがリセット後のリーダ１の最初の動作である場合、方法２７００は終了する。最初の動作でない場合、方法２７００はステップ７０８へと進む。ステップ７０８において、「ビープ停止」フ

ラグ、「移動イベントなし」フラグ、及び「イベント座標なし」フラグが設定され、メモリ４７に格納される区別用識別子が「ＮＯ＿ＣＡＲＤ」に設定される。次のステップ７０９において、ＲＥＭＯＶＥメッセージがコンピュータ１００に送信され、ＣＰＵ２０５により処理される。方法２７００はステップ７０９の後に終了する。

【０１９２】

６．９．３　走査タッチパネルルーチン

図２８は、ソフトウェアアーキテクチャ２００を組み込むシステム６００のリーダ１のタッチパネル８を走査する方法２８００を示すフローチャートである。上述のように、走査タッチパネルルーチンはカードボタン押下と同等のタッチパネル接触をチェックし、それに従って応答する。方法２８００はＣＰＵ４５により実行されるものであり、ステップ８０１で開始される。ステップ８０１において、パネル８が触れられている場合、方法２８００はステップ８０２へと進む。パネル８が触れられていない場合、方法２８００はステップ８１２へと進む。ステップ８１２において、パネル８が以前に触れられていた場合、方法２８００はステップ８１３へと進む。以前に触れられていなかった場合、方法２８００は終了する。

【０１９３】

ステップ８１３において、「ビープ停止」フラグ、「移動イベントなし」フラグ、及び「イベント座標なし」フラグが設定される。ステップ８１４において、メッセージ種別がＲＥＬＥＡＳＥに設定され、方法２８００はステップ８０５へと進む。

【０１９４】

方法２８００はステップ８０２へと続く。ステップ８０２において、接触がなかった状態以来の初めての接触の認知である場合、方法２８００はステップ８０３へと進む。次のステップ８０３において、ＣＰＵ４５はステップ７０３の結果をチェックすることによって、不良カードがリーダ１へと挿入されたか否かを判定する。不良カードがリーダ１へと挿入された場合、方法２８００はステップ８１５へと進む。ステップ８１５において、ＢＡＤ ＣＡＲＤメッセージがコンピュータ１００に送信され、メモリ２０６に格納され、方法２８００は終了する。ステップ８０３において、ステップ７０３の結果をチェックすることによってカードが有効である、あるいは、ステップ７０１のチェックによってリーダ１にカードが挿入されていないと判定された場合、方法２８００はステップ８０４へと進む。ステップ８０４において、図１９のメッセージヘッダ中のメッセージ種別がＰＲＥＳＳに設定される。次のステップ８０５において、ＣＰＵ４５はタッチパネルインタフェース４１を介して接触座標（すなわち、ユーザ押下位置のＸＹ座標）を判定する。次のステップ８０７において、オフセット機能及びスケール機能が座標に適用される。オフセット機能及びスケール機能は、タッチパネル８の座標空間をカード１０の座標空間へとマップする。方法２８００は次のステップ８０７へと続く。ステップ８０７において、ＣＰＵ４５がチェックステップ７０１により送信されたメッセージがＭＯＶＥである、及び／又は、カードがリーダ１に挿入されていないと判定する場合、方法２８００は直接ステップ８０９へと進む。それ以外の場合、方法２８００はステップ８０８へと進み、Ｘ１値、Ｙ１値、Ｘ２値、及びＹ２値が接触座標の位置する範囲を形成する最初のユーザインタフェース要素を探し出すためにカード１０のメモリ１９が検索され、一致するユーザインタフェース要素と関連付けられたデータがカード１０から読み取られる。ステップ８０９において、メッセージが任意のデータと共に関連するコンピュータ１００に送信され、コンピュータ１００のＣＰＵ２０５がそのメッセージを処理する。方法２８００は次のステップ２８１１へと進む。ステップ２８１１において、ビープ音が鳴って方法２８００は終了する。

【０１９５】

ステップ８０２において、接触がなかった状態以来の初めての接触の認知でない場合、方法２８００はステップ８１６へと進む。ステップ８１６において、ステップ８０１で検知された接触が移動である場合、方法２８００はステップ８１７へと進む。移動でない場合、方法２８００は終了する。ステップ８１７において、メッセージ種別がＭＯＶＥに設定

され、方法２８００はステップ８０５へと進む。例えば、ＭＯＶＥメッセージは図１９及び図２２により定義されるような接触位置のＸＹ座標と共に送信することができ、ＰＲＥＳＳメッセージ及びＲＥＬＥＡＳＥメッセージは図１９及び図２３により定義されるような接触位置のＸＹ座標及びユーザインタフェースオブジェクト（すなわち、印１４のうちの１つ）と関連付けられるデータと共に送信することができる。ステップ８０７において、メッセージがＭＯＶＥであると判定された場合、ステップ８０９において、ＣＰＵ４５はコンピュータ１００にＭＯＶＥメッセージを送信する。ＣＰＵ２０５はＸＹ座標をカーソル情報として処理し、ビデオディスプレイ１０１に表示されるカーソルを移動させる。この場合、次のＲＥＬＥＡＳＥメッセージはカーソル位置で表示されるオブジェクトを選択する（例えば、プログラムを実行する、項目を選択する、又はＵＲＬをロードする）コマンドとして解釈することができる。更に、イベント座標なし（図１３参照）がカード１０に設定されている場合、リーダ１は接触の位置のＸＹ座標を送信せずに、ユーザインタフェースオブジェクトと関連付けられるデータをコンピュータ１００又はＳＴＢ６０１中のイベントマネージャ３０１に送信しても良い。

【０１９６】

加えて、アプリケーション３０４が図１７に示すようなユーザインタフェースオブジェクト構造及びステップ８０８のような照合機能を有する場合、リーダ１は接触位置のＸＹ座標をアプリケーション３０４に送信しても良い。その結果、ＣＰＵ２０５は同じ照合機能を実行し、ユーザインタフェースオブジェクトと関連付けられるデータをイベントマネージャ３０１から読み、読んだデータと関連付けられるサービス識別子１１０６により識別されるサービス（ゲームなど）をカードユーザに提供する。例えば、図４１のステップ４２０５において、ＣＰＵ２０５はメッセージのデータフィールドにデータが存在するか否かを判定する。データがデータフィールドにある場合、ＣＰＵ２０５はデータを読み、図４１の次のステップでデータの処理を行なう。データがデータフィールドに存在しない場合、ＣＰＵ２０５はメッセージからＸＹ座標を読み、その座標に対して照合機能を実行し、ユーザが押下した印と関連付けられるデータを取得する。また、イベントマネージャ３０１は、それ自体にとって利用可能なユーザインタフェースオブジェクト構造を使用してこの機能を実行することができる。

【０１９７】

従って、カードユーザがタッチペイン１８上で指を動かすことによってリーダ１を（カード１０を挿入せずに）マウスとして使用する場合、ユーザはＴＶディスプレイ上に表示されたＳＴＢメニューのＳＴＢサービスのうちの１つを選択することができる。また、カードユーザが挿入されたカード１０と共にリーダ１を使用し、何らかの印１４を選択する場合、ユーザはコンピュータ１００又はＳＴＢ６０１からサービス（ゲームなど）を受信する。特に、ユーザがＳＴＡＲＴ印を選択する場合、所望のゲームをコンピュータ１００又はＳＴＢ６０１において実行することができ、ゲーム中のオブジェクトがＫＩＣＫ印１４の選択に従ってボールをキックする。

【０１９８】

カード１０に対して予めカードごとのフラグ値を定義することによって、種々のカード１０をユーザに提供することができる。例えば、「移動イベントなし」のフラグ（すなわち、情報）が予めカード１０に設定されている場合、リーダ１はフラグに基づいてマウスとして動作しないように構成することができる。これに対し、「移動イベントなし」のフラグが予めカード１０に設定されていない場合、リーダ１はフラグに基づいてマウスとして機能するように構成することができる。

【０１９９】

図１３に示すように、リーダ１はデフォルト状態を有する。このデフォルト状態において、リーダ１は音声フィードバックを提供し、マウスとして機能し、押下イベント、押下解除イベント、及びその他のイベントに対する座標を送信する。また、リーダ１は音声フィードバックを提供せず、マウスとして機能せず、座標も送信しないデフォルト状態を提供することもできる。

【０２００】
リーダ１がカードごとのフラグ値を使用して「ビープ機能」を実行するように構成される場合、リーダ１は「ビープ」を鳴らし、図２７及び図２８に示すフローチャートに従って方法を実行する。また、リーダ１がカードごとのフラグ値を使用して「マウス機能」を実行するように構成される場合、リーダ１はマウスとして機能し、図２７及び図２８のフローチャートに従って方法を実行する。更に、リーダ１がカードごとのフラグ値を使用して「照合機能」を実行するように構成される場合、リーダ１は押下イベント、押下解除イベント、及び移動イベントに対する座標を送信し、図２７及び図２８のフローチャートに従って方法を実行する。
【０２０１】
図２８のステップ８０８のようにＥＭ３０１でも照合機能が実行される。また、カード１０はマウス機能及び／又は基本機能（例えば、ユーザが選択した印と関連付けられるデータをＥＭ３０１に送信）のみを有するカードとして構成することもできる。このため、カードごとのフラグ値をランダムに組み合わせることによって、種々のカード１０をユーザに提供することができる。
【０２０２】
ここで説明するように、サービス識別子１１０６はシステム６００にとって不可欠な識別子である。少なくとも区別用識別子１１１０中のサービス識別子１１０６をＥＭ３０１に送信することによって、サービスをユーザに提供することができる。
【０２０３】
上述のサービス固有識別子１１０７は、特定のアプリケーションと共に使用されるようにベンダにより設定されるのが好ましい。このため、ベンダが各カード１０に対して一意的なサービス固有識別子１１０７を定義する場合、カード１０も一意的になるであろう。サービス固有識別子１１０７が特定のカードが配布された手段（郵便、列車内での配布など）についての情報を提供するのに使用されている場合、サービスにアクセスするのに使用されるのがどのカードであるかの記録を提供するファイルにそのサービス固有識別子１１０７を加えることができる。この情報はどれほど効果的な種々の配布手段が使用されたかを後で判定する際に使用される。
【０２０４】
６．９．４　１０ミリ秒待機プロセス
図２９は、１０ミリ秒待機ルーチン２９００を示すフローチャートである。１０ミリ秒待機ルーチン２９００は１０ミリ秒が経過するまでＣＰＵサイクルを消費するようにループする。遅延プロセス２９００はＣＰＵ４５により実行されるものであり、ステップ９０１で開始される。ステップ９０１において、所定のプロセスカウンタがクリアされる。次のステップ９０２において、カウンタが増分される。ステップ９０３において、１０ミリ秒が経過していない場合、方法２９００はステップ９０２に戻る。経過した場合、遅延プロセス２９００は終了する。
【０２０５】
７．０　イベントマネージャ
イベントマネージャ３０１は、ソフトウェアアーキテクチャ２００のプロセスコンポーネントのうちの１つである。イベントマネージャ３０１はアーキテクチャ２００の各規則を実施し、その他のプロセスコンポーネント間の整合性のある挙動を保証する。
【０２０６】
７．１　システムにおける役割
大半の通信はイベントマネージャ３０１を介して行なわれるので、イベントマネージャ３０１は、アーキテクチャ２００の中でもディレクトリサービス３１１コンポーネントを除く全てのプロセスコンポーネントが直接送受信可能でなければならない唯一のコンポーネントである。イベントマネージャ３０１はアーキテクチャ２００の各規則の実施者として機能するが、１つの別個のプログラムとして構成される必要はない。また、イベントマネージャ３０１は、イベントマネージャの役割の一部を実行する複数の高信頼中継器又はそ

の他の別々のプロセスコンポーネントから形成することができる。これは、例えば、効率又は安全上の理由から行なうことができる。

【０２０７】

イベントマネージャ３０１は、Ｉ／Ｏデーモン３００及びランチャ３０３などのソフトウェアアーキテクチャ２００のその他の種々の部分を組み込んでも良い。イベントマネージャ３０１はブラウザコントローラなどのアプリケーションでさえも組み込んで良い。

【０２０８】

イベントマネージャ３０１は、直接に又は高信頼中継器を介してディレクトリサービス３１１を除くシステム６００の全てのプロセスコンポーネントと通信することができる。これらのコンポーネントには、Ｉ／Ｏデーモン３００、ランチャ３０３、及びアプリケーション３０４のいずれかが含まれる。イベントマネージャ３０１は他のプロセスコンポーネントと通信するのに任意の適切な通信方法を使用することができる。好適な通信方法は、ほぼ世界的に実現されている伝送制御プロトコル／インターネットプロトコル（ＴＣＰ／ＩＰ）であるが、Ｕｎｉｘ(登録商標）ソケットなどのその他のＯＳ特有の方法も使用することができる。プロセスコンポーネントが統合されるときには、通信するのに使用される方法は内部データの別々のスレッド間での送受信となる可能性がある。

【０２０９】

イベントマネージャ３０１は、イベントマネージャ３０１をキルすることが可能な、あるいは、イベントマネージャ３０１のＣＰＵ時間又はネットワーク帯域幅を枯渇させることが可能な他のプロセスを含む他のプロセスコンポーネントからの干渉に影響されないように構成されるのが好ましい。これにより、イベントマネージャ３０１は確実にシステム６００の最終的な制御を維持することができる。

【０２１０】

７．２　内部の要求事項

イベントマネージャ３０１は、ポーリング（注：ポーリングはＣＰＵ負荷の関係で推奨しない）、割込み駆動型Ｉ／Ｏ、各コンポーネントを読み書きする別々のスレッドを有する方法、又は目的を達成するその他の任意の適切な方法などの方法により、アーキテクチャ２００の他の全てのプロセスコンポーネント３００、３０３、３０４、及び３０６に対して無閉塞Ｉ／Ｏを実行する。これにより、１つのコンポーネントの寿命が別のコンポーネントにより尽きてしまうことが確実になくなり、ユーザの待ち時間も減少する。

【０２１１】

また、イベントマネージャ３０１は、全ての入力データの有効性をチェックし、可能であれば出力前にデータを修復するように構成される。これは、高信頼コンポーネントからのデータを含む。イベントマネージャ３０１もフェールセーフであるのが好ましい。コンポーネント３００、３０３、３０４、及び３０６のうちの１つから不測のデータを受信する場合、イベントマネージャ３０１はそのデータを処理し、完全に不可避でない限りは終了しないように構成される。

【０２１２】

イベントマネージャ３０１は、相当に長い時間実行されるように要求される可能性があり、時間が経過しても性能が低下しないことを保証するように構成される。イベントマネージャ３０１は、伝送メカニズムが所定のイベントマネージャプロトコル（すなわち、ＥＭプロトコル）を使用する任意のコンポーネントとの通信において信頼性が高いことを前提とするのが好ましいが、Ｉ／Ｏデーモン３００を介してリモートリーダ１と通信を行なうのに使用される伝送メカニズムは信頼性が低く、入力データの一部が正しくないか、あるいは、欠落している恐れがあることを前提としている。

【０２１３】

７．３　手順

イベントマネージャ３０１は、システム６００の動作の一部に直接関与しているが、アーキテクチャ２００のその他の動作の多くにも透過的に参加している。イベントマネージャ３０１は、データパケットの通過の際にこれを変更せずに使用する点で透過的である。特

に、第８．０節を参照して手順を詳細に後述する。

【０２１４】

図３０は、ソフトウェアアーキテクチャ２００を組み込むシステム６００により実行されるイベントの概要プロセス３０１０を示すフローチャートである。プロセス３０１０は、システム６００の構成によってＣＰＵ２０５により実行される。プロセス３０１０はステップ３０００で開始される。ステップ３０００において、イベントマネージャ３０１の開始を含むシステム初期化ルーチンが実行される。ステップ３０００において、Ｉ／Ｏデーモンも通常イベントマネージャ３０１と共に開始される。

【０２１５】

次のステップ３７００において、イベントマネージャ３０１はランチャ３０３を開始させる。ステップ３３００において、イベントマネージャ３０１はメッセージをランチャ３０３に渡し、ランチャ３０３がどのアプリケーション３０４を実行すべきかを判定できるようにする。ランチャ３０３は対応するアプリケーション３０４を開始させる。プロセス３０１０は次のステップ３４００へと続く。ステップ３４００において、新規のカード１０がリーダ１に挿入されるときなどに現在実行中のアプリケーション３０４が必要とされなくなると、ランチャ３０３は実行中のアプリケーションの実行を終了するために実行中のアプリケーションに終了メッセージを供給する。システム６００の電源が落とされる（スイッチを切られる）ときには全てのアプリケーションが終了させられる。

【０２１６】

図３１は、イベントマネージャ３０１により実行されるイベントを受信する方法３０００を示すフローチャートである。方法３０００は、コンピュータで実現する場合にはＣＰＵ２０５により実行することができる。また、セットトップボックスで実現する場合にはＣＰＵ４３０５により実行することができる。方法３０００はステップ３１０１で開始される。ステップ３１０１において、ランチャ３０３が開始される。次のステップ３１０３において、イベントマネージャ３０１はイベントを受信する。次のステップ３１０５において、ステップ３１０３で受信されたイベントがリモートリーダ１からのものでない場合、方法３０００はステップ３１０７へと進む。ステップ３１０７において、コンポーネント識別子（ＸＩＤ）がチェックされ、必要に応じて訂正される。方法３０００は次のステップ３１０９へと続く。ステップ３１０９において、イベントを送信しようとする新規のアプリケーションがそのイベントの送信を許可される場合、方法３０００はステップ３１１１へと進む。ステップ３１１１において、イベントが宛先プロセスコンポーネントに送信され、方法３０００はステップ３１０３へと戻る。送信元のアプリケーションがステップ３１０９においてイベントの送信を許可されない場合、方法３０００はステップ３１１３へと進む。ステップ３１１３において、イベントは放棄され、方法３０００はステップ３１１３へと戻る。

【０２１７】

ステップ３１０５において、イベントがリモートリーダ１からのものである場合、方法３０００はステップ３１１５へと進む。ステップ３１１５において、イベントがＢＡＤＣＡＲＤイベント、ＬＯＷＢＡＴイベント、ＩＮＳＥＲＴイベント、又はＲＥＭＯＶＥイベントの場合、方法３０００はステップ３１１７へと進む。それ以外のイベントの場合、方法３０００はステップ３１１９へと進む。ステップ３１１７において、イベントはランチャ３０３に渡され、方法３０００はステップ３１０３へと戻る。ステップ３１１９において、区別用識別子がＮＯ＿ＣＡＲＤ識別子である場合、ステップ３１１７において、対応するメッセージがランチャ３０３に渡される。ＮＯ＿ＣＡＲＤ識別子でない場合、方法３０００はステップ３１２１へと進む。ステップ３１２１において、区別用識別子のサービス識別子部分が現在のフロントアプリケーションを判定するのに使用されるサービス識別子と比較される。サービス識別子がフロントアプリケーションを判定するのに使用されたサービス識別子と同じでなく、区別用識別子のサービス識別子部分が特殊な総称サービス識別子でない場合、方法３０００はステップ３１１７へと進む。ステップ３１１７において、このメッセージはランチャ３０３に渡される。フロントアプリケーション判定用のサー

ビス識別子と同じであるか、あるいは、特殊な総称サービス識別子である場合、方法３０００はステップ３１２３へと進む。ステップ３１２３において、イベントはフロントアプリケーションに送信され、方法３０００はステップ３１０３へと戻る。

【０２１８】

７．４　フォーカス変更

イベントマネージャ３０１は、現在のフロントアプリケーションに対するものでないEM_LOSING_FOCUSイベントを問題なく無視することができる。イベントマネージャ３０１は、フロントアプリケーションになるアプリケーション及びそのアプリケーションと関連付けられるサービス識別子に対するEM_GAINING_FOCUSメッセージに注意する必要がある。イベントマネージャ３０１は、同じサービス識別子を有する同じアプリケーションに対する複数のEM_GAINING_FOCUSイベント及び現在フロントアプリケーションでないアプリケーションに対するEM_LOSING_FOCUSイベントを問題なく無視することができる。無視されるメッセージは通常通りに渡される。

【０２１９】

７．５　リーダメッセージ

イベントマネージャ３０１は、メッセージを正しいコンポーネントに配布する役割も担う。イベントマネージャ３０１は所定のプロトコル規則に従うように構成されるが、これについては詳細に後述する。

【０２２０】

７．６　メッセージ送信に対する規制

イベントマネージャ３０１の更なる役割は、メッセージの送信に対する所定の規制を実施することである。

【０２２１】

８．０　イベントマネージャプロトコル

イベントマネージャプロトコル（ＥＭプロトコル）は、ディレクトリサービス３１１を除くアーキテクチャ２００の全てのコンポーネント間で通信を行なうのに使用されるプロトコルである。一般的に、全てのメッセージは指定受信者に渡される前にイベントマネージャ３０１を通過するように構成される。ＥＭプロトコルは、例えば、伝送制御プロトコル／インターネットプロトコル（ＴＣＰ／ＩＰ）などの信頼性の高い通信プロトコルの上で実現されるデータグラムベースのプロトコルである。イベントマネージャ３０１は、送信される全てのデータが変更されずに正しい順序で到着することを前提とするように構成される。イベントマネージャ３０１は、アーキテクチャ２００のプロセスコンポーネント間で同期をとる信頼性の高い方法があることを前提としない。

【０２２２】

全てのマルチバイト値がインターネットバイト順（すなわち、ビッグエンディアン（big endian））で送信される。この例外は、各サービスを表す「区別用識別子」の値である。この値は、幾つかの１バイトから成るブロックとして送信され、常にそのように（すなわち、区別用識別子の値はバイト配列の問題のため、通常、数値として格納されることはない）扱われる。

【０２２３】

８．１　通信方法

イベントマネージャプロトコルは、アーキテクチャ２００のコンポーネントとシステム６００のハードウェアコンポーネントとの間での通信方法のように、ＴＣＰ／ＩＰを前提とするように構成されるのが好ましい。また、信頼性の高い転送を保証する既知の通信方法を使用することもできる。例えば、「ＵＮＩＸソケット」などのオペレーティングシステム特有の方法も使用することができる。データは、マルチスレッドのアプリケーションなどにおける内部データ構造を直接介してプロセスコンポーネント３０１、３０３、３０４、及び３０６間で受け渡しすることができる。

【０２２４】

コンポーネント間の通信の代替の方法が使用されるアーキテクチャの場合、バイト配列の

問題が考慮されなければならない。アプリケーションが異なるバイト配列を有するマシン上で実行することが可能であるか、あるいは、データがネットワークバイト順（全てのコンポーネントはデフォルトでこの順序を前提とする）であることを期待するコンポーネントと通信を行なうことが要求される場合、全ての対象となる通信はネットワークバイト順で行なうことができる。

【０２２５】

８．２　データ形式

８．２．１　基本的なデータ型

データ型に言及すべく以下の段落で使用される略語の一部は以下の通りである：

・ｉｎｔ８：　　８ビット符号付き値、
・ｕｎｉｔ８：　８ビット符号無し値、
・ｉｎｔ１６：　１６ビット符号付き値、
・ｕｎｉｔ１６：　１６ビット符号無し値、
・ｉｎｔ３２：　３２ビット符号付き値、
・ｕｎｉｔ３２：　３２ビット符号無し値、及び
・ｘｉｄ＿ｔ：　３２ビット符号無し値

８．２．２　コンポーネントアドレッシング

アーキテクチャ２００中の全てのアドレス指定可能なプロセスコンポーネントには、「ｘｉｄ」（又はコンポーネント識別子）と呼ばれる３２ビット符号無し値が割り当てられる。この数値は個々のシステム６００のインスタンスの範囲内では一意的である。プロセスコンポーネントの一部のｘｉｄは常に同じである。そのｘｉｄは以下の通りである：

・イベントマネージャ３０１：EM_EVENT_MANGER_XID
・マスタランチャ：EM_MASTER_LAUNCHER_XID
・ランチャ３０３：EM_FIRST_APP_XID
・ディスプレイマネージャ３０６：EM_DISPLAY_MANAGER_XID

ｘｉｄ値は１バイトの型フィールドと３バイトの識別子とに分割される。種々の型が以下の表１に示される。

【０２２６】

【表１】

表1

値	型
内部ｘｉｄ	これらｘｉｄ値は転送可能ではなく、全てのコンポーネントは内部で使用することができる。EMにより見られる場合には放棄される
コアシステムのｘｉｄ	これらはユーザインタフェースカードシステムのコアシステムコンポーネントを識別する。これらのコンポーネントには、イベントマネージャ、ランチャ、及びマスタランチャが含まれる
標準アプリケーション	これは必要に応じてランチャにより開始／終了される標準アプリケーションである
特殊なアプリケーション	これはアプリケーションを開始／終了するための標準規則により制御されない特殊なアプリケーションを識別する。このアプリケーションは、ビデオオンデマンドプレーヤ又はブラウザコントローラのような他のアプリケーションにより制御することができる機能をユーザインタフェースカードシステムに提供するために作成されるアプリケーションである
リーダ	リーダにはイベントマネージャによりｘｉｄが割り当てられる。このｘｉｄは、イベントマネージャの存続中システムにアクセスするのに使用される各リーダに固有のものである。イベントマネージャ及びシステムが再始動される場合、リーダのｘｉｄは変更することになる

【０２２７】
８．３　メッセージ種別
ＥＭプロトコルには２２個のコアメッセージがあり、これらは以下のラベルを有するのが好ましい：
・EM_NEW_LAUNCHER
・EM_KILL_LAUNCHER
・EM_APP_REGISTER
・EM_EXIT_NOW
・EM_CLOSE
・EM_APP_STARTING
・EM_APP_DYING
・EM_GAINING_FOCUS
・EM_LOSING_FOCUS
・EM_LIST_MESSAGES
・EM_LIST_APPS
・EM_SEND_MESSAGE
・EM_POST_MESSAGE
・EM_GET_MESSAGE
・EM_DELETE_MESSAGE
・EM_READER_INSERT
・EM_READER_REMOVE
・EM_READER_BADCARD
・EM_READER_MOVE
・EM_READER_PRESS
・EM_READER_RELEASE
・EM_READER_LOW_BATT
これらのメッセージについては、以下の段落で詳細に説明する。
【０２２８】
８．３．１　メッセージヘッダ
システム６００内で送信されるメッセージは、好ましくは、以下の情報を含むヘッダ部分を有する：
ｖｅｒｓｉｏｎ：　これは、コンポーネントにより使用されるプロトコルのバージョン番号を表す。これは、常にEM_PROTOCOL_VERSIONに設定されるべきであり、EM_PROTOCOL_VER　SIONはライブラリにより使用されるバージョンとしてライブラリヘッダ中に定義される。
ｔｙｐｅ：　これは、ヘッダが開始するメッセージ種別を表し、後述の列挙されたメッセージ種別のうちの１つに設定される。メッセージの長さにはラベルｄａｔａＬｅｎｇｔｈが割り当てられる
ｒｅｓｅｒｖｅｄ：　これは、この２バイトの値が予約されていることを表し、０に設定されるべきである。
ｔｉｍｅｓｔａｍｐ：　これは、データパケットのタイムスタンプを表す。
ｔｏ＿ｘｉｄ：　これは、特定のパケットの宛先ｘｉｄを表す。これは、パケットの最終宛先であり、イベントマネージャが指定の最終受信者である場合はイベントマネージャにのみ設定されるべきである。
ｆｒｏｍ＿ｘｉｄ：　これは、パケットの送信元ｘｉｄを表す。
ｄａｔａＬｅｎｇｔｈ：これは、ヘッダに続くデータの長さを表す。この値は０となる可能性がある。異なる種別のメッセージはメッセージヘッダに続くデータに対して異なる要求事項を課す。コンポーネントは種別からメッセージの長さを想定すべきでない。メッセージの適正なサイズと異なる場合であってもｄａｔａＬｅｎｇｔｈフィールドのバイト数は常に読まれ、適正でないｄａｔａＬｅｎｇｔｈによってのみストリームは破損する可能

性があることが保証される。

【０２２９】

８．３．２　EM_NEW_LAUNCHER

EM_NEW_LAUNCHERメッセージは、イベントマネージャ３０１が新規のランチャ３０３を要求するときに送信される。ソフトウェアアーキテクチャ２００がマスタランチャを含む場合、このメッセージはイベントマネージャ３０１とこのようなマスタランチャとの間でのみ送信される。このメッセージを含むパケットは、新規のランチャがイベントマネージャ３０１に接続する必要があるという情報も含む。EM_NEW_LAUNCHERメッセージは以下の情報を含むのが好ましい：

ｐｏｒｔ：　　これは、イベントマネージャ３０１が新規の接続の有無を聞き取っているポート番号を表す。

ｈｏｓｔ：　　これは、イベントマネージャ３０１上で実行中のマシンのホスト名を表す。

【０２３０】

８．３．３　EM_KILL_LAUNCHER

EM_KILL_LAUNCHERメッセージは、イベントマネージャ３０１がマスタランチャに現在のランチャ３０３をキルすることを望むときに送信される。EM_KILL_　LAUNCHERメッセージと関連付けられるデータはない。

【０２３１】

８．３．４　EM_APP_REGISTER

EM_APP_REGISTERメッセージは、ランチャ３０３に対してアプリケーションが起動し、アーキテクチャ２００の残りのコンポーネントにメッセージを受信する準備ができたことを通知するときに送信される。登録される前にアプリケーション３０４が送信するメッセージはイベントマネージャ３０１により破棄されることになる。

【０２３２】

EM_APP_REGISTERメッセージは以下の情報を含むのが好ましい：

ｘｉｄ：　　これは、アプリケーションランチャ３０３により割り当てられたコンポーネント識別子を表す。送信される情報の残りは、残りのフィールドが可変長であるために構造により表すことができない。ｘｉｄに続くデータは、一連のヌルで終結する文字列であり、最大長は終端のヌルを除いて２５６文字である。この文字列は、ａからｚまでの小文字及び大文字、数字０から９、及び各文字（．，から＿）から構成される。文字列が２５６文字よりも長い場合、２５６文字で切り捨てられる。

Ａｐｐｌｉｃａｔｉｏｎ　Ｎａｍｅ：これは、現在のアプリケーションを他のアプリケーションに対して識別するために使用される名前を表す。

Ｓｅｒｖｉｃｅ　Ｇｒｏｕｐ：これは、アプリケーションがその一部となることを希望するサービスグループの１つ以上の名前を表す。

【０２３３】

ブラウザコントローラなどの永続的なアプリケーションは１度登録されるだけで良い。このような永続的なアプリケーションはEM_GAINING_FOCUSイベントを取得するたびに登録する必要がない。

【０２３４】

８．３．５　EM_EXIT_NOW

EM_EXIT_NOWメッセージは、アプリケーションが強制されて終了しようとするときにランチャ３０３により送信される。EM_EXIT_NOWメッセージと関連付けられるデータはない。

【０２３５】

８．３．６　EM_CLOSE

EM_CLOSEメッセージは、現在のセッションが終了することを示し、アプリケーションを起動状態に戻すために永続的なアプリケーションに送信される。このメッセージを受信すると、アプリケーションは入力／出力フォーカスの変更ではなく、新規のセッションの開始として次のEM_GAINING_FOCUSイベントを処理するように要求される。EM_CLOSEメッセージ

と関連付けられるデータはない。

【０２３６】

８．３．７　EM_APP_STARTING

EM_APP_STARTINGメッセージは、アプリケーションが始動しようとしているときにランチャ３０３によりイベントマネージャ３０１に送信される。EM_APP_STARTINGメッセージは以下の情報を含むのが好ましい：

ｘｉｄ：　　これは、始動しようとするアプリケーションのコンポーネント識別子を表す。

【０２３７】

８．３．８　EM_APP_DYING

EM_APP_DYINGメッセージは、アプリケーションが終了したときにランチャ３０３によりイベントマネージャ３０１に送信される。EM_APP_DYINGメッセージはランチャ３０３がアプリケーションの終了を確信した後で初めて送信される。EM_APP_DYINGメッセージは以下の情報を含むのが好ましい：

ｘｉｄ：　　これは、終了したアプリケーションのコンポーネント識別子を表す。

【０２３８】

８．３．９　EM_GAINING_FOCUS

EM_GAINING_FOCUSメッセージは、アプリケーション３０４がリモートリーダ１からの入力の受信を開始しようとするときにランチャ３０３によりアプリケーションに送信される。EM_GAINING_FOCUSメッセージは以下の情報を含むのが好ましい：

ｉｄ：　　これは、アプリケーションに送信されるリモートリーダ１のメッセージの区別用識別子を表す。

Ｄａｔａ：　　これは、フォーカスを受け取ろうとするときにアプリケーションに送信される追加のデータを表す。これは各サービス特有のものであり、データを解釈するのはアプリケーションの責任である。追加のデータは、バイト配列の問題に関してチェックを受けない。これはアプリケーションが対処すべきである。マルチバイトデータはアプリケーションによりネットワークバイト順で送信され、受信側アプリケーションはこの順序であることを前提とする。受信側アプリケーションがブラウザコントローラであるとき、このデータの一例はブラウザコントローラがロードするように指示されるＵＲＬである。

【０２３９】

８．３．１０　EM_LOSING_FOCUS

EM_LOSING_FOCUSメッセージは、アプリケーション３０４がリモートリーダ１及びディスプレイ１０１からの入力／出力フォーカスを手放そうとするときに送信される。EM_LOSING_FOCUSメッセージには追加のデータはない。

【０２４０】

８．３．１１　EM_LIST_APPS

EM_LIST_APPSメッセージは、アプリケーションがある時点において他のどのアプリケーションが実行中であるか知ることを希望するときに送信される。EM_LIST_APPSメッセージは、アプリケーションリストを含むデータフィールドを伴ってアプリケーションに戻される。このメッセージはプロセスコンポーネント３０１から３０６のうちのいずれかに宛てて送信される必要はない。イベントマネージャ３０１は、ヘッダのｔｏ＿ｘｉｄフィールドに関係なく、EM_LIST_APPSメッセージが正しいコンポーネント（通常、ランチャ３０３）に送信されることを確認する。どのアプリケーションを列挙すべきか決定するのが受信側コンポーネントの役割である。

【０２４１】

応答として使用される場合、EM_LIST_APPSメッセージは２つの形式を有する。第１の形式はEM_LIST_APPSが要求として送信されるときに使用される形式であり、第２の形式は応答として送信されるときの形式である。要求と関連付けられる追加のデータはない。

【０２４２】
EM_LIST_APPSメッセージは以下の情報を含むのが好ましい：
ａｐｐ＿ｘｉｄ：　これは、記述されるアプリケーションのｘｉｄを表す。
ａｐｐ＿ｄｅｓｃ：　これは、アプリケーションが最初に登録されるときにランチャ３０３に与えられる名前の文字列を表す。
【０２４３】
８．３．１２　EM_SEND_MESSAGE
EM_SEND_MESSAGEメッセージは、システム６００中の任意の２つの同時に実行されているアプリケーション間で送信することができる。アーキテクチャ２００によりこのメッセージに強制される構造はないが、通信アプリケーションは共通のデータ構造について同意する必要がある。
【０２４４】
８．３．１３　EM_LIST_MESSAGES
EM_LIST_MESSAGESメッセージは、現在の掲示板(message board)にある全てのメッセージのリストを取得するのに使用される。この掲示板はアーキテクチャ２００において使用される。掲示板の詳細については、第８．４．７．１節を参照して後述することとする。EM_LIST_MESSAGESメッセージは要求形式及び応答形式を有する。要求形式と関連付けられるデータはない。応答は以下の情報を含むのが好ましい：
ｍｅｓｓａｇｅ＿ｃｏｕｎｔ：これは、現在掲示板にあるメッセージの数を表し、０に等しくなる可能性がある。
Ｍｅｓｓａｇｅｓ：　これは、以下の構造を有する可変サイズの構造の可変数（すなわち、ｍｅｓｓａｇｅ＿ｃｏｕｎｔに等しい）を表す。
【０２４５】
各メッセージは以下の情報を含むのが好ましい：
ｍｅｓｓａｇｅ＿ｉｄ：これは、このメッセージのメッセージ識別子を表す。
ｐｏｓｔｅｒ＿ｉｄ：　これは、このメッセージを提示するコンポーネントのｘｉｄ（コンポーネント識別子）を表す。
ｍｉｍｅ＿ｔｙｐｅ：　これは、このメッセージと関連付けられるデータのMultipurpose internet Mail Extension-type（ＭＩＭＥタイプ）を表し、ゼロ長となり得るヌル終結文字列である。この場合でも、終端の０は存在する。
ｍｅｓｓａｇｅ＿ｄｅｓｃ：これは、メッセージが提示アプリケーションにより提示されたときに割り当てられたこのメッセージの記述を表す。これは、終端の０を含まない最大２５５文字長のヌル終結文字列である。この文字列の長さは０の可能性がある。この場合でも、終端の０は存在する。
【０２４６】
８．３．１４　EM_POST_MESSAGE
EM_POST_MESSAGEメッセージは、アーキテクチャ２００において使用される掲示板に何らかのデータを提示するのに使用される。このメッセージは、サービスグループの変更があるまで続き、任意の実行中のアプリケーションによりアクセスすることができる。また、EM_POST_MESSAGEメッセージは任意の現在実行中のアプリケーションにより削除することができ、完全に信頼できるものであるとは想定されない。メッセージが提示されると、提示したアプリケーションにそのメッセージのメッセージ識別子を通知するために、アプリケーションにメッセージが戻される。このメッセージはアプリケーションによりランチャ３０３に送信される。アプリケーション（すなわち、メッセージを提示したアプリケーション）からのメッセージは以下の情報を含む：
ｍｅｓｓａｇｅ＿ｄｅｓｃ：これはメッセージの記述を表す。これは、終端の０を含まない最大２５５文字長となり得るヌル終結文字列である。記述は０バイトとなる可能性があるが、終端の０をもたなければならない。
ｍｉｍｅ＿ｔｙｐｅ：　これは、提示されるメッセージデータのＭＩＭＥタイプを表す。ＭＩＭＥタイプは必要でないが、終端の０はなければならない。

ｍｅｓｓａｇｅ＿ｄａｔａ：これは、掲示板に提示されるデータを表す。

【０２４７】

また、アプリケーションに戻されるメッセージは以下の情報を含むのが好ましい：

ｍｅｓｓａｇｅ＿ｉｄ：これはメッセージ識別子を表し、このメッセージ識別子によりメッセージの取得又は削除を行なうことができる。

【０２４８】

８．３．１５　EM_GET_MESSAGE

EM_GET_MESSAGEメッセージは、掲示板からメッセージを取得するのに使用される。EM_GET_MESSAGEメッセージは、コンポーネントが取得を希望するメッセージのメッセージ識別子を伴って送信され、その識別子を有するメッセージが存在しないというメッセージ又はエラーを伴ってコンポーネントに戻される。このメッセージはアプリケーション３０４によりランチャ３０３に送信される。

【０２４９】

メッセージの要求時に含まれる情報は以下の通りである：

ｍｅｓｓａｇｅ＿ｉｄ：これは、アプリケーションが取得を希望するメッセージのメッセージ識別子を表す。

ｆｌａｇｓ：　　これはフラグワードである。全ての未使用ビットは０に設定されるべきである。フラグの記述は以下の表２に示される：

【０２５０】

【表２】

表２

フラグ	記述	値
EM_GM_DELETE	メッセージの送信後、それを掲示板から削除する	０ｘ０１

【０２５１】

応答は以下の情報を有する：

ｅｒｒｏｒ：　　エラーが発生すると、これは以下の表３中の値のうちの１つに設定される。

【０２５２】

【表３】

表３

値	記述
EM_GM_NO_ERROR	エラー発生なし。メッセージはメッセージフィールドにある
EM_GM_NO_SUCH_MESSAGE	掲示板にはそのメッセージ識別子を有するメッセージはない

【０２５３】

ｍｅｓｓａｇｅ＿ｉｄ：これは取得されたメッセージのメッセージ識別子を表す。

ｍｉｍｅ＿ｔｙｐｅ：　これは、取得されたメッセージのＭＩＭＥタイプを表す。これはヌル終結文字列である。このメッセージと関連付けられたＭＩＭＥタイプが存在しない場合、文字列はゼロ長であるが、終端の０は存在する。

ｍｅｓｓａｇｅ：　　エラーの発生がない場合、このフィールドは掲示板に提示されるデータを含むことになる。長さはヘッダ中のｄａｔａＬｅｎｇｔｈの値－エラーフィールドのサイズにより判定される。

【０２５４】

８．３．１６　EM_DELETE_MESSAGE

EM_DELETE_MESSAGEメッセージは、掲示板からメッセージを削除するのに使用される。存在しないメッセージを削除するのはエラーではない。このメッセージはフロントアプリケ

ーションによりランチャ３０３に送信される。EM_DELETE_MESSAGEメッセージは以下の情報を含むのが好ましい：
ｍｅｓｓａｇｅ＿ｉｄ：これは、削除対象のメッセージのメッセージ識別子を表す。
【０２５５】
８．３．１７　ユーザインタフェースカードリーダメッセージ
ユーザインタフェースカードリーダメッセージはリモートリーダ１により生成され、イベントマネージャプロトコルに従うように、イベントマネージャ３０１によりカプセル化される。リモートリーダ１により生成されるメッセージは３種類ある。「単純な」メッセージ、「移動」メッセージ、及び「押下／押下解除」メッセージである。移動メッセージは座標が付加された単純なメッセージであり、押下／押下解除メッセージはデータ及び座標が付加された単純なメッセージである。
【０２５６】
８．３．１７．１　単純なメッセージ
以下のメッセージが単純なメッセージである：
・EM_READER_INSERT
・EM_READER_REMOVE
・EM_READER_BADCARD
・EM_READER_LOW_BATT
これらの単純なメッセージは以下の情報を含むのが好ましい。
ｉｄ：　これは、リモートリーダ１により送信された区別用識別子を表し、ＢＡＤＣＡＲＤメッセージの場合は意味をもたない。
【０２５７】
８．３．１７．２　移動メッセージ
EM_READER_MOVEメッセージは以下の情報を含むのが好ましい：
ｉｄ：　これは、リモートリーダ１により送信された区別用識別子を表し、カードなしメッセージの場合、全てが０に設定される。
Ｘ：　これは、ｘ値を表す。
Ｙ：　これは、ｙ値を表す。
【０２５８】
８．３．１７．３　押下／押下解除メッセージ
EM_READER_PRESSメッセージ及びEM_READER_RELEASEメッセージは以下の情報を含むのが好ましい：
ｉｄ：　これは、リモートリーダ１により送信された区別用識別子を表す。
Ｘ：　これは、ｘ値を表す。
Ｙ：　これは、ｙ値を表す。
ｄａｔａ：これは、押下又は押下解除と関連付けられる（ユーザインタフェース要素データと関連付けられる）任意のデータを表す。
【０２５９】
８．４　手順
以下の段落は、アーキテクチャ２００の各プロセスコンポーネントが後続するメインの手順を説明する。
【０２６０】
８．４．１　新規のアプリケーションの開始
図３２は、新規のアプリケーションを開始する方法３３００の詳細を示すフローチャートであり、ランチャ３０３が新規のアプリケーションを開始するたびに実行される。方法３３００は、コンピュータで実現する場合にはＣＰＵ２０５により実行することができる。また、セットトップボックスで実現する場合にはＣＰＵ４３０５により実行することができる。方法３３００は第１のステップ３３０１で開始される。ステップ３３０１において、ランチャ３０３はサービス識別子をＵＲＬへと変換するためのマッピングを実行する。次のステップ３３０３において、ランチャ３０３はイベントマネージャのホスト名及びポ

ート番号をランチャ３０３に通知するアプリケーションを取り込み、開始させる。方法３３００は次のステップ３３０５へと続く。ステップ３３０５において、ランチャ３０３はイベントマネージャ３０１に開始するアプリケーションのｘｉｄを通知するEM_APP_STARTINGメッセージをイベントマネージャ３０１に送信する。次のステップ３３０７において、新規のアプリケーションはイベントマネージャ３０１に接続し、ランチャ３０３にEM_APP_REGISTERメッセージを送信する。更に、通常は、新規のアプリケーションへのフォーカス変更が存在する。

【０２６１】

８．４．２　アプリケーションの終了

図３３は、ソフトウェアアーキテクチャ２００を組み込むシステム６００においてアプリケーションを終了する方法３４００を示すフローチャートである。方法３４００は、コンピュータで実現する場合にはＣＰＵ２０５により実行することができる。また、セットトップボックスで実現する場合にはＣＰＵ４３０５により実行することができる。この方法はランチャ３０３が実行中のアプリケーションを終了させるたびに使用される。方法３４００はステップ３４０１で開始される。ステップ３４０１において、ランチャ３０３は実行中のアプリケーションにEM_EXIT_NOWメッセージを送信する。ランチャ３０３はこの時点でタイムアウトを設定し、作業を片付けて終了する機会をアプリケーションに与える。次のステップ３４０３において、実行中のアプリケーションは片付け作業を行ない、終了する。これに対し、アプリケーションがEM_EXIT_NOWメッセージを無視すると、ランチャ３０３がタイムアウトになり、アプリケーションを強制的に終了させる。ステップ３４０５において、ランチャ３０３はイベントマネージャ３０１にEM_APP_DYINGメッセージを送信してアプリケーションの終了を通知すると共に、あらゆる待ち状態のデータを破棄し、アプリケーションへの接続がオープンのままである場合には接続を終了すべきであると通知する。方法３４００は終了する。

【０２６２】

８．４．３　永続的なアプリケーションのセッションの終了

図３４は、ソフトウェアアーキテクチャ２００を組み込むシステム６００上で永続的なアプリケーションの現在のセッションを終了する方法３５００を示すフローチャートである。方法３５００は、コンピュータで実現する場合にはＣＰＵ２０５により実行することができる。また、セットトップボックスで実現する場合にはＣＰＵ４３０５により実行することができる。方法３５００はアプリケーションの終了と類似しているが、アプリケーションが実際に終了する訳ではない。方法３５００はステップ３５０１で開始される。ステップ３５０１において、ランチャ３０３は永続的なアプリケーションにEM_CLOSEメッセージを送信する。次のステップ３５０３において、永続的なアプリケーションは初期状態にリセットし、方法３５００は終了する。これは、外部のサーバへの接続の終了、デフォルトのウェブページのロードなどを伴っても良い。永続的なアプリケーションが受信する次のEM_GAINING_FOCUSイベントは、新規のセッションの開始であると想定される。

【０２６３】

８．４．４　フォーカス変更

図３５は、ソフトウェアアーキテクチャ２００を組み込むシステム６００上でフォーカス変更を実行する方法３６００を示すフローチャートである。方法３６００は、コンピュータで実現する場合にはＣＰＵ２０５により実行することができる。また、セットトップボックスで実現する場合にはＣＰＵ４３０５により実行することができる。方法３６００は、アプリケーションに入力／出力フォーカスを受け取る又は失うことを通知するのに使用される。この通知はアプリケーションが終了する信号ではない。第１のステップ３６０１において、ランチャ３０３は現在入力／出力フォーカスを有するアプリケーションを変更する決定を行ない、通常はカード変更に基づいて入力フォーカスを受け取る予定のアプリケーションにEM_GAINING_FOCUSイベントを送信する。このイベントの送信はイベントマネージャ３０１により使用され、所定の条件に基づいてどのアプリケーションが入力／出力フォーカスを受け取るべきかが決定される。ステップ３６０３において、ランチャ３０３

は前のフロントアプリケーションにEM_LOSING_FOCUSイベントを送信し、方法３６００は終了する。このメッセージの重要度は低く、現在のフロントアプリケーションが変わらないときには送信されないが、EM_GAINING_FOCUSは必要である（すなわち、EM_GAINING_FOCUSイベントがブラウザコントローラ４０２にベースＵＲＬを通知するのに使用されるブラウザコントローラの場合）。

【０２６４】

８．４．５　メッセージの受け渡し(Message Passing)

アーキテクチャ２００によりサポートされるアプリケーション間のメッセージの受け渡しには２種類ある。現在のサービスグループと同程度に永続的な掲示板を介する方法と２つのコンポーネントが後述するように直接相互に通信を行なう直接メッセージ方式である。

【０２６５】

８．４．５．１　掲示板(Message Board)

アーキテクチャ２００の１つのコンポーネント（通常は、ランチャ３０３）が掲示板を維持するが、イベントマネージャ３０１にはどのコンポーネントがこの維持を行なうかが分かっている。掲示板は、掲示板を管理するプロセスコンポーネントにより識別子として３２ビット符号無し数値を割り当てられたメッセージのリストから形成される。各メッセージは、テキスト記述、メッセージデータに対するオプションのＭＩＭＥタイプ、及びメッセージ自体から形成される。アプリケーションは、EM_LIST_MESSAGESメッセージを送信することによって、現在掲示板にある全てのメッセージのリストを要求することができる。この要求に対して、関連するメッセージ識別子を伴って現在掲示板にある全てのメッセージのテキスト記述が戻されることになる。アプリケーションは、必要とするメッセージのメッセージ識別子と共にEM_GET_MESSAGEを送信することによって特定のメッセージを要求することができる。掲示板のリストの取得から実際のメッセージの要求までの間にメッセージが削除される可能性もある。EM_GET_MESSAGEメッセージ応答のエラーフィールドはこれを示すように構成される。

【０２６６】

８．４．５．２　直接通信

２つのアプリケーションは、直接通信を使用することによって相互に任意のデータを直接送信することができる。これは、EM_SEND_MESSAGEメッセージを使用することによって一方のアプリケーションが他方のアプリケーションにデータを送信することにより実行される。２つのアプリケーションでは、このメッセージに対するデータ形式が一致する必要があり、バイト配列の問題を考慮しなければならない。相手側のアプリケーションのコンポーネント識別子を取得するために、アプリケーションはEM_LIST_APPSメッセージを送信することによって全ての実行中のアプリケーションのリストが送信されるように要求することができる。このメッセージは、現在実行中の全ての公開されているアプリケーションのリストを戻す。

【０２６７】

８．５　リーダメッセージ

本節では、各EM_READER_* メッセージを転送するためにイベントマネージャ３０１により使用される規則の概要を説明する。以下のメッセージは、どのアプリケーションが現在フォーカスをもっているかに関わらず、常にランチャ３０３に送信される。

・EM_READER_INSERT
・EM_READER_REMOVE
・EM_READER_BADCARD
・EM_READER_LOW-BATT

対応するフィールド１１０６中に現在のフロントアプリケーションと同じサービス識別子を有するカード１０からのものである場合、以下のメッセージは現在のフロントアプリケーションに送信される。サービス固有識別子はこの比較では考慮されない。サービス識別子が現在のフロントアプリケーションと異なる場合、あるいは、区別用識別子がＮＯ＿ＣＡＲＤの現在値（すなわち、全て０）である場合、メッセージは前述のようにランチャ３

０３に送信される。
・EM_READER_PRESS
・EM_READER_RELEASE
・EM_READER_MOVE
【０２６８】
８．６　メッセージ送信に対する規制
システム６００の安全性及び安定性を改善するためには、メッセージの送信に対して規制が課されるのが好ましい。この規則に違反するメッセージはイベントマネージャ３０１により破棄されることになる。
【０２６９】
８．６．１　全てのコンポーネントに対する規制
リモートリーダ１以外のコンポーネントにはEM_READER_*メッセージを送信する許可が与えられない。
【０２７０】
８．６．２　イベントマネージャに対する規制
イベントマネージャ３０１は規則の実施者であり、そのように任意の必要なメッセージを送信することができる。イベントマネージャ３０１は、EM_KILL_LAUNCHERメッセージ及びEM_NEW_LAUNCHERメッセージのみを生成するだけで良いように構成されるが、メッセージをコピーし、そのコピーを目的のコンポーネントでないプロセスコンポーネントに送信することができる。また、イベントマネージャ３０１はコンポーネント間の全ての送信を扱う。
【０２７１】
８．６．３　ランチャに対する規制
ランチャ３０３は、アーキテクチャ２００の全てのコンポーネント３０１から３０６にメッセージを送信する。ランチャ３０３が送信することができないメッセージは以下の通りである：
・EM_KILL_LAUNCHER
・EM_NEW_LAUNCHER
【０２７２】
８．６．４　アプリケーションに対する規制
アプリケーションが他のアプリケーション（ランチャ３０３を含む）に送信するのは以下のメッセージのみである：
・EM_APP_REGISTER
・EM_SEND_MESSAGE
・EM_LIST_APPS
・EM_POST_MESSAGE
・EM_GET_MESSAGE
・EM_DELETE_MESSAGE
・EM_LIST_MESSAGES
【０２７３】
８．７　コンポーネント手順リスト
本節では、アーキテクチャ２００の各コンポーネントが関係する機能を列挙する。
【０２７４】
８．７．１　イベントマネージャ
イベントマネージャ３０１は以下の手順に直接関与する：
・システム初期化
・システム起動
・新規のアプリケーションの開始
・アプリケーションの終了
・フォーカス変更

・メッセージの受け渡し
・リーダメッセージ
【０２７５】
８．７．２　ランチャ
ランチャ３０３は以下の手順に関与する：
・システム初期化
・システム起動
・新規のアプリケーションの開始
・アプリケーションの終了
・フォーカス変更
・メッセージの受け渡し（場合による）
・リーダメッセージ（場合による）
【０２７６】
８．７．３　アプリケーション
アプリケーション３０４は以下の手順に関与する：
・新規のアプリケーションの開始
・アプリケーションの終了
・アプリケーションが永続的な場合のセッションの終了
・フォーカス変更
・メッセージの受け渡し
・リーダメッセージ（場合による）
【０２７７】
９．０　Ｉ／Ｏデーモン
Ｉ／Ｏデーモン３００は、リモートリーダ１からイベントマネージャ３０１に送信され、双方向プロトコルの場合は逆方向にも送信されるデータを転送する役割を担う。Ｉ／Ｏデーモン３００は、システム６００のハードウェアから直接に、あるいは、ＩＲリンク又は標準シリアルハードウェア接続などのリモートリーダ１とのインタフェースであるオペレーティングシステムドライバを介して読むことができるように構成される。また、Ｉ／Ｏデーモン３００には、ＴＣＰ／ＩＰポートで聞き取りながらイベントマネージャ３０１が接続するのを待つことが要求される。ここで、Ｉ／Ｏデーモン３００はＴＣＰ／ＩＰストリーム中にカプセル化した状態でデータをリモートリーダ１からイベントマネージャ３０１に送信する。
【０２７８】
Ｉ／Ｏデーモン３００は、リモートリーダ１のデータをイベントマネージャ３０１に送信する場合及びＩ／Ｏデーモン３００とリモートリーダ１との間でのオプションの双方向プロトコル構成において逆方向の送信を行なう場合を除き、システム６００のその他の部分とは通信を行なわない。
【０２７９】
Ｉ／Ｏデーモン３００の機能はシステム６００に存在しなければならないが、Ｉ／Ｏデーモン３００は独立したコンポーネントである必要はない。例えば、イベントマネージャ３０１がリモートリーダ１とインタフェースをとるのに使用されるハードウェアと同じマシン上で実行されている場合、Ｉ／Ｏデーモン３００はイベントマネージャ３０１へと統合することができる。
【０２８０】
Ｉ／Ｏデーモン３００は、システム６００のその他の部分がリモートで実行されている場合、最小限のハードウェア上で稼働するように構成される。
【０２８１】
９．１　要求事項
９．１．１　一般的な要求事項
Ｉ／Ｏデーモン３００が実現されるプラットフォームは、リモートリーダ１から信号を受

信（オプションとして、信号を送信）できるように構成されなければならない。また、プラットフォームは、システムのその他の部分（すなわち、イベントマネージャ（ＥＭ）３０１）と通信を行なうために、ＴＣＰ／ＩＰスタック又はその他の信頼性の高い通信方法を実現するのが好ましい。Ｉ／Ｏデーモン３００は、多重化Ｉ／Ｏを行なうように要求される可能性があり、アーキテクチャ２００のＩ／Ｏシステムは多重化Ｉ／Ｏをサポートするように構成されるのが好ましい。アーキテクチャ２００は、Ｉ／Ｏデーモン３００が聞き取りを行なうであろうポートを、例えば、コマンドライン引数として割り当てるように構成されるのが好ましい。

【０２８２】

９．１．２　内部の要求事項

Ｉ／Ｏデーモン３００は、リモートリーダ１により使用されるプロトコルを理解する必要はない。Ｉ／Ｏデーモン３００に要求されるのは、受信する全てのデータを任意の聞き取り中のＥＭ（イベントマネージャ）に転送することのみである。Ｉ／Ｏデーモン３００は、通信リンクのトランスポートプロトコルにより（すなわち、エラー訂正コード又はそれに類するものを介して）サポートされない限り、リモートリーダ１からの送信のエラーを訂正する必要がない。使用中のトランスポートプロトコルがエラー検出をサポートするが、エラー訂正はサポートしない場合、エラーチェックに合格しないデータをイベントマネージャ３０１に渡すことができる。

【０２８３】

９．１．３　外部インタフェースの要求事項

Ｉ／Ｏデーモン３００は、１つ以上のＴＣＰ／ＩＰ接続を受け入れることができるのが好ましい。イベントマネージャ３０１に送信されるデータストリームは、リモートリーダ１により送信されるデータの内容である。使用される通信プロトコルの一部として送信される全てのヘッダ情報及びフッタ情報は除去されるのが好ましい。バイト配列はビッグエンディアンである。アーキテクチャ２００の通信方法が使用不可能になる（例えば、エラー発生が原因）場合、Ｉ／Ｏデーモン３００はエラー状態が起こると即座に全ての接続を終了する。

【０２８４】

９．２　外部インタフェース

Ｉ／Ｏデーモン３００の外部インタフェース（不図示）は、最小限のハードウェア上で実行可能であるように意図的に簡単なものとなっている。Ｉ／Ｏデーモン３００は以下のように構成されるのが好ましい。

【０２８５】

９．２．１　起動手順

Ｉ／Ｏデーモン３００は、何らかの方法で、例えば、コマンドライン引数により指定されるＴＣＰ／ＩＰポートで聞き取りを行なう。Ｉ／Ｏデーモン３００にＴＣＰ／ＩＰポートを通知する厳密な方法は実現例に依存する。リモートリーダ１と通信するために使用される通信ハードウェアは必要に応じて初期化され、リモートリーダ１から送信されるデータを読む方法はデータを受信できる状態にあるように構成される。接続を待つ間、Ｉ／Ｏデーモン３００はリモートリーダ１により送信されるデータを消費するので、接続されたときには新規のデータのみが送信される。この新規のデータはメッセージ境界で開始する必要がない。

【０２８６】

９．２．２　イベントマネージャからの接続

ＴＣＰ／ＩＰポートで接続が成功する場合、Ｉ／Ｏデーモン３００はその接続を受け入れ、接続を下ってリモートリーダ１から受信されるデータの送信を開始するように構成される。Ｉ／Ｏデーモン３００が既にイベントマネージャ（ＥＭ）３０１に接続されている場合、Ｉ／Ｏデーモン３００には２つの選択肢がある。まず、Ｉ／Ｏデーモンは接続を受け入れ、全ての現在接続されているイベントマネージャを下って全てのデータを送信することができる。この選択肢はシステムデバッギングのために提供される。第２の方法は第２

の接続を拒否し、既に接続されているＥＭへのデータの送信を継続する方法である。ストリームの暗号化は、ポートトンネリングなどの他の何らかの方法によって外部で処理することができる。

【０２８７】

９．２．３　イベントマネージャからの接続の終了

いつでもイベントマネージャ３０１への接続が終了する場合には、Ｉ／Ｏデーモン３００はイベントマネージャ３０１に送信されるのを待っているリモートリーダ１からのデータを破棄するように構成される。これが接続されている唯一のイベントマネージャである場合、Ｉ／Ｏデーモン３００は初期の起動状態に戻るように構成される。これにより、Ｉ／Ｏデーモン３００はリモートリーダ１により送信されるデータを消費しながら接続を待つ。

【０２８８】

９．２．４　回復不能エラーの出現

対処不可能であり、Ｉ／Ｏデーモン３００の終了を引き起こすであろうエラーをＩ／Ｏデーモン３００が検出する場合、Ｉ／Ｏデーモン３００はＥＭへの全ての接続を終了し、ＥＭにＩ／Ｏデーモン３００がエラーを検出したことを通知するように構成される。エラーの例としては、リモートリーダ１と通信するのに使用されるハードウェアが使用不可能になる場合、あるいは、Ｉ／Ｏデーモン３００が終了を引き起こすであろう信号を受信する場合が含まれる。Ｉ／Ｏデーモン３００はエラーに遭遇すると即座に全ての接続を終了するように構成される。

【０２８９】

１０．０　ランチャ

ランチャ３０３は、許可されたアプリケーション及び基本アプリケーションの構成規則などのサイト特有の規則を実施するプロセスコンポーネントである。ランチャ３０３は、システムアーキテクチャ２００の他のコンポーネントプロセス３００、３０１、３０４、３０５、及び３０６が、一般家庭のセットトップボックス６０１から非常に特殊なアプリケーション（例えば、現金自動預払機）に至る広範囲のアプリケーションにおいて使用されるのを許可する。ランチャ３０３は、ネットワークごと又は施設ごとに個別に作成することができる。

【０２９０】

ランチャ３０３は特権を有するように構成される。例えば、ランチャ３０３は、システム６００が起動する際にイベントマネージャ３０１に接続する最初のコンポーネントになるように構成することができる。更に、ランチャ３０３は、リモートリーダ１により送信される全ての「ＬＯＷ＿ＢＡＴＴ」メッセージ、「ＢＡＤＣＡＲＤ」メッセージ、「ＩＮＳＥＲＴ」メッセージ、及び「ＲＥＭＯＶＥ」メッセージを受信すると共に、任意の時点においてフロントアプリケーションが関連付けられるスマートカード１０以外のカードから送信される全ての「ＰＲＥＳＳ」メッセージ、「ＲＥＬＥＡＳＥ」メッセージ、及び「ＭＯＶＥ」メッセージを受信する。また、ランチャ３０３は特殊な「ＮＯ＿ＣＡＲＤ」区別用識別子を有するＰＲＥＳＳメッセージ、ＲＥＬＥＡＳＥメッセージ、及びＭＯＶＥメッセージも受信する。ランチャ３０３は、更に、EM_GAINING_FOCUSイベント及びEM_LOSING_FOCUSイベントを介してどのアプリケーションをフロントアプリケーションにするかに関する制御権を有する。

【０２９１】

ランチャ３０３は、アプリケーションがいつ開始され、いつ終了させられる必要があるかを決定するように構成される。また、ランチャ３０３は、常にそうである訳ではないが、アプリケーションを開始／終了するのに使用される。この役割は、例えば、アプリケーション３０４がアーキテクチャ２００のその他のコンポーネントとは別のマシンで実行される場合に、ランチャ３０３の指示に応じて別のアプリケーションが引き受けることができる。

【０２９２】

イベントが現在のフロントアプリケーションではなく、ランチャ３０３に送信されることにより、ランチャ３０３はどのアプリケーションが任意の時点において実行されるべきかを決定することができる。アプリケーションを強制的に終了するように構成されることは、ランチャ３０３がどのアプリケーションが現在実行中であるべきかを強制できることを意味する。また、ランチャ３０３には、アプリケーションをいつ開始／終了するかをイベントマネージャ３０１に通知することも要求される。

【０２９３】

図３６は、ランチャ３０３により実行される方法３７００の概要を示すフローチャートである。方法３７００は、コンピュータで実現する場合にはＣＰＵ２０５により実行することができる。また、セットトップボックスで実現する場合にはＣＰＵ４３０５により実行することができ、リモートサーバのＣＰＵによっても実行することができる。方法３７００は第１のステップ３７０１で開始される。ステップ３７０１において、ランチャ３０３はイベントマネージャ３０１に接続し、次のステップ３７０２へと続く。ステップ３７０２において、永続的なアプリケーションが開始される。次のステップ３７０３において、ランチャ３０３はイベントを待ち、イベントが受信されるとステップ３７０５へと進む。ステップ３７０５において、イベントがＮＯ＿ＣＡＲＤ識別子である場合、プロセスはステップ３７０７へと進む。ＮＯ＿ＣＡＲＤ識別子でない場合、プロセスはステップ３７０９へと進む。ステップ３７０７において、ランチャ３０３はＮＯ＿ＣＡＲＤ識別子に応じて所定のシステム特有の機能（例えば、ディスプレイ１０１にメッセージを表示）を実行し、方法３７００はステップ３７０３へと戻る。

【０２９４】

決定ステップ３７０５におけるイベントがＮＯ＿ＣＡＲＤ識別子でないと判定されると、別の決定ステップ３７０９へと入り、イベントがＰＲＥＳＳ、ＲＥＬＥＡＳＥ、ＲＥＭＯＶＥ、又はＭＯＶＥであるか否かが判定される。この決定ステップ３７０９が「ｙｅｓ」を戻す場合、すなわち、イベントが前述のイベントのうちの１つである場合、方法３７００はステップ３８００へと進む。イベントが前述のイベントのうちの１つでない場合、方法３７００は更なる決定ステップ３７１３へと進む。ステップ３８００において、ランチャ３０３はフローチャートの図３７を参照して説明したプロセスステップに従ってアプリケーションを変更する。

【０２９５】

ステップ３７０９におけるイベントがＰＲＥＳＳイベント、ＲＥＬＥＡＳＥイベント、ＲＥＭＯＶＥイベント、及びＭＯＶＥイベントのうちの１つでない場合、決定ステップ３７１３に入る。この決定ステップ３７１３はＢＡＤＣＡＲＤイベント又はＬＯＷ＿ＢＡＴＴイベントに関する判定を行なう。ステップ３７１３において、イベントがＢＡＤＣＡＲＤイベント又はＬＯＷ＿ＢＡＴＴイベントである場合、方法３７００はステップ３７１５へと進み、いずれのイベントでもない場合、方法３７００はステップ３７１７へと進む。ステップ３７１５において、ランチャ３０３はユーザに対して発生したイベントに関するフィードバック（例えば、ＬＯＷ＿ＢＡＴＴイベントが判定される場合はディスプレイ１０１上に「バッテリ低下」メッセージを表示し、ＢＡＤＣＡＲＤイベントの判定の際には「不正なカード」を表示）を行ない、方法３７００はステップ３７０３へと戻る。決定ステップ３７１３におけるイベントがＢＡＤＣＡＲＤイベントでもＬＯＷ＿ＢＡＴＴイベントでもない場合、ステップ３７１７に入る。

【０２９６】

ステップ３７１７において、イベントがＡＰＰ＿ＲＥＧＩＳＴＥＲである場合、方法３７００はステップ３９００の「アプリケーション登録」へと進む。ＡＰＰ＿ＲＥＧＩＳＴＥＲでない場合、方法３７００はステップ３７２５へと進む。ステップ３９００において、図３８を参照してここで説明するように（すなわち、第８．３．４節を参照して先に説明したように、アプリケーションが他のコンポーネント３０１、３０２、及び３０６にメッセージの受信が可能な状態になったことを通知する）アプリケーションが登録され、方法３７００はステップ３７０３へと戻る。ステップ３９００に従ってアプリケーションを登

録する方法については、図３８のフローチャートを参照して詳細に後述する。ステップ３７２５においてイベントは破棄され、方法３７００はステップ３７０３へと戻る。

【０２９７】

図３７は、アプリケーションを変更する方法３８００を示すフローチャートであり、この方法はランチャ３０３により実行される。方法３８００は、コンピュータで実現する場合にはＣＰＵ２０５により実行することができる。また、セットトップボックスで実現する場合にはＣＰＵ４３０５により実行することができ、リモートサーバのＣＰＵによっても実行することができる。方法３８００はステップ３８１７で開始される。ステップ３８１７において、ランチャ３０３によりＲＥＭＯＶＥメッセージが受信されると、プロセスは直接ステップ３８１３へと進む。ＲＥＭＯＶＥメッセージが受信されない場合、方法３８００は決定ステップ３８０１へと続く。決定ステップ３８０１において、イベントにより表されるサービスが登録されているアプリケーションと関連付けられている場合、方法３８００は直接ステップ３８１９へと進む。アプリケーションと関連付けられていない場合、方法３８００はステップ３８０３へと続く。ステップ３８０３において、新規のアプリケーションの位置及び／又は名前と新規のアプリケーションと関連付けられる初期データとを判定するために、サービス識別子参照が実行される。例えば、初期データはブラウザ４０３へとロードされるＵＲＬであっても、あるいは、メディアプレーヤアプリケーションへとロードされるメディアファイルであっても良い。次のステップ３８０５において、アプリケーションが既に実行中の場合、方法３８００はステップ３８１９へと進む。アプリケーションが実行中でない場合、方法３８００はステップ３８０９へと進む。ステップ３８０９において、新規のアプリケーションがアプリケーション３０４から得られる。次のステップ３８１１において、新規のアプリケーションがフロントアプリケーションとして開始される。ステップ３８１２において、イベントマネージャ３０１にはこの新規のフロントアプリケーションのコンポーネント識別子（Ｘｉｄ）が通知される。

【０２９８】

イベントにより表されるサービスが登録されているアプリケーションと関連付けられている場合（ステップ３８０１から）、あるいは、アプリケーションが既に実行中の場合、決定ステップ３８１９に入る。ステップ３８１９において、ＩＮＳＥＲＴメッセージがランチャ３０３により受信されたと判定される場合、方法３８００は終了する。判定されない場合、方法３８００はステップ３８０７へと進む。ステップ３８０７において、新規のアプリケーションがすぐに変更状態になることを示すＧＡＩＮＩＮＧ＿ＦＯＣＵＳイベントが新規のアプリケーションに送信される。新規のアプリケーションにＧＡＩＮＩＮＧ＿ＦＯＣＵＳイベントが送信された後、あるいは、決定ステップ３８１７でＲＥＭＯＶＥイベントが検知された結果、制御が決定ステップ３８１３に渡される。ステップ３８１３において、現在のフロントアプリケーションが存在するか否かが判定される。既にフロントアプリケーションが存在しない場合、方法３８００は終了する。フロントアプリケーションが存在する場合、前のフロントアプリケーションにＬＯＳＩＮＧ＿ＦＯＣＵＳイベントが送信され、前のフロントアプリケーションは方法３８００が終了する前に当座のタスクを終了することができる。

【０２９９】

図３８は、新規のアプリケーションを登録する方法又はプロセス３９００を示すフローチャートであり、この方法はランチャ３０３により実行される。方法３９００は、コンピュータで実現する場合にはＣＰＵ２０５により実行することができる。また、セットトップボックスで実現する場合にはＣＰＵ４３０５により実行することができ、リモートサーバのＣＰＵによっても実行することができる。プロセス３９００はステップ３９０１で開始される。ステップ３９０１において、アプリケーションを含み、図３６のステップ３９００を参照して言及される新規のサービスグループリストが生成される。次のステップ３９０３において、このアプリケーションにＧＡＩＮＩＮＧ＿ＦＯＣＵＳイベントが送信される。ステップ３９０５において、いずれのアプリケーションも新規のサービスグループの一部でなく、永続的でもない場合、方法３９００はステップ３９０７へと進む。サービス

グループの一部であるか、あるいは、永続的である場合、方法３９００は終了する。ステップ３９０７において、サービスグループの一部でないアプリケーションにはＥＸＩＴ＿ＮＯＷイベントが送信され、方法３９００は次のステップ３９０８へと進む。ステップ３９０８において、イベントマネージャ３０１は新規のサービスグループの一部でないアプリケーションが終了したことを通知される。その後、方法３９００は終了する。

【０３００】

図３９は、ランチャ３０３からイベントを受信するときにアプリケーションにより実行されるプロセスステップ４０００を示すフローチャートである。方法４０００は、コンピュータで実現する場合にはＣＰＵ２０５により実行することができる。また、セットトップボックスで実現する場合にはＣＰＵ４３０５により実行することができ、リモートサーバのＣＰＵによっても実行することができる。方法４０００はステップ４００１で開始される。ステップ４００１において、ランチャ３０３はイベントマネージャ３０１に接続し、方法４０００はステップ４００２へと進む。ステップ４００２において、ランチャ３０３にＡＰＰ＿ＲＥＧＩＳＴＥＲメッセージを送信することによってアプリケーションが登録される。図３９に示すフローチャートを次のステップ４００３までたどると、アプリケーションはイベントを待ち、イベントが受信されるとプロセスはステップ４００５へと進む。ステップ４００５において、イベントがＧＡＩＮＩＮＧ＿ＦＯＣＵＳイベントである場合、方法４０００はステップ４００７へと進む。ＧＡＩＮＩＮＧ＿ＦＯＣＵＳイベントでない場合、方法４０００はステップ４００９へと進む。ステップ４００７において、オプションとして区別用識別子を使用すると共にオプションとしてＧＡＩＮＩＮＧ＿ＦＯＣＵＳイベントのデータフィールドを使用することによって、必要に応じてアプリケーションが初期化される。初期化に使用されるこのデータフィールドは、ロード対象のＵＲＬ、ロード対象のファイル名などを含んでも良い。制御はステップ４００３におけるイベント待ちに戻る。

【０３０１】

ステップ４００９において、イベントがＰＲＥＳＳイベント、ＲＥＬＥＡＳＥイベント、又はＭＯＶＥイベントである場合、方法４０００はステップ４０１１へと進む。いずれのイベントでもない場合、方法４０００はステップ４０１３へと進む。ステップ４０１１において、イベントに応じてアプリケーション特有の動作が実行される。アプリケーション特有の動作は、イベントからのデータ（すなわち、カード１０上の印と関連付けられるデータ（例えば、ＵＲＬ、キャラクタ又はビデオの名前））、Ｘ／Ｙ位置又は区別用識別子、あるいは、これらの任意の組み合わせを使用して実行される。

【０３０２】

アプリケーション特有の動作は、通常、カード１０上の印と関連付けられる。例えば、印は特定のＵＲＬと関連付けることが可能であり、印が押下されるときにＵＲＬがアクセスされても良い。このため、例えば、コンピュータ１００又はＳＴＢ６０１はＵＲＬにより指定されたウェブページから所望のプログラムをダウンロードすることができ、カードユーザはシステム６００からサービス（すなわち、プログラムダウンロード）を受け取ることができる。更に、印は特定のメモリアドレスと関連付けることが可能であり、印が押下されるときにそのメモリアドレスにデータを格納するのにアドレスを使用することができる。従って、例えば、コンピュータ１００又はＳＴＢ６０１はメモリアドレスにより指定されたメモリ又はネットワーク上のファイルサーバから所望の画像データをダウンロードすることができ、カード１０のユーザはシステム６００からサービス（例えば、画像データダウンロード）を受け取ることができる。ステップ４０１１の後、図３９に示すように、方法４０００はステップ４００３へと戻る。

【０３０３】

上述の図３９のフローチャートによると、対応する決定ステップ４００５又は４００９において、イベントがＧＡＩＮＩＮＧ＿ＦＯＣＵＳイベント、ＰＲＥＳＳイベント、ＲＥＬＥＡＳＥイベント、及びＭＯＶＥイベントのうちのいずれかであると判定されない場合、プロセスステップ４０００はフィルタを通過してステップ４１０３へと至る。ステップ４

０１３において、イベントがＬＯＳＩＮＧ＿ＦＯＣＵＳイベントである場合、ステップ４０１３において、方法４０００はステップ４０１５へと進む。ＬＯＳＩＮＧ＿ＦＯＣＵＳイベントでない場合、方法４０００は決定ステップ４０１７へと進む。ステップ４０１５において、アプリケーションは非アクティブ状態へと戻り、方法４０００はステップ４００３へと戻る。ステップ４０１７において、イベントがＥＸＩＴ＿ＮＯＷイベントである場合、方法４０００は終了する。ＥＸＩＴ＿ＮＯＷイベントでない場合、方法４０００はステップ４０１９へと進む。ステップ４０１９において、イベントは無視され、方法４０００はステップ４００３へと戻る。

【０３０４】

図４０は、ランチャ３０３からイベントを受信するときにブラウザコントローラ４０２アプリケーションにより実行される方法４１００を示すフローチャートである。方法４１００は、コンピュータで実現する場合にはＣＰＵ２０５により実行することができる。また、セットトップボックスで実現する場合にはＣＰＵ４３０５により実行することができ、リモートサーバのＣＰＵによっても実行することができる。方法４１００はステップ４１０１で開始される。ステップ４１０１において、ブラウザアプリケーションはランチャ３０３にＡＰＰ＿ＲＥＧＩＳＴＥＲメッセージを送信する。次のステップ４１０３において、ブラウザアプリケーションはイベントを待ち、イベントが受信されると方法４１００はステップ４１０５へと進む。ステップ４１０５において、イベントがＧＡＩＮＩＮＧ＿ＦＯＣＵＳイベントである場合、方法４１００はステップ４１０７へと進む。ＧＡＩＮＩＮＧ＿ＦＯＣＵＳイベントでない場合、方法４１００はステップ４１０９へと進む。ステップ４１０７において、アプリケーションは必要に応じて初期化される。例えば、アプリケーションはＧＡＩＮＩＮＧ＿ＦＯＣＵＳメッセージのデータフィールドを読み、データフィールドがＵＲＬを表す場合、そのＵＲＬをロードする。初期化は、初期ＵＲＬをブラウザアプリケーション４０３へとロードし、ＵＲＬのベースを格納することによってブラウザコントローラ４０２上で実行される。方法４１００は次のステップ４１２１へと続く。ステップ４１２１において、区別用識別子がイベントから判定される。次のステップ４１２３において、現在のトップレベル文書中で引数としての区別用識別子１１１０と共にＪａｖａＳｃｒｉｐｔコールバック関数（Ｎｏｔｉｆｙ＿Ｃａｒｄ＿ＩＤとして知られるのが好ましい）が呼び出され、方法４１００はステップ４１０３へと戻る。

【０３０５】

ステップ４１０９において、イベントがＰＲＥＳＳイベント、ＲＥＬＥＡＳＥイベント、又はＭＯＶＥイベントである場合、方法４１００はステップ４２００へと進む。いずれのイベントでもない場合、方法４１００はステップ４１１３へと進む。ステップ４２００において、イベントに応じてブラウザアプリケーション特有の動作が実行される。ブラウザアプリケーション特有の動作については、図４１のフローチャートを参照して詳細に後述する。ステップ４２００の後、方法４１００はステップ４１０３へと戻る。

【０３０６】

ステップ４１１３において、イベントがＬＯＳＩＮＧ＿ＦＯＣＵＳイベントである場合、方法４１００はステップ４１１５へと進む。ＬＯＳＩＮＧ＿ＦＯＣＵＳイベントでない場合、方法４１００はステップ４１１７へと進む。ステップ４１１５において、ブラウザアプリケーションは非アクティブ状態へと戻り、方法４１００はステップ４１０３へと戻る。

【０３０７】

ステップ４１１７において、イベントがＥＸＩＴ＿ＮＯＷイベントである場合、方法４１００は終了する。ＥＸＩＴ＿ＮＯＷイベントでない場合、方法４１００はステップ４１１９へと進む。ステップ４１１９において、イベントは無視され、方法４１００はステップ４１０３へと戻る。

【０３０８】

図４１は、ソフトウェアアーキテクチャ２００を組み込むシステム６００上で実行されるブラウザアプリケーション方法４２００を示すフローチャートである。方法４２００は、

コンピュータで実現する場合にはＣＰＵ２０５により実行することができる。また、セットトップボックスで実現する場合にはＣＰＵ４３０５により実行することができ、リモートサーバのＣＰＵによっても実行することができる。方法４２００はステップ４２０１で開始される。ステップ４２０１において、イベントがＰＲＥＳＳイベントである場合、方法４２００はステップ４２２５へと進む。ＰＲＥＳＳイベントでない場合、方法４２００はステップ４２０３へと進む。ステップ４２０３において、イベントは無視され、方法４２００は終了する。ステップ４２２５において、区別用識別子がイベントから判定される。次のステップ４２２７において、現在の区別用識別子について現在のページに通知されている場合、方法４２００はステップ４２０５へと進む。通知されていない場合、方法４２００はステップ４２２９へと進む。ステップ４２２９において、現在のトップレベル文書中で引数としての区別用識別子と共にＮｏｔｉｆｙ＿Ｃａｒｄ＿ＩＤとして知られるＪａｖａＳｃｒｉｐｔコールバック関数が呼び出され、方法４２００はステップ４２０５へと進む。

【０３０９】

ステップ４２０５において、データがイベントから得られる。次のステップ４２０７において、データが１文字である場合、方法４２００はステップ４２０９へと進む。１文字でない場合、方法４２００はステップ４２１１へと進む。ステップ４２０９において、文字がブラウザアプリケーション４０３に送信され、方法４２００は終了する。これは、ユーザがキーボードのキー又は従来のリモコンのボタンを押下するのと同じ効果を提供するのに使用されても良い。現在のページは、Hyper Text Mark-up Language（ＨＴＭＬ）により提供されるような既存の方法を使用して、任意のキー押下を受信する際に実行される動作を提供しても良い。

【０３１０】

ステップ４２１１において、データが「ｊｓ：」で開始される場合、方法４２００はステップ４２１３へと進む。「ｊｓ：」で開始されない場合、方法４２００はステップ４２１５へと進む。ステップ４２１３において、現在のトップレベル文書中のＪａｖａＳｃｒｉｐｔ関数が呼び出され、方法４２００は終了する。指定のデータはオプションとしてＪａｖａＳｃｒｉｐｔ関数に対する引数を含んでも良い。例えば、データ「ｊｓ：ｈｅｌｌｏ」はブラウザコントローラがＪａｖａＳｃｒｉｐｔ関数「ｈｅｌｌｏ」を呼び出すことを示し、データ「ｊｓ：ｈｅｌｌｏ（ｗｏｒｌｄ）」はブラウザコントローラが引数「ｗｏｒｌｄ」と共にＪａｖａＳｃｒｉｐｔ関数「ｈｅｌｌｏ」を呼び出すことを示すことになる。

【０３１１】

ステップ４２１５において、データが「ｃｍｄ：」で開始される場合、方法４２００はステップ４２１７へと進む。「ｃｍｄ：」で開始されない場合、方法４２００はステップ４２１９へと進む。ステップ４２１７において、指定のブラウザ関数が呼び出され、方法４２００は終了する。例えば、データ「ｐｒｉｎｔ」により、ブラウザコントローラはデータ「ｂａｃｋ」がブラウザコントローラにより前に表示されたページに戻るようにブラウザに指示するように指示するであろう。

【０３１２】

ステップ４２１９において、データが絶対ＵＲＬである場合、方法４２００はステップ４２２１へと進む。絶対ＵＲＬでない場合、方法４２００はステップ４２２３へと進む。ステップ４２２１において、データがＵＲＬとしてブラウザアプリケーション４０３へとロードされ、方法４２００は終了する。

【０３１３】

ステップ４２２３において、ベースＵＲＬが付加された後、データはＵＲＬとしてブラウザアプリケーション４０３へとロードされ、方法４２００は終了する。

【０３１４】

図４０を参照して先に説明したブラウザコントローラアプリケーションの変形例がソフトウェアプログラムを制御するプログラムコントローラである。ソフトウェアプログラムは

、１つ以上のキー押下イベント（例えば、キーボードのキー押下イベント又はゲームコントローラでのそれに類するイベント）と共に通常制御される任意のプログラムを含むことができる。プログラムコントローラは、対話式ゲームなどの既存のソフトウェアプログラムのカードベースの制御を提供するのに使用することができる。プログラムコントローラプロセスは、以下に示すような例外はあるが、図４０を参照して説明したのとほぼ同様に動作する。ステップ４１０５におけるイベントがＧＡＩＮＩＮＧ＿ＦＯＣＵＳイベントである場合、プログラムコントローラプロセスはＧＡＩＮＩＮＧ＿ＦＯＣＵＳメッセージから制御対象のソフトウェアプログラムに対するリソースロケータを取得するステップへと進む。プロセスは、リソースロケータにより指定されるソフトウェアプログラムを取得・開始するステップへと進む。プログラムコントローラプロセスはステップ４１０３へと進む。更に、ステップ４１０９において、ＰＲＥＳＳイベント、ＲＥＬＥＡＳＥイベント、又はＭＯＶＥイベントを検査する代わりに、方法４１００におけるこの変形例は実質的にＰＲＥＳＳイベントについてチェックするであろう。イベントがＰＲＥＳＳイベントである場合、プロセスはイベントからデータを取得し、データから最初の文字をとり、その文字のキー押下を実現し、結果的に、ユーザがキーボード上でその文字をタイプしたのと同じ効果が得られるステップへと進む。

【０３１５】

１０．１　ランチャに対する特殊なルーティング規則

ランチャ３０３は特殊な一連のルーティング規則を有し、以下のイベントを常時受信する：

・EM_REMOTE_INSERT
・EM_REMOTE_REMOVE
・EM_REMOTE_BADCARD

また、ランチャは、サービス識別子が現在のフロントアプリケーションと一致しない場合、あるいは、区別用識別子がＮＯ＿ＣＡＲＤの現在の識別子（すなわち、全て０）を表す場合、EM_REMOTE_PRESSメッセージ、EM_REMOTE_ RELEASEメッセージ、及びEM_REMOTE_MOVEメッセージを受信する。メッセージが一致するか否かを判定するために、サービス固有識別子は無視される。

【０３１６】

ランチャ３０３は、EM_GAINING_FOCUSイベントをそれ自体に送信することによって明確にそれ自体をフロントアプリケーションにするように構成することができる。この例では、全てのメッセージはメッセージのサービス識別子に関わらずランチャ３０３に送信されることになる。ランチャ３０３は、プロトコルによってこれらのメッセージのいずれかに応答する必要はない。

【０３１７】

１０．２　サンプル実現例

本節はランチャ構成の幾つかの例の概要を説明する。

【０３１８】

１０．２．１　総称ランチャ

総称ランチャは、ブロードバンドインターネット接続能力を有するオープンなセットトップボックス又はコンピュータ環境で使用することができる。この構成によると、ランチャ３０３はローカルマシン又は指定のリモートマシンへとダウンロードし、実行させることができるアプリケーションが存在することを前提とする。また、総称ランチャは、ブラウザコントローラ４０２を介してブラウザ４０３を使用するアプリケーションを使用できるように構成することができる。

【０３１９】

総称ランチャは、永続的なアプリケーションをサポートすると共に、アプリケーションをダウンロードするように構成することができる。システム６００を稼働させるコンピュータ１００は、相当に高速なインターネット接続が利用可能であるのが好ましい。この例では、アプリケーション３０４の一部は、上述のように、ブラウザコントローラ４０２と呼

ばれる永続的なアプリケーションにより処理されるＪａｖａＳｃｒｉｐｔを有するウェブページであっても良い。更に、アプリケーション３０４の一部は協働するように設計することができる。また、総称ランチャは、リモートリーダ１により使用される通信リンクの信頼性が低く（すなわち、ＩＲリンク）、メッセージが損失する可能性があることを前提とするのが好ましい。

【０３２０】

１０．２．２　総称ランチャに対する規則

以下の規則は、ランチャ３０３がシステム６００を定義するのに使用されるのが好ましい規則である。

【０３２１】

・ＮＯ＿ＣＡＲＤの現在の識別子（すなわち、全て０）を有するEM_REMOTE_PRESSイベント及びEM_REMOTE_RELEASEイベントは、ユーザがフロントアプリケーションから出ることを希望する合図として使用される。システム６００はその構成により、ディスプレイ１０１上に「カードを挿入してください」メッセージを生成するか、あるいは、前のアプリケーションへと戻ることになるであろう。

【０３２２】

・EM_REMOTE_BADCARDイベントにより、ランチャ３０３はカードが不良であることを示すフィードバックをユーザに提供する。

【０３２３】

・リモートリーダ１からイベントマネージャ３０１までの通信方法の信頼性が低いことが前提とされているため、EM_REMOTE_INSERTイベント及びEM_REMOTE_REMOVEイベントは、セッションの境界を提供することが期待されていない。

【０３２４】

・EM_REMOTE_PRESSメッセージ、EM_REMOTE_RELEASEメッセージ、又はEM_REMOTE_MOVEメッセージを受信する場合、ランチャ３０３はサービスマッピングを行ない、サービス識別子がダウンロード可能なアプリケーションへと分解される場合、対応するアプリケーションがダウンロードされ、実行される。マッピングは、カードからのサービス情報を伴ってディレクトリサーバ３０５に問合せを行なうことによって実行される。ディレクトリサーバ３０５から戻される値は、アプリケーション位置及び関連するサービスデータである。アプリケーション位置は、アプリケーションの位置又はローカルアプリケーションとしてランチャが認識する値を指定する。サービスデータは、EM_GAINING_FOCUSメッセージに含まれてアプリケーションに送信される初期化データである。アプリケーション位置が空の場合、ランチャ３０３はＵＲＬとなるであろうサービスデータに基づいてどのアプリケーションを使用すべきかを決定するように構成される。

【０３２５】

・新規のアプリケーションがEM_APP_REGISTERメッセージと共に登録されるとき、指定のサービスグループが現在実行中の一連のアプリケーションと比較され、重複がない場合、現在実行中の他の全てのアプリケーションは終了するように通知される。新規のアプリケーションは現在のフロントアプリケーションになり（EM_GAINING_FOCUSイベントを使用）、前のフロントアプリケーションにはEM_LOSING_FOCUSイベントが送信される。この状況が発生し、サービス識別子がウェブページへと分解される場合、EM_GAINING_FOCUSメッセージを使用することによって、データフィールドにウェブページのアドレス(位置）が入った状態でフォーカスがブラウザコントローラ４０２へと変更される。データフィールドは、ランチャ３０３にサービス識別子がウェブページへと分解されたことを通知する問合せに含まれて戻される。この状況で、EM_LOSING_FOCUSイベントも現在のフロントアプリケーションに送信される。他の全てのアプリケーションは終了するように通知される。

【０３２６】

１０．３　単独使用のシステムの例

アーキテクチャ２００は単一の専用アプリケーションと共に使用するように構成することができる。この例では、ユーザインタフェースの一部又は全体を印刷可能な物理的なトー

クン（例えば、バンクカード）を有するのが有利な場合にランチャ３０３を使用することができる。後述する例は自動現金預払機の形態をとるが、特殊なアプリケーションとして説明されているに過ぎず、自動現金預払機に限定されるものとして解釈すべきではない。このようなシステムは、１枚又は少なくとも非常に限定された枚数のカードを使用できるように構成することができる。このシステムでは、挿入されるカードに関わらず、他のアプリケーション３０４は開始されない。ランチャ３０３はシステムコントローラの役割に加え、単一のアプリケーション３０４の役割を担う。イベントマネージャ３０１には変更が加えられない。

【０３２７】

単独使用のシステムは、例えば、自動現金預払機に使用することができる。銀行は、表面に一般に用いられるオプションをつけた個人用バンクカードを作製することができる。このカードは、自動現金預払機に対する唯一又は補助のインタフェースとして使用される。この例では、自動現金預払機はイベントマネージャ３０１及びアーキテクチャ２００の他のコアプロセスコンポーネントを含むのが好ましい。この例では、リモートリーダ１とイベントマネージャ３０１との間の通信リンクは信頼性の高いものでなければならない。

【０３２８】

１０．３．１　規則

以下の規則は、ランチャ３０３が単独使用のシステムの銀行の現金自動預払機の一例を定義するのに使用することができる：

・参加銀行と関連付けられるカードによるものでないイベントは、ランチャが端末に互換性のないカードの画面を表示する原因となる。

・EM_REMOTE_BADCARDイベントは無視される。

・EM_REMOTE_INSERTイベントはトランザクションを開始するのに使用される。

・EM_REMOTE_REMOVEイベントはトランザクションを終了するのに使用される。

・EM_REMOTE_PRESSイベント、EM_REMOTE_RELEASEイベント、及びEM_REMOTE_MOVEイベントはユーザインタラクションとして扱われる。これらのイベントは実行中のアプリケーションのようにランチャにより直接処理されるのが好ましい。

・外部のディレクトリサーバへのサービスマッピングが行なわれることはない。カードが特定の自動現金預払機（ＡＴＭ）が認識するものでない場合、カードは拒否される。

【０３２９】

これらの規則は、ＡＴＭの形態で専用のアプリケーションを提供するように単独使用のシステムを構成することができる方法の例である。

【０３３０】

１０．４　ディレクトリサービスの動作

図５８は、ディレクトリサービス３１１により実行されるプロセス５８００の概要を示すフローチャートである。プロセス５８００はコンピュータ１００のＣＰＵ２０５により実行される。このＣＰＵ２０５がディレクトリサービス３１１の役割を果たす。図５８に示すソフトウェアプログラムは、システム６００Ａのメモリ２０６又はＣＤ－ＲＯＭ２１２あるいはシステム６００Ｂのメモリ４３０６などのメモリ媒体に格納される。プロセス５８００は第１のステップ５８０１で開始される。ステップ５８０１において、ディレクトリサービス３１１が開始される。次のステップ５８０２において、ＣＰＵはランチャ３０３からの入力イベントを待つ。イベントはイベントマネージャ３０１を介して読取り装置１からランチャ３０３に送信される。次のステップ５８０３において、ＣＰＵはランチャ３０３から区別用識別子を含む要求を受信する。この区別用識別子はディレクトリサービス３１１によりマップされる。ランチャ３０３とディレクトリサービス３１１との間の接続は図８に示される。

【０３３１】

次のステップ５８０４において、ＣＰＵはディレクトリマッピングテーブルを検索し、テーブルが区別用識別子に対応するエントリを有するか否かをチェックする。ディレクトリマッピングテーブルは、通常、サービス識別子と対応するアプリケーション位置（ＵＲＬ

など）及びサービスデータとの間の関係を含み、更に、区別用識別子と対応するアプリケーション位置及びサービスデータとの間の関係を含む。通常、サービス識別子を含む関係はカード１０に対して使用され、ディレクトリサービス３１１は、そのサービスに対して使用することができる全てのカード１０に対するサービスレベル情報（例えば、カード１０に対するサービスを提供するように実行されるアプリケーション３０４の位置）を維持することを目的とする。また、区別用識別子を含む関係はカード１０に対して使用され、ディレクトリサービス３１１は、同一のサービス固有識別子を有する実際のカード１０又はカードのグループ１０に特有の情報（例えば、カード１０に対するサービスを提供するべく再生されるメディアファイルの位置）を維持することを目的とする。ディレクトリマッピングテーブルは、通常、ハードディスク２１０又はメモリ２０６に格納される。ステップ５８０４において、ディレクトリマッピングテーブル中に区別用識別子に対するエントリがある場合、次のステップ５８０５において、ＣＰＵはこのエントリからアプリケーション位置及びサービスデータを取得し、ステップ５８０６へと移る。ステップ５８０４において、テーブル中に区別用識別子に対するエントリが存在しない場合、ステップ５８０８において、ＣＰＵはこの値の関連部分（通常は、図１１に示すような最初の５バイト）をとることによって区別用識別子からサービス識別子を抽出する。次のステップ５８０９において、ＣＰＵはサービス識別子に対応するエントリを求めてディレクトリマッピングテーブルを検索する。エントリが見つかった場合、次のステップ５８１０において、ＣＰＵはこのエントリからアプリケーション位置及びサービスデータを取得し、ステップ５８０６へと移る。ステップ５８１１において、エントリが見つからない場合、特定の区別用識別子に対する要求が行なわれたことを示すエントリがログファイルに配置される。ステップ５８１２において、供給された区別用識別子のサービス識別子部分がディレクトリサービス３１１にとって未知のものであることを示すエラーがランチャ３０３に戻される。フローはステップ５８０２へと続く。

【０３３２】

ステップ５８０６において、すなわち、区別用識別子又はサービス識別子の検索が成功した場合、区別用識別子と対応するアプリケーション位置及びサービスデータとがログファイルに書き込まれ、ＣＰＵは要求を行なったランチャ３０３にアプリケーション位置及びサービスデータを戻す。フローはステップ５８０２へと続き、別のイベントを待つ。

【０３３３】

１１　総則

通常、アプリケーション３０４はハードディスクドライブ２１０に常駐し、ＣＰＵ２０５による実行時に読み出され制御される。プログラム及びネットワーク２２０から取り込んだデータの中間記憶装置は、半導体メモリ２０６を使用し、場合によってはハードディスクドライブ２１０と協働させるように使用して達成されても良い。アプリケーション３０４は、ＣＤ－ＲＯＭ又はフロッピーディスク上に符号化された形でユーザに供給され、対応するドライブ２１２又は２１１を介して読み取られる場合もあれば、ネットワーク２２０からモデム装置２１６を介して読み取られる場合もある。その他のコンピュータ可読な媒体からコンピュータシステム１００へとソフトウェアアプリケーションをロードするその他のメカニズムには、磁気テープ、ＲＯＭ又は集積回路、光磁気ディスク、コンピュータモジュール１０２と別の装置との間の無線又は赤外線の伝送チャネル、スマートカード、コンピュータのＰＣＭＣＩＡカードなどのコンピュータ可読なカード、及び電子メール送受信並びにウェブサイトなどに記録された情報を含むインターネット並びにイントラネットが含まれる。以上の媒体は、関連するコンピュータ可読な媒体のほんの一例である。上述の例の組み合わせを含め、その他のコンピュータ可読な媒体も使用可能である。

【０３３４】

また、上述のプロセスコンポーネント３０１から３０６は、上述の関数及び下位関数を実行する１つ以上の集積回路などの専用のハードウェアで実現することもできる。このような専用のハードウェアは、グラフィックＣＰＵ、デジタル信号ＣＰＵ、又は、１つ以上のマイクロＣＰＵ及び関連メモリを含んでも良い。専用ハードウェアの例は、図６（ｂ）を

参照して説明したテレビ用のセットトップボックス６０１である。
【０３３５】
１２　その他の変形例
１２．１　セッション識別子
上述のように、区別用識別子は、リーダ１からコンピュータ１００又はセットトップボックス６０１に送信される全てのＩＮＳＥＲＴメッセージ、ＲＥＭＯＶＥメッセージ、ＰＲＥＳＳメッセージ、ＲＥＬＥＡＳＥメッセージ、及びＭＯＶＥメッセージに含まれる。また、区別用識別子はＩＮＳＥＲＴメッセージのみと共に送信することもできる。この例では、新規のカード１０の挿入時に、リーダ１がセッション識別子（不図示）を生成する。セッション識別子はカード挿入の現在のセッションを識別する。セッション識別子は、例えば、擬似乱数（２バイトのデータを用いて表すことができる）とすることも、カードが挿入されるたびに増分される（所定の値に達すると０にリセット）数値とすることもできる。リーダ１はコンピュータ１００又はセットトップボックス６０１にＩＮＳＥＲＴメッセージを送信する。このメッセージは先に説明した区別用識別子及び新規の挿入ごとに生成されるセッション識別子を含む。後続するＰＲＥＳＳメッセージ、ＲＥＬＥＡＳＥメッセージ、及びＭＯＶＥメッセージは、全て区別用識別子を含む必要がないが、セッション識別子と先に説明したユーザインタフェースオブジェクトデータ又は押下座標とを含むであろう。
【０３３６】
セッション識別子を使用するとき、システム６００は、リーダ１からＩＮＳＥＲＴメッセージを受信する際にイベントマネージャ３０１がセッション識別子を現在のセッション識別子として格納し、区別用識別子を現在の区別用識別子として格納することを除いては図６（ａ）及び図６（ｂ）を参照して説明したのと同様の処理を実行する。ＰＲＥＳＳメッセージ、ＲＥＬＥＡＳＥメッセージ、又はＭＯＶＥメッセージを受信するとき、イベントマネージャ３０１はセッション識別子が現在のセッション識別子と等しいことを確認する。等しい場合、イベントマネージャ３０１は、全てのメッセージにおいて使用される区別用識別子を現在の区別用識別子に設定する。これに対し、セッション識別子が現在のセッション識別子と等しくない場合、イベントマネージャ３０１は、ディスプレイマネージャ３０６及び表示装置１０１を介してメッセージが対応するＩＮＳＥＲＴメッセージを伴わずに受信されたことをユーザに通知する。この場合ユーザは、例えば、カード１０を抜き取って再挿入するように要求される。
【０３３７】
１２．２　ユーザ押下のその他の特徴
上述のように、情報の送信は、リーダ１のタッチパネル８上でのオブジェクトの押下、移動、及び押下解除（通常、指又はスタイラスを用いる）に関する。しかし、リーダ１は、タッチパネル８からのインタラクションに関する追加の情報をシステム６００により使用するためにコンピュータ１００又はセットトップボックス６０１に送信することができる。例えば、追加の情報は、押下の結果タッチパネル８に対して作用する圧力の時間及び量を表すことができる。この追加の情報は、リーダ１からシステム６００に送信されるＰＲＥＳＳメッセージに組み込むことも、システム６００内で送信されるEM_READER_PRESSメッセージに組み込むこともできる。この例では、情報はリーダ１に挿入されるカードに対応するアプリケーション３０４に渡される。アプリケーションは、追加の情報を利用して、例えば、特定の動作に対して付加的な効果をもたらすことができる。例えば、アプリケーションは音量の増加を示す印の押下と関連付けられるときに、圧力情報を使用して音量の増加量を判定することができる。すなわち、選択された印に対する押下が強いと音量の増加率が高くなり、逆に、選択された印に対する押下が弱いと増加率が低くなる。
【０３３８】
タッチパネル８とのインタラクションの時間（すなわち、持続時間）に関し、追加の情報の使用の別の例を以下で説明する。押下の持続時間が非常に短い場合、押下は「タップ」であると見なすことができる。これに対し、押下の持続時間が非常に長い場合、キー押下

の永続的な「押し下げ」であると見なすことができる。この例では、追加の情報はインスタントソフトウェアアプリケーションと対話するモードに追加の次元を加えることができる。例えば、タッチパネル８上での「タップ」をソフトウェアアプリケーションに現在の（画面上の）カーソル位置に表示される項目を選択するように指示する命令とすることができる。

【０３３９】

１２．３　座標なし

ＰＲＥＳＳメッセージ及びＲＥＬＥＡＳＥメッセージは、ユーザのタッチパネル８とのインタラクションの座標データを含まないように構成することができる。この例では、座標データのみが、ＭＯＶＥメッセージと共にリーダ１からシステム６００に送信される。ＰＲＥＳＳメッセージ及びＲＥＬＥＡＳＥメッセージに座標データを含まないことの利点は、アプリケーション３０４が座標からユーザインタフェース要素データへとマッピングするのに座標情報を必要としない場合に、リーダ１によりシステム６００に送信されるメッセージのサイズを縮小できることである。

【０３４０】

１２．４　双方向プロトコル

単方向又は双方向通信は、リーダ１とコンピュータ１００又はセットトップボックス６０１との間の通信に使用することができる。図１０を参照したリーダ１の説明及び図８及び図９を参照したＩ／Ｏデーモンの説明は、リーダ１からコンピュータ１００又はセットトップボックス６０１への情報の送信及び逆方向の送信を含むものであった。コンピュータ１００又はセットトップボックス６０１からリーダ１への情報の送信は、カード１０に格納されたデータを変更するのに使用することができる。例えば、スマートカード１０のメモリチップ上に格納されたユーザインタフェースオブジェクトデータの変更に使用することができる。

【０３４１】

また、双方プロトコルはプロトコルにおけるハンドシェーキングを可能にするために使用することができる。例えば、リーダ１とセットトップボックス６０１又はコンピュータ１００との間の双方向プロトコルは、カードがリーダ１に挿入されるときにシステム６００がＩＮＳＥＲＴメッセージの受信を確認することができるように使用することができる。双方向プロトコルをサポートするシステム６００は、アプリケーションがカード１０上の格納されたデータの一部を変更するために要求を送信することができるように、イベントマネージャ３０１を介してＩ／Ｏデーモン３００に送信されるイベントマネージャプロトコルでの追加のメッセージを提供すべきである。Ｉ／Ｏデーモン３００は、リーダ１にメッセージを送信して要求される動作を引き起こすことができる。例えば、システム６００が双方向プロトコルを使用する場合、システム６００はアプリケーションがユーザの許可又はシステム定義の特権なしにカードを変更することができないことを保証するための機密保護メカニズムを提供することができる。このようなシステムの一例において、イベントマネージャ３０１は、アプリケーションが現在挿入されているカードを変更しても良いか否かを尋ねる表示メッセージをユーザに対して提示することができる。ユーザは、タッチパネル８の第１の領域を押下することによって提案に同意することができると共に、タッチパネル８の第２の領域を押下することによって提案への不同意を示すことができる。ユーザがカード１０の変更に同意する場合、イベントマネージャ３０１は、アプリケーション３０４からの要求がＩ／Ｏデーモン３００へと渡され、更に、リーダ１へと渡されるようにすることができる。これに対し、ユーザが変更に不同意の場合、イベントマネージャ３０１はメッセージを放棄するので情報はリーダ１に送信されない。

【０３４２】

１２．５　代替の読取り装置

上述のシステム６００Ａ及び６００Ｂにおいて、読取り装置１はカード１０に重なるように構成されるほぼ透明な感圧膜を有する。読取り装置１の費用を削減するために、読取り装置１は感圧膜の代わりに複数のユーザ操作可能なスイッチをレセプタクルの周囲に配置

しても良い。レセプタクルには、データ、すなわち、区別用識別子とデータを各スイッチと関連付けるための関係情報とを読むためにスマートカード１０を挿入することが可能である。各スイッチのうちの操作可能なものはスマートカード上の印と視覚的と関連付けられるため、ユーザはカード上の少なくとも１つの印に対応するスイッチのうちの少なくとも１つを選択することができる。この場合、ＣＰＵ４５はカード１０からの関係情報及び区別用識別子に基づいてユーザにより押下されるスイッチに対応するデータを読み、それをイベントマネージャ３０１に送信する。

【０３４３】

１３．０　代替のソフトウェアアーキテクチャ

システム６００により表されるハードウェアアーキテクチャに対する更なる一般的なソフトウェアアーキテクチャ４９００が図４８に示される。このアーキテクチャ４９００は、これまで各節で説明したものの代替のソフトウェアアーキテクチャを表す。代替のアーキテクチャ４９００は、ユーザの自宅における非常に低レベルのハードウェア要件（すなわち、簡易セットトップボックス）から、例えば、セットトップボックス６０１の機能がパーソナルコンピューティングシステム上で実現される強力なホームシステムに至るまでスケール変更されるように構成される。更に、代替の構成４９００は、ハードウェアシステム６００内で実現されるのが好ましい。

【０３４４】

１３．１　構造

アーキテクチャ４９００は、６つの別個のプロセス及び１つのプロセスクラスへと分割される。別個のプロセスは、アーキテクチャ２００と同様にＩ／Ｏデーモンと呼ばれるスマートカードインタフェース４９０２、イベントマネージャ４９０４、ディスプレイマネージャ４９０６、マスタランチャ４９０８、（アプリケーション）ランチャ４９１０、及びディレクトリサービス４９１２を含む。プロセスクラスは１つ以上のスマートカードアプリケーション４９２０により形成される。アーキテクチャ４９００には、セットトップボックス６０１により通常形成されるスマートカードリモート接続ごとに、１つのカードデーモン４９０２、１つのイベントマネージャ４９０４、１つのディスプレイマネージャ４９０６、及び１つのランチャ４９１０と、各ランチャ４９１０を稼動させているコンピュータごとに１つのマスタランチャ４９０８と、全てのシステムに対して少なくとも１つのディレクトリサービス４９１２とが存在する。

【０３４５】

この形態において、図４８の破線により示すように、アーキテクチャ４９００は３つの別個の部分４９１４、４９１５、及び４９１６へと物理的に分離することができる。各部分は物理的に別々の計算装置上で実行させることができる。システムの各部分間の通信は、アーキテクチャ２００と同様にＴＣＰ／ＩＰストリームを使用して実行される。

【０３４６】

Ｉ／Ｏデーモン４９０２は、スマートカードリモートリーダ１から受信されるデータグラムをＴＣＰ／ＩＰストリームへと変換するプロセスである。Ｉ／Ｏデーモン４９０２は、リーダ１により使用されるデータ形式を理解するのではなく、スマートカードリモートデータ形式の変更とは関係なく動作することを意図し、複数のバージョンのリーダ１と協働する機能を提供する。

【０３４７】

Ｉ／Ｏデーモン４９０２は、ユーザがシステム６００を開始するときに開始されるが、これは、セットトップボックスシステム６００Ｂの場合、セットトップボックス６０１の電源が投入されるときである。コンピュータシステム６００Ａの場合、Ｉ／Ｏデーモン４９０２は、イベントマネージャ４９０４及びマスタランチャ４９０８が開始された後、ユーザがスマートカードシステムを開始するときに開始されても良い。

【０３４８】

イベントマネージャ４９０４は、全ての通信がイベントマネージャ４９０４を経由するという点でアーキテクチャ４９００の中心を形成する。イベントマネージャ４９０４は、ス

マートカードリモートリーダ１により生成され、Ｉ／Ｏデーモン４９０２により中継される全てのイベントを収集する役割を担う。これらのイベントは、種々のプロセス及び実行中の各アプリケーションへと再分散される。

【０３４９】

イベントマネージャ４９０４の更なる役割は、挙動の悪いアプリケーションを他の挙動の良いアプリケーションから切り離すことである。この点に関して、イベントマネージャ４９０４を経由するイベントは、イベントマネージャ４９０４がチェック可能な範囲で正しいことが保証される。イベントマネージャ４９０４は、イベントが有効なヘッダ及び正しいデータ長を有することをチェックすることが要求されるが、通常、データが正しい形式であることをチェックするようには構成されていない。

【０３５０】

異なるバージョン間でのプロトコルの変更にも、イベントマネージャ４９０４が対処する。可能な場合には、動作中のアプリケーション４９２０が理解するデータ形式に適合するようにイベントは書き換えられる。書き換えが可能でない場合、イベントマネージャ４９０４は送信元のアプリケーション４９２０にエラーを報告する。異なるデータ形式バージョンが使用されている場合、イベントマネージャ４９０４は混乱の発生を最小限に抑えることを保証する。

【０３５１】

ディスプレイマネージャ４９０６は、どの動作中のアプリケーション４９２０が特定の出力装置１１６、典型的にはディスプレイ（例えば、１１６）に対して優先権を有するかに関して制御するように動作するアプリケーション４９２０と協働して動作する。どのビデオストリームがイベントマネージャ４９０４を介してディスプレイ１１６に送信されるかを選択するのがディスプレイマネージャ４９０６の役割である。この情報はアプリケーション４９２０の各ランチャ４９１０から得られる。一般的に、フロント（すなわち、フォアグラウンド）アプリケーションのみがビデオディスプレイストリームを生成する。更に、ディスプレイマネージャ４９０６は、整合性のない入力ストリームから一定のストリームを出力するように維持し、推定データを用いて出力ストリームの一部を補充するように動作しても良い。

【０３５２】

イベントマネージャ４９０４は、いつアプリケーション４９２０を開始／終了しなければならないかを決定する、あるいは、実際にアプリケーション４９２０を開始／終了する責任はない。後述するように、これらの動作はランチャ４９０８及び４９１０の責任である。更に、イベントマネージャ４９０４は、ユーザ画面又はその他の出力装置１１６にはその存在を示さない。スマートカードの最初の挿入の表示などのシステム関連のフィードバックはランチャ４９１０により実行される。

【０３５３】

代替のアーキテクチャ４９００を組み込む図６（ｂ）のシステム６００Ｂの場合、通常、システム６００Ｂに接続することを許された全てのセットトップボックス６０１に対してイベントマネージャ４９０４が実行されるであろう。アーキテクチャ４９００を組み込むシステム６００Ａの場合、マスタランチャ４９０８が開始された後にスマートカードシステム６００Ａが開始されるときにイベントマネージャ４９０４が開始されることになる。

【０３５４】

マスタランチャ４９０８の役割は、イベントマネージャ４９０４のいずれかの要求に応じてランチャ４９１０を開始することである。Ｉ／Ｏデーモン４９０２がイベントマネージャ４９０４に接続するとき、イベントマネージャ４９０４はマスタランチャ４９０８にイベントマネージャ４９０４に対する第１のプロセスを開始するように要求する。この第１のプロセスは、通常、任意のスマートカードアプリケーション４９２０に対するランチャ４９１０であろう。また、マスタランチャ４９０８は、イベントマネージャ４９０４の要求時にアプリケーション４９２０のランチャ４９１０を停止すると共に、イベントマネージャ４９０４に正しいランチャ４９１０が終了したことを通知する役割を担う。

【０３５５】
代替のアーキテクチャ４９００を組み込む図６（ｂ）のシステム６００Ｂの場合、スマートカードアプリケーション４９２０を実行する物理的に別々のサーバ１５０、１５２の各々に対して、常に、１つのマスタランチャ４９０８が実行されることになる。この１つのマスタランチャ４９０８は、そのサーバ上にランチャ４９１０を要求する全てのイベントマネージャ４９０４に対する要求に対処する。システム６００Ａの場合、マスタランチャ４９０８はスマートカードシステムのその他の部分より前又はそれと同時に動作を開始する。

【０３５６】
カードディレクトリサービス４９１２は、スマートカード１０内に格納されるベンダーアプリケーション値（サービス識別子の値）を後述するベンダーアプリケーション対（サービス識別子）と関連付けられるアプリケーション４９２０を指し示すUniform Resource Locator（ＵＲＬ）などのアプリケーション位置へと変換するために設けられる。ディレクトリサービス４９１２は、アプリケーション４９２０がランチャ４９１０に対して別々のシステム上で実行可能であるようにランチャ４９１０を変更することによって複数の部分へと分割することができる。ディレクトリサービス４９１２は、分散型参照システムを使用してこの機能を実行する。このシステムでは、現在問合せを所持するディレクトリサービスがその答えを知らない場合、問合せは別のディレクトリサーバへと渡される。このような分散型システムにより、各ディレクトリサーバはベンダーアプリケーションＩＤ対からＵＲＬへの変換について限られた知識を有することになるが、依然として全てのＩＤをＵＲＬへと変換することができる。これにより、各ディレクトリにおけるデータベースの簡易化を含め、数多くの利点が提供される。頑強性が向上し、サーバは問合せが許可されている間、動作不能になる（すなわち、クラッシュ又はサービスから外される）ことを許される。

【０３５７】
図５２を参照すると、スマートカード１０を従来のスマートカードと区別する制御テンプレートカスタマイズ情報は、ベンダ識別子、カード識別子、及びアプリケーション識別子によるデータの組を含む。ベンダ識別子／アプリケーション識別子対は、アーキテクチャ２００に関して先に説明したサービス識別子に相当する。また、カード識別子は、アーキテクチャ２００に関して先に説明したサービス固有識別子に相当する。更に、各アイコン４８０４と関連付けられるのは、ユーザがアイコン４８０４上のタッチパネルを押下するときにイベントデータとして送信される対応データである。このイベントデータは、特定のアプリケーション４９２０に渡されるとそのアプリケーション内の特定の動作を実施する。また、例えば、アイコンの押下ではなくアイコンの押下解除を検知するため、更に、ユーザがタッチパネル８を横断するように指をスクライブして特定の機能を実行する場合に押下の移動を検知するため、ユーザ動作の検知が組み込まれても良い。このような動作のたびに、カード上に格納されるイベントデータを送信することができる。このデータはその都度カード上のメモリの異なる位置から読まれても良い。この代替のアーキテクチャ４９００においてベンダーアプリケーション識別子対として実現されるサービス識別子により、スマートカードと関連付けられるアプリケーションのベンダは相互に区別することができる。スマートカードと関連付けられるアプリケーションのベンダを区別する必要がない場合のアーキテクチャ４９００の展開において、ベンダ識別子及びアプリケーション識別子は、単一の値：サービス識別子として扱うことができる。

【０３５８】
スマートカード１０のリーダ１への挿入により開始される第１のプロセスは、汎用システム（一般家庭など）ではランチャ４９１０になるであろう。特殊なシステムでは、特殊なアプリケーションが開始されても良い。例えば、バンキングテラーはバンキングアプリケーションを開始するであろう。別の例としては、アプリケーションの特定の部分集合のみを開始する限定的なランチャの使用が含まれる。ランチャ４９１０は、特定の１つのイベントマネージャ４９０４に対して他のアプリケーション４９２０を開始するスマートカー

ドアプリケーションである。アプリケーション４９２０を開始／終了させ、実際にアプリケーションを開始／終了するのはランチャ４９１０の決定である。ランチャ４９１０は、イベントマネージャ４９０４にいつアプリケーションが開始／終了するかを通知し、アプリケーション４９２０にいつフォーカスを受け取る／失うか、あるいは、いつ終了する必要があるかを通知する。この点に関して、複数のアプリケーション４９２０が同時に動作している場合、現在画面に表示されているアプリケーションがフォーカスを有するアプリケーションである。別のアプリケーションが優先権を得ようとすると、ランチャ４９１０は現在のアプリケーションにフォーカスが失われることを通知するので、現在のアプリケーションは当座のタスクを完了することができる。また、ランチャ４９１０は新規のアプリケーションにフォーカスが得られ、まもなく新規のアプリケーションが変更状態に入ることを通知する。ランチャ４９１０はアプリケーションを強制的に終了できるように構成されなければならない。

【０３５９】

開始される第１のアプリケーション４９２０（すなわち、通常は、ランチャ４９１０）には特権が与えられ、リモートリーダ１により生成された「ＮＯ＿ＣＡＲＤ」イベント、「Ｂａｄ＿ＣＡＲＤ」イベント、及び「ＰＯＷＥＲ＿ＯＦＦ」イベントを受信する。また、第１のアプリケーション４９２０は、現在のフロントアプリケーションでない各アプリケーション４９２０用のイベントも受信し、これらのイベントを正確に解釈するように動作する。これは上述の特定のアプリケーションに関連するので、ランチャはあらゆる変化を正確に解釈する。ランチャ４９１０は、その他のアプリケーションを開始・停止する権利を含む特殊な権利を有するアプリケーション４９２０である。

【０３６０】

ランチャ４９１０は、イベントマネージャ４９０４によりその開始が要求されるときにのみ開始されるのが好ましい。また、ランチャ４９１０は、イベントマネージャ４９０４により終了するように指示することもできる。

【０３６１】

アプリケーションは、対応するスマートカード１０上での第１のユーザ選択に対する応答として、あるいは、アプリケーション４９２０の別の１つの要求に応じてランチャ４９１０により開始される。この点に関して、アーキテクチャ４９００は、プログラミング中に１つ以上のアプリケーションサービスグループのメンバとして編成される各アプリケーション４９２０を介して、従来の構成に対する実質的な改良を提供する。

【０３６２】

１３．２　アプリケーションサービスグループ

アプリケーションサービスグループは、単に同時に動作するのではなく協力して動作し、機能の特定の集合を提供する複数のスマートカードアプリケーション４９２０から構成される。サービスグループの一部を形成するアプリケーション４９２０は、同時に実行されることを許されると共に、データの交換を行なう際に利用されるであろう通信手段（すなわち、イベントマネージャ４９０４）を共有する。このようなアプリケーション４９２０は、各々が特定のユーザインタフェース又はユーザインタフェースの集合に対応する機能の集合を提供するプロセス又は下位プロセスである。このアプリケーション４９２０は、可視のディスプレイを有しても有しなくても良い。

【０３６３】

図４９に表される例を参照すると、サービスグループは、その一部を形成するアプリケーション４９２０が開始され、そのサービスグループをイベントマネージャ４９０４に登録するときに開始される。図４９に示すように、第１のアプリケーション４９２６は２枚のスマートカード４９２４及び４９２６をそれ自体と関連付け、第２のアプリケーション４９３４はスマートカード４９２８、４９３０、及び４９３２を用いて動作可能である。従って、カード４９２２をリーダ１へと挿入する際に、カードデーモン４９０２は、ランチャ４９１０を介してアプリケーション４９２６を開始するイベントマネージャ４９０４にその発生を通知する。アプリケーション４９２６の開始により、サービスグループ４９３

６が使用可能になる。このグループはアプリケーション４９３４を含む。現在確立されているサービスグループに対応するアプリケーションは、関連するスマートカードを挿入することによって開始されても良い。例えば、カード４９２２の抜取り及びカード４９３２の挿入は、アプリケーション４９３４を起動するように作用し、サービスグループ４９３６をアクティブ状態に維持する。更に、同じサービスグループの一部を形成するアプリケーションの開始は、同じサービスグループの他のアプリケーションの終了を引き起こさない。その他のアプリケーションはバックグラウンドで実行を継続する。

【０３６４】

サービスグループの終了は、空のリモートリーダ１に接触する、あるいは、サービスグループ４９４２中のアプリケーション４９４０に対応するカード４９３８などの別のサービスグループに対応するスマートカードを挿入することによって引き起こされる。サービスグループの終了により、現在そのサービスグループの一部として実行されている全てのアプリケーションも終了させられる。

【０３６５】

同じサービスグループ下で実行されるアプリケーションは、図４９に示すように、サービスグループ定義のプロトコル４９５０によってイベントマネージャ４９０４を介して相互に通信しても良い。プロトコル４９５０において、アプリケーション（例えば、４９２６及び４９３４）間で送信されるデータパケットの形式及び内容は、同じサービスグループ内に共存するアプリケーションの作成者により定義されるべきである。

【０３６６】

図５０に示されるのは、アーキテクチャ４９００内のサービスグループの別の特徴である。ここでは、サービスグループは他のいずれかのサービスグループの一部として実行される１つ以上のアプリケーションを含んでも良い。これらのアプリケーションは、複数のサービスグループにまたがって要求される可能性があるサービスを提供する。このようなアプリケーションの一例は、ユーザの住所及びクレジットカードの詳細（ユーザが一度これらの詳細を提供することに同意した場合）を提供することができる個人識別サービスである。この点に関して、このようなサービスは、オンラインショッピング及びオンラインバンキングを含む金融取引を必要とする数多くのその他のサービス又はトランザクションのコンポーネントを形成しても良い。

【０３６７】

アーキテクチャ４９００をサポートするためのアプリケーションの設計は、開発されるアプリケーションがアーキテクチャ４９００により提供される新規の特徴を必要とするか否かによって、アプリケーション設計に対する既存のアプローチと同じであっても、あるいは、異なっていても良い。何らかの変形を施しても、既存のアプリケーションはアーキテクチャ４９００の下で機能するであろう。このような変形の例としては、既存のアーキテクチャの下で稼働する各アプリケーションが、実行中のアプリケーションと同じ名前を有するサービスグループをもつ（すなわち、各アプリケーションはメンバを１つだけ有する独自のサービスグループを形成する）ことを想定できる場合、あるいは、一意的なグループ名を選択するその他の方法がある。この方法は、他のアプリケーションと協働しない既存のアプリケーションに対してグループ名をもたないことを含む。

【０３６８】

同じサービスグループ内の各アプリケーションは、同じ物理ハードウェア上で動作する必要はなく、ＯＳ定義の方法を使用することによって相互に直接通信可能でなくても良い。２つの通信方法は、イベントマネージャ４９０４において実現され、アプリケーション間通信の標準的な方法を提供するのが好ましい。これらの方法は：

（ｉ）メッセージが１つのアプリケーションにより別のアプリケーションに送信されるデータグラムベースのプロトコルと、

（ｉｉ）アプリケーション４９２０により、同じサービスグループ中の任意のアプリケーション４９２０がメッセージを読むことができる共通の領域へとメッセージが掲示される掲示板に基づくプロトコルである。

【０３６９】
イベントマネージャ４９０４は、アプリケーション４９２０間で渡されるデータに対して構造を強制しない。全てのメッセージは既知の長さのただのデータブロックである。データに対して強制される他の任意の構造は、特定のサービスグループのアプリケーションにより理解されれば良い。データブロックには、ポスティングアプリケーションによってイベントマネージャ４９０４により格納される型（例えば、生データ、．ｗａｖ、．ｄｏｃなど）が付与されても良い。
【０３７０】
サービスグループ中の１つのアプリケーションから同じサービスグループ中の別のアプリケーションへの任意の長さのデータの送信を可能にするのには、データグラム方式が使用される。この方法では、送信側アプリケーションが受信側アプリケーションの識別（ＩＤ）番号（上述のようにＸｉｄとも呼ばれる）を知っている必要がある。ＩＤ番号は、アプリケーション４９２０が開始されるときに対応するランチャ４９１０により生成され、アプリケーション４９２０は一意的に識別される。ＩＤ番号はイベントマネージャ４９０４のコンテキストにおいてのみ一意的である。このように、多くの実行中のアプリケーションは同じＩＤ番号をもつことができるが、全てのＩＤ番号はそのアプリケーションが接続される同じイベントマネージャ４９０４に接続される全てのアプリケーション４９２０において一意的である。イベントマネージャ４９０４は、２重のＩＤ番号が使用されるときにそれを検知することができるが、この検知を保証すると共に新規のアプリケーションの開始を防止するのは、対応するランチャ４９１０の責任である。
【０３７１】
データグラム方式を使用してメッセージを送信するために、送信側アプリケーションはイベントマネージャ４９０４から宛先アプリケーションのＸｉｄを取得し、Ｘｉｄを使用してメッセージのアドレス指定を行ない、イベントマネージャ４９０４を介してメッセージを宛先アプリケーションに送信する。イベントマネージャ４９０４は、メッセージを含むパケットに対してデータ長及びヘッダの送信者フィールドが正しいことを確認する以外には何も行なわない。
【０３７２】
データグラム方式を利用可能にするには、イベントマネージャ４９０４はサービスグループ中で他のどのアプリケーションが実行中であるかを判定する何らかの方法をアプリケーションに提供しなければならない。この情報は、アプリケーションが他のアプリケーションの能力を判定するための何らかの方法も含む。アーキテクチャ４９００では、アプリケーションがイベントマネージャ４９０４に登録するときにアプリケーションが列挙する機能文字列のリストを使用してこれが実行される。イベントマネージャ４９０４はどんな形であろうとこれらの機能を理解する必要がなく、機能のリストはサービス固有のものである。サービス中の他のアプリケーションのみが各機能文字列が何を意味するのかを理解する必要がある。
【０３７３】
イベントマネージャ４９０４は、この方法を使用することで送信可能なメッセージのサイズに上限を課しても良い。
【０３７４】
アーキテクチャ４９００では、上述の掲示板により、データを即座にサービスグループ中の全てのアプリケーションに同報通信することができると共に、サービスグループ中のアプリケーションは中央のリポジトリにデータを格納することができる。これにより、いずれか１つのアプリケーションが常にサービスグループに存在する必要がなくなる。また、掲示板により、スマートカードは任意の順序で挿入することができ、サービスグループ中のアプリケーションは任意の順序で実行することができる。アプリケーションはサービスグループに提供するデータを掲示板に掲示し、アプリケーションがアクションをとる必要があるときには、必要なデータを求めて掲示板を検査することができる。
【０３７５】

データを掲示板に掲示するために、ポスティングアプリケーションは、掲示することが望まれるデータ、すなわち、記述文字列と、場合によっては、何らかの形態のタイピング情報（ＭＩＭＥタイプなど）とをイベントマネージャ４９０４に送信する。アプリケーションがタイプ情報を供給しない場合、イベントマネージャ４９０４はデータにデフォルトのタイプ（例えば、デフォルトのバイナリデータ、ＭＩＭＥタイプアプリケーション／オクテットストリーム）を割り当てることになる。イベントマネージャ４９０４は、掲示板のメッセージを識別するのに使用されるメッセージ識別子をこのメッセージに割り当てる。このメッセージ識別子は、他のアプリケーションにより掲示板からメッセージを取得するのに使用される。また、メッセージ識別子はポスティングアプリケーションが掲示板からメッセージを削除するのにも使用される。掲示板とサービスグループに対応する掲示板に残っているメッセージとは、サービスグループが終了させられると破壊される。

【０３７６】

掲示板からメッセージを取得するには、アプリケーションは必要なメッセージのメッセージ識別子を探さなければならない。アプリケーションは掲示板上のメッセージのリストを取得することができるが、このリストはメッセージ識別子、ポスタ識別子、及び掲示板上の各メッセージのメッセージ記述を含むであろう。第２の方法も、イベントマネージャ４９０４から実行中のアプリケーションのリストを取得することを含む。これにより、アプリケーションには、各アプリケーションがサービスに対して提供する機能が提供される。掲示板にメッセージを要求するアプリケーションは、情報を必要とするアプリケーションのアプリケーション識別子（Ｘｉｄ）を相互参照し、そのアプリケーションにより掲示される全てのメッセージを取得することができる。

【０３７７】

掲示板上のメッセージ及びメッセージ記述の双方の形式はサービスグループにより決定されるものであり、完全に任意のものであっても良い。イベントマネージャ４９０４はデータに対していかなる構造も強制しない。

【０３７８】

このような通信方法をサポートするために、イベントマネージャ４９０４には掲示板を維持することが要求される。イベントマネージャ４９０４にとっては、掲示板は単なる既知の長さのデータブロックのリストとして見える。アプリケーションが掲示板にメッセージを掲示するとき、イベントマネージャ４９０４はそのデータ及び長さを格納する。アプリケーションが掲示板からメッセージを読むとき、イベントマネージャ４９０４はそのデータをアプリケーションに送信する。また、イベントマネージャ４９０４は、掲示板の内容のリストを要求するアプリケーションに対してこれを作成する。

【０３７９】

イベントマネージャ４９０４は、サービスグループ中の全てのアプリケーションにより掲示することが可能な全てのメッセージの合計サイズのみならず、各アプリケーションが掲示することが可能なメッセージの合計サイズを制限しても良く、その結果、各アプリケーション及び各サービスグループはメッセージサイズ制限を有するようになる。また、アプリケーション及びサービスグループが掲示するメッセージの数を制限しても良い。メッセージの記述のサイズも最大長にまで制限されても良い。

【０３８０】

１３．３　システム初期化

本節では、図４８のソフトウェアアーキテクチャ４９００を組み込むシステム６００を最初に開始するプロセスを説明する。これは、分散型セットトップボックスシステム６００Ｂのみならず、コンピュータシステム６００Ａにも関連する。

【０３８１】

最初に、マスタランチャ４９０８が開始され、ネットワーク２２０を介して通信ポートで応答の有無を聞き取る。イベントマネージャ４９０４が開始され、イベントマネージャ４９０４はマスタランチャ４９０８に接続する。

【０３８２】

アーキテクチャ４９００のこれらの２つのコア部分を開始する順序は、システム６００Ａの場合には任意であるが、セットトップボックスシステム６００Ｂにおいて使用されるときには明白な利点を有する。システム６００Ｂでは、イベントマネージャ４９０４が開始されるときにはマスタランチャ４９０８は既に実行中であり、より多くのユーザがサービスを申し込むときにはより多くのイベントマネージャを開始し、ユーザがサービスを終了するときには実行中のイベントマネージャの数を減少させることができる。

【０３８３】

１３．４　システム起動

本節では、図６Ａ又は図６Ｂのハードウェアアーキテクチャ及び図４８の代替のソフトウェアアーキテクチャを組み込むスマートカードシステムを開始するプロセスを説明する。この説明は、イベントマネージャ４９０４及びマスタランチャが既に実行中であり、オープンな接続を有することを前提とする。

（ｉ）Ｉ／Ｏデーモン４９０２が開始され、イベントマネージャ４９０４への接続を開始する。

（ｉｉ）イベントマネージャ４９０４はＩ／Ｏデーモン４９０２からの接続を受け入れる。サービス課金を実行することができるのはこの段階である。例えば、ユーザが料金の支払いを済ませていない場合、接続を拒否することができる。

（ｉｉｉ）イベントマネージャ４９０４は、マスタランチャ４９０８にイベントマネージャ４９０４がどのポートで聞き取りを行なっているかを通知してマスタランチャ４９０８に新規のランチャ４９１０を要求し、入力接続を待つ。

（ｉｖ）マスタランチャ４９０８は新規のランチャ４９１０を開始し、新規のランチャ４９１０にイベントマネージャ４９０４のアドレス及びポート番号を教える。

（ｖ）新規のランチャ４９１０はイベントマネージャ４９０４との接続を開始する。

（ｖｉ）イベントマネージャ４９０４は接続を受け入れる。

【０３８４】

システム６００は、ユーザがスマートカードをリーダ１へと挿入し、最初のボタン押下を開始するときには、アプリケーション４９２０を開始する準備ができている。

【０３８５】

１３．５　第１のスマートカードサービスの開始

本節では、図４８のソフトウェアアーキテクチャを組み込むシステム６００上で他のサービスが実行中でない場合にスマートカードサービスを開始するプロセスを説明する。これは、システムが最初に開始されるときの状況であり、タイムアウトを介して、あるいは、ユーザがスマートカード１０を挿入せずにリモート操作装置１に接触したためにサービスが終了する場合にも発生する。

（ｉ）ユーザがスマートカード１０をリーダ１へと挿入し、タッチパネル８を押下する。

（ｉｉ）押下イベントがイベントマネージャ４９０４に送信され、イベントマネージャ４９０４はパケットの形式を変更し、ランチャ４９１０へと転送する。

（ｉｉｉ）ランチャ４９１０はパケットを受信し、いずれのサービスもアクティブでないことを認識し、スマートカード１０のサービス識別子（ベンダ識別子及びアプリケーション識別子）及びサービス固有識別子（カード識別子）を伴ってディレクトリサービス４９１２に問合せを行なう。

（ｉｖ）問合せは適切なアプリケーション４９２０の位置を戻し、この位置をランチャ４９１０が取り込む。アプリケーション４９２０は、一般的に、ネットワーク２２０のいずれかの場所のサーバコンピュータ上の記憶装置からリモートで供給されるが、ランチャ４９１０に対してローカルに実行されなければならない場合もある。高度なシステムでは、アプリケーションはランチャからリモートで実行されても良い。

（ｖ）ランチャ４９１０は、イベントマネージャ４９０４に新規のアプリケーション４９２０が開始されることを通知する。

（ｖｉ）アプリケーション４９２０は、実行されるランチャへのダウンロードを終了すると、ランチャ４９１０により開始される。

（vii）アプリケーション４９２０はイベントマネージャ４９０４との接続を開始し、イベントマネージャ４９０４がその接続を受け入れると、ランチャ４９１０に登録する。これは、アプリケーション４９２０がどのサービスグループの一部であるか及びアプリケーションがどの機能を実行可能であるのかを含む。
（viii）ランチャ４９１０は、新規のアプリケーション４９２０にフォーカスが得られることを通知する。
【０３８６】
この段階でアプリケーション４９２０は開始され、イベントを受信できる状態になる。リーダ１により生成されるＰＲＥＳＳメッセージ、ＲＥＬＥＡＳＥメッセージ、及びＭＯＶＥメッセージは、そのアプリケーションにより意図される限り、イベントマネージャ４９０４によりアプリケーション４９２０へと転送される。アプリケーション４９２０は、登録が完了するまでいかなる形であろうとイベントマネージャ４９０４と対話することができない。更に、イベントマネージャ４９０４はイベントをアプリケーションへと転送することはない。イベントマネージャ４９０４が登録されていないアプリケーション４９２０から受信するアプリケーション登録イベント以外のいかなるイベントも破棄されることになる。
【０３８７】
１３．６　アプリケーションの開始、制御、及び停止
図５６（ａ）及び図５６（ｂ）は、ソフトウェアアーキテクチャ４９００を組み込むシステム６００上でユーザにサービスを提供するために、複数のアプリケーション４９２０のうちの１つのアプリケーション（アプリケーション＃１～＃ｎ）を開始、制御、及び停止する方法５６００を示す。方法５６００のプロセスは、システム６００ＡのＣＰＵ２０５又はシステム６００ＢのＣＰＵ４３０５などのＣＰＵにより実行される。方法５６００を示すソフトウェアプログラムは、システム６００ＡのＣＤ－ＲＯＭ２１２又はシステム６００Ｂのメモリ４３０６などのメモリ媒体に格納される。ユーザがスマートカード１０をリーダ１に挿入し、タッチパネル８を押下して所望の印を選択すると、リーダ１のＣＰＵ４５はスマートカード１０からカードヘッダ１１００及び選択された印と関連付けられるデータを読み、選択された印と関連付けられた押下イベント（例えば、押下メッセージ）をイベントマネージャ４９０４に送信する。イベントマネージャ４９０４はパケットの形式を変更する。イベントマネージャ４９０４は、押下パケット（例えば、EM-READER PRESS）をランチャ４９１０に送信する。ソフトウェアプログラムはＣＰＵにより実行されるが、このＣＰＵは、カードインタフェース（デーモン）４９０２がリーダ１から押下イベントを受信し、それをイベントマネージャ４９０４に送信すると、少なくとも同じ計算装置のカードインタフェース（デーモン）４９０２、イベントマネージャ４９０４、ランチャ４９１０、及びアプリケーション４９２０を実行する。これに対し、ソフトウェアプログラムが別々の計算装置の各ＣＰＵにより実行される場合、イベントマネージャ４９０４を実行する第１の計算装置の第１のＣＰＵがステップ５６０３から５６０８を実行し、少なくともランチャ４９１０及びアプリケーション４９２０を実行する第２の計算装置の第２のＣＰＵがステップ５６０９から５６３６を実行する。
【０３８８】
ステップ５６０３において、イベントマネージャ４９０４を実行することによって、ＣＰＵはカードインタフェース４９０２を介してリーダ１から押下イベントを受信し、次のステップ５６０５において、押下イベント中のサービス識別子（ベンダ識別子及びアプリケーション識別子）が既に実行中のアプリケーション４９２０のうちのフロントアプリケーション（例えば、アプリケーション＃１）のサービス識別子と一致するか否かを判定する。次のステップ５６０５においてマッチングテーブルを使用してサービス識別子がフロントアプリケーション（例えば、アプリケーション＃１）のサービス識別子と一致すると判定される場合、次のステップ５６０８においてイベントマネージャ４９０４を実行することによって、ＣＰＵは押下パケットをフロントアプリケーションへと転送し、方法５６００は終了する。アプリケーション４９２０の各アプリケーションと対応するサービス識別

子との間の関係を有するテーブルが、メモリ２０６又はメモリ４３０６のＲＡＭに格納される。ステップ５６０５において、サービス識別子がフロントアプリケーションのサービス識別子と一致しないと判定される場合、次のステップ５６０７において、ＣＰＵはイベントマネージャ４９０４からランチャ４９１０へと押下パケットを転送する。次のステップ５６０９において、ランチャを実行することによって、ＣＰＵはサービス識別子を伴ってディレクトリサービス４９１２に問合せを行ない、サービス識別子に対応する新規のアプリケーション（例えば、アプリケーション＃２）の位置を受信する。次のステップ５６１１において、ランチャ４９１０を実行することによって、ＣＰＵはその位置から新規のアプリケーションを取り込む。次のステップ５６１３において、ランチャ４９１０を実行することによって、ＣＰＵは新規のアプリケーション（例えば、アプリケーション＃２）を実行する。次のステップ５６１５において、ＣＰＵは新規のアプリケーションとイベントマネージャ４３０６との間の接続を開始する。イベントマネージャ４３０６が接続を受け入れると、ＣＰＵは新規のアプリケーションをランチャ４９１０に登録し、アプリケーションはランチャ４９１０にどのサービスグループの一部であるかを通知する。次のステップ５６１６において、ＣＰＵはメモリ２０６又はメモリ４３０６のＲＡＭに格納されたサービスグループテーブルを使用して、新規のアプリケーションが現在実行中のアプリケーションとサービスグループを共有するか否かを判定する。各サービス識別子と対応するサービスグループとの間の関係を有するテーブルは、メモリ２０６又はメモリ４３０６中のメモリのＲＡＭに格納される。例えば、テーブルにおいて、サービス識別子１（アプリケーション＃１）及びサービス識別子３（アプリケーション＃３）がサービスグループＡに対応し、サービス識別子２（アプリケーション＃２）及びサービス識別子４（アプリケーション＃４）がサービスグループＢに対応する。次のステップ５６１６において、新規のアプリケーションが現在実行中のアプリケーションとサービスグループを共有すると判定される場合、次のステップ５６３５においてランチャ４９１０を実行することによって、ＣＰＵは現在のアプリケーション（フロントアプリケーション）にフォーカスが失われることを通知する。次のステップ５６３６においてランチャ４９１０を実行することによって、ＣＰＵは新規のアプリケーションにフォーカスが得られることを通知し、方法５６００は終了する。この場合、ＣＰＵは現在のアプリケーション（フロントアプリケーション）の実行をバックグラウンドで継続するが、リーダ１からはイベントを受信しない。現在のアプリケーションを実行することによって、ＣＰＵは同報メッセージ及びメッセージを特定のアプリケーションに送信することができるが、サービスグループから外れることはできない。

ステップ５６１６において、新規のアプリケーションが現在実行中のアプリケーションとサービスグループを共有しないと判定される場合、ステップ５６１７においてランチャ４９１０を実行することによって、ＣＰＵは現在実行中のアプリケーションに終了するように通知し、タイムアウトを設定する。次のステップ５６２１においてランチャ４９１０を実行することによって、ＣＰＵはタイムアウトを待ち、新規のアプリケーションを除く残りのアプリケーションを終了させる。次のステップ５６２３においてランチャ４９１０を実行することによって、ＣＰＵはイベントマネージャ４９０４に終了した又は終了させられたアプリケーションを通知する。次のステップ５６３６においてランチャ４１０を実行することによって、ＣＰＵは新規のアプリケーションにフォーカスが得られることを通知し、方法５６００は終了する。この場合、ＣＰＵは新規のアプリケーションを実行中であり、EM-READER PRESS、EM-READER-RELEASE、及びEM-READEF MOVEなどのそのアプリケーション用の押下パケットを受信する。システム６００Ａ又は６００Ｂは１つだけアプリケーションを有する新規のサービスを実行中である。

【０３８９】

１３．７　２つのアプリケーション間でのデータの受け渡し

本節では、図４８のソフトウェアアーキテクチャを組み込むシステム６００上でデータグラムプロトコルを使用して２つのアプリケーション４９２０（アプリケーション＃１）及び４９２０（アプリケーション＃２）間でデータの受け渡しを行なうプロセスを説明する

。この方法では、送信側アプリケーション＃１が受信側アプリケーション＃２のアプリケーション識別子（Ｘｉｄ）を知っている必要がある。
（i）送信側アプリケーション＃１は送信を希望するデータを収集する。
（ii）送信側アプリケーション＃１はランチャ４９１０に現在のサービスグループで実行中のアプリケーションのリストを要求する。
（iii）ランチャ４９１０はアプリケーション＃１に現在のサービスグループ中の全てのアプリケーションのリストを送信する。このリストは、アプリケーションが提供した記述文字列のみならず、各アプリケーションがランチャ４９１０に実行可能であると通知した各機能を含む。このリストの順序では、直前のアプリケーションが先頭に列挙される。
（iv）送信側アプリケーション＃１は、データの適切な受信者の有無を調べる。受信者が存在しない場合、処理の進め方を決定するのはアプリケーション＃１の役割である。アプリケーション＃１は、例えば、データを送信しなくても良く、場合によっては、ユーザに別のスマートカードを挿入するように要求しても良い。カードの挿入により所要のアプリケーションが開始されるであろう。
（v）適切な受信者が存在する場合、送信側アプリケーション＃１はイベントマネージャ４９０４を介して受信側アプリケーション＃２にデータを送信する。
（vi）イベントマネージャ４９０４はメッセージヘッダをチェックして送信側アプリケーション＃１がデータ長及び送信者フィールドを正しく記入していることを確認し、メッセージを受信側アプリケーション＃２に渡す。このようなアプリケーション＃２が実行されていない場合、イベントマネージャ４９０４はメッセージを破棄し、送信側アプリケーション＃１にエラーメッセージを戻す。

【０３９０】

１３．８　掲示板へのデータの掲示

本節では、ソフトウェアアーキテクチャ４９００を組み込むシステム６００上で共通の掲示板にデータを掲示するプロセスを説明する。
（i）ポスティングアプリケーション４９２０は掲示板に掲示することを希望するデータを収集する。
（ii）ポスティングアプリケーション４９２０はデータをその簡潔な記述と共にイベントマネージャ４９０４に送信する。

【０３９１】

１３．９　掲示板からのデータの取得

本節では、ソフトウェアアーキテクチャ４９００を組み込むシステム６００上で別のアプリケーションにより予め掲示板に掲示されているデータを取得するプロセスを説明する。
（i）要求元アプリケーション＃２は、イベントマネージャ４９０４に掲示板上のメッセージのリストを要求する。
（ii）イベントマネージャ４９０４は、アプリケーション＃２に掲示板上のメッセージのリストを送信する。このリストは、データの簡潔な記述、メッセージを掲示板に掲示したアプリケーション４９２０に対するアプリケーション識別子（Ｘｉｄ）、及び掲示板上の全てのメッセージに対するメッセージ識別子を含むであろう。
（iii）アプリケーション＃２は、イベントマネージャ４９０４にメッセージ識別子を用いて特定のメッセージを要求することも、ランチャ４９１０に現在実行中の全てのアプリケーションのリストを要求することもできる。
（iv）アプリケーション＃２が実行中のアプリケーションのリストを要求した場合、ランチャ４９１０はそれをアプリケーション＃２に送信する。このリストは、アプリケーション識別子（Ｘｉｄ）と対応するアプリケーションがランチャ４９１０に実行可能であると報告した機能のリストとを含むであろう。
（v）要求元アプリケーション＃２は、求める機能を実行するアプリケーションからの全て又は一部のメッセージを見つけることができる。

【０３９２】

１３．１０　掲示板からのデータの削除

本節では、ソフトウェアアーキテクチャ４９００を組み込むシステム６００上で同じアプリケーション又は別のアプリケーションにより予め掲示板に掲示されているデータを削除するプロセスを説明する。
（i）要求元アプリケーション＃２は、イベントマネージャ４９０４に掲示板上のメッセージのリストを要求する。
（ii）イベントマネージャ４９０４は、アプリケーション＃２に掲示板上のメッセージのリストを送信する。このリストは、データの簡潔な記述、ポスティングアプリケーションのアプリケーション識別子（Ｘｉｄ）、及び掲示板上の全てのメッセージに対するメッセージ識別子を含むであろう。
（iii）アプリケーション＃２は、メッセージ識別子を指定することによってイベントマネージャ４９０４に特定のメッセージを削除するように要求する。
【０３９３】
１３．１１　適用例
例Ａ：　カード配列
実施される可能性のあるアプリケーションカード配列は幾つも存在する。アーキテクチャ４９００では、アプリケーション４９２０に対してどのカード配列又はカード配列の組み合わせが採用されるかに関して制約がない。
【０３９４】
図５１Ａに示すサービスグループ内の順次カード配列では、アプリケーションの特定の集合に対するスマートカード１０は指定の順序で挿入される必要がある。例えば、カードＡの後には、カードＢ、カードＣが続き、抜き取り及び／又は再挿入の場合でも同じ配列に従う。
【０３９５】
サービスグループ内の階層カード配列では、アプリケーションの特定の集合に対するカードは図５１Ｂに示すようにツリー状に挿入される必要がある。カードＡが挿入される場合、カードＢ又はカードＣのみを挿入しても良い。カードＢが抜き取られる場合、カードＡを再挿入しなければならない。カードＣが挿入される場合、カードＤのみが挿入されても良く、カードＤが抜き取られる場合、カードＣのみが挿入されても良い。
【０３９６】
サービスグループにおける完全網目状カード配列により、アプリケーションの集合に対するカードを任意の順序で挿入・使用できる。
【０３９７】
例Ｂ：　ピザ注文サービス
従来のピザ注文アプリケーションでは、ピザの種類に関して複数の選択肢（vegetarian、supreme、及びmeat loversなど）が提示されているが、トッピングのカスタマイズ又は特別割引の利用に対する機能は提供されていない。
【０３９８】
アーキテクチャ４９００の下でJoe's Pizzeriaサービスグループを構成するアプリケーションの集合の一例は以下のようになるであろう：
（i）Joe's Pizzaメニュー、
（ii）トッピングスペシャリスト、
（iii）現在の特別割引、及び
（iv）個人識別子
これらのアプリケーションの各々は、他のアプリケーションと協働し、フル機能のピザ注文サービスを創設するように仕向けることができる。Joe's Pizzaメニューアプリケーションは、ピザの種類（vegetarian、supremeなど）、ドリンク（コーラ、ライムなど）、及びサイドメニュー（ガーリックブレッド、パスタなど）を顧客が選択することができるユーザインタフェースを提供する。また、このアプリケーションは、現在の注文の買い物かご形式のリストを保持し、注文をリセットするためのボタン及び完了するためのボタンをスマートカードに設ける。

【０３９９】
トッピングスペシャリストアプリケーションは、顧客が現在注文予定のピザのリスト内を移動し、選択したピザにカードの表面に印刷された一連のトッピングの中からトッピングを追加／削除できるユーザインタフェースを提供する。注文可能なピザのリストは実行中のJoe's Pizzaメニューアプリケーションから得られる。ピザのトッピングの変更は、ピザの注文の修正のためにJoe's Pizzaメニューアプリケーションへと伝えられるであろう。
【０４００】
現在の特別割引アプリケーションは、Joe's Pizzeriaで利用可能な現在の特別割引のリスト内を移動するためのコントロールを提供する。選択される特別割引は、現在の注文に適用されるようにJoe's Pizzaメニューアプリケーションへと伝えられる。
【０４０１】
個人識別子アプリケーションは、ユーザが提供を希望する詳細によって、ユーザの自宅の住所及び自宅の電話番号をJoe's Pizzaメニューアプリケーションへと選択的に伝える方法を提供する。
【０４０２】
例Ｃ：　写真現像サービス
従来の写真アルバムアプリケーション及びＴシャツアプリケーションでは、現在選択されている写真の送受信のためにクリップボードが共有される（ファイルとして）。しかし、写真の修整（例えば、トリミング又は輝度の向上）のための機能、あるいは、複数の連結されたカードを有する機能はない。これらの連結されたカードは１本のフィルム全体を表し、各カードは現在最大２０枚の写真のみを含み、各写真はボタンとして機能するのに十分な大きさのアイコンにより表される。
【０４０３】
アーキテクチャ４９００に関して、写真現像サービスは以下の１組のカードを有するように設計されても良い：
（i）Ｆｉｌｍ＿１ａ、
（ii）Ｆｉｌｍ＿１ｂ、
（iii）Ｔシャツプリンタ、及び
（iv）フォトエンハンサ
Ｆｉｌｍ＿１ａカード及びＦｉｌｍ＿１ｂカードは、それぞれ４０枚の写真を含む１本のAdvantix（Kodak Corp, USAの登録商標）全体を表す。いずれのカードを最初に挿入しても良い。いずれかのカードが一度挿入されると、挿入されたカードの表面に印刷された写真への直接アクセスを用いることで双方のカードにわたる写真一式へのアクセス権が提供される。これは、スライドショー機能が双方のカードに対応する写真を循環させるであろうことを意味する。各カードは、写真現像サービスグループ中の別のアプリケーションを用いてユーザ用のサービスグループクリップボードに特定の写真参照を追加するためのボタンを有し、アプリケーションは現在閲覧中の写真への参照を戻す機能を提供するであろう。
【０４０４】
Ｔシャツプリンタアプリケーションは、直前に見た写真（フィルムアプリケーションから得られるものへの参照）を使用してＴシャツ転写物をインスタントで印刷するか、あるいは、クリップボードにある１組の写真からＴシャツ転写物を作成する機能を提供する。
【０４０５】
簡易な写真編集サービスの一部として、フォトエンハンサアプリケーションは、直前に見た写真（Ｔシャツアプリケーション及びフィルムアプリケーションのうちの直前にフォアグラウンドにあった方から得られる）に対して動作する。フォトエンハンサは、自動トリミング、鮮鋭化、ぼかし、淡色化、暗色化などの操作を提供しても良い。このとき、変更をフォトサーバに戻して永続化することができる。
【０４０６】

例Ｄ：　ビデオ電子メールサービス
従来のビデオ電子メールアプリケーションは、カードの表面に記載されるビデオ電子メールユーザにビデオ電子メールメッセージを送信する手段を提供する。設計を幾らか変更すれば、アーキテクチャ４９００によるビデオ電子メールサービスを創設することが可能である。このサービスでは、スマートカード名刺をアドレス帳の所有者に供給する複数ユーザからアドレス帳を編纂することができる。ビデオ電子メールサービスを形成するアプリケーションは以下の通りである：
（i）ビデオ電子メール送信、
（ii）ビデオ電子メールメールボックス、
（iii）ビデオ電子メールアドレス帳、及び
（iv）名刺
ビデオ電子メール送信アプリケーションは、挿入される個人識別カード又は挿入される名刺からアドレスが得られる点を除き、従来のアプリケーションとほぼ同様に動作する。
【０４０７】
ビデオ電子メールメールボックスアプリケーションは、リモートサーバからビデオ電子メールメッセージを取得する機能を提供すると共に、ビデオ電子メール送信アプリケーションにより返信アドレスとして使用される送信者のアドレスを提供する。
【０４０８】
アドレス帳機能は、ビデオ電子メールアドレス帳アプリケーションにより提供される。このアプリケーションにより、ユーザは挿入される様々な名刺、個人識別子カード、又はビデオ電子メールメールボックスカードからアドレスのリストを増強することができる。ビデオ電子メール送信アプリケーションにより使用するため、アドレスのリストから１つ以上のエントリが選択されても良い。
【０４０９】
例Ｅ：　買い物かごサービス
従来のソフトウェアアーキテクチャでは、オンラインショッピングを提供するアプリケーションは、買い物かご手段、注文手段、料金請求手段、及び発送手段を含む独自の購入システムを維持する必要があった。アーキテクチャ４９００の一部として利用可能な特徴を利用する買い物かごサービスは、これらの機能が各オンラインショッピングアプリケーションから分離できるようにし、他の機能のためのユーザインタフェース領域を広げるであろう。このような買い物かごサービスの一部を形成するであろうアプリケーションは以下の通りである：
（i）電子配送買い物かご、
（ii）Davy Jonesオンライン、及び
（iii）Pace Bros.オンライン
電子配送買い物かごアプリケーションは、総合買い物かご管理機能、支払機能、及び注文機能を提供する。
【０４１０】
Davy Jonesオンラインアプリケーション及びPace Bros.オンラインアプリケーションは、対応するデパートから仕入れ可能な品目のリストを関連する品目記述と共に閲覧する機能を提供する。購入を希望する品目が見つかったときには、電子配送買い物かごアプリケーションを経由してその品目を買い物かごに追加し、注文・配送に備えることができる。
【０４１１】
先の説明から明らかなように、管理プロセスの部門分け及びアプリケーションの適切な起動を介して柔軟性の拡張を許容するカードインタフェースシステムを実現するのにアーキテクチャ４９００を使用しても良い。これにより、アプリケーションは機能的結果を達成するために協力的に動作することができる。更に、複雑度に見合ったプラットフォームで手順を動作させる能力を介して、種々の複雑度のハードウェアプラットフォームからアーキテクチャ４９００の種々のコンポーネントを動作させることができる。このようなプラットフォームは、処理能力の限られたローエンドのセットトップボックスから、家庭用Ｐ

Ｃ及びリモートサーバコンピュータにまで及ぶ。特に、「ダム」セットトップボックスの場合、カードデーモン４９０２はセットトップボックスの内部から１つ以上のリモートサーバコンピュータの全てのプロセスのバランスにより実行されるであろう。これに対し、スマートセットトップボックス又は家庭用パーソナルコンピュータの場合、全てのプロセスは、ネットワーク２２０を介する外部通信が必要不可欠な場合を除き、１つのハードウェアの内部から動かされるであろう。

【０４１２】

また、アーキテクチャ４９００は、不正なデータ吸上げ及び詐欺行為からユーザ及びベンダの双方を保護するために、特定のアプリケーションに適した機密保護モデルをサポートするように拡張することができる。

【０４１３】

イベントコマンドの導管として作用するイベントマネージャ４９０４により、アーキテクチャは通信プロトコルのある範囲のバージョンにわたって開発されたアプリケーションと共に動作することができる。このようなアーキテクチャは、通常、時の経過と共に開発される可能性があるからである。

【０４１４】

アーキテクチャ４９００により、カードインタフェースシステム６００は、カードアプリケーションが予期される動作モードに従っていないときでも機能を継続することができる。これは、アプリケーションの突然の終了、コマンドによる終了の拒否、及びシステム６００への不正な又は余分なデータの送信を含む。アーキテクチャ４９００は、同じサービスグループに所属するアプリケーションの各カードによってマルチカードアプリケーションをサポートし、それにより、カードが抜き取られて新規のカードが挿入されるときにアプリケーション実行の維持が保証される。

【０４１５】

１３．１２　アプリケーション管理システム

コンピューティングリソースを過負荷状態にしたり、適切な応答を保証したりすることなく複数のアプリケーションが同時に動作可能であるようにするために、サービスグループの概念、その確立、及びその消滅を利用してアーキテクチャ４９００の説明を行なってきた。

【０４１６】

複数のアプリケーションを考慮する際の代替のアプローチは、アプリケーション間のデータフローがデータの作成者からデータの消費者への方向であると解釈することから生じる。図５５は有向グラフを示し、グラフ方向は、群機能を実行するために消費者から作成者へと向かう。この場合、Ｔシャツは名前と表面に転写される写真とを有し、そのデータは複数の他のアプリケーションから得られる。このようなグラフ構造内のアプリケーションの管理は、グラフの各ノードのアクセス可能性によって決まる。具体的には、グラフ中のノードが到達不可能になると、そのノードにおけるアプリケーションは認識している機能を実行することができないので、そのアプリケーションは終了させられるべきである。更に、そのアプリケーションの生成物の消費者がそのサービスに対する登録を解除するときには、ノードへのリンクは削除されるべきである。アプリケーションは開始されるとツリーに配置される。アプリケーションが消費者が望む種類の作成者である場合、アプリケーションはツリー中のそのノードの下に配置される。

【０４１７】

上述のように、アプリケーション４９２０は、アーキテクチャ２００に関して先に説明したサービス識別子と同等の対応するベンダ識別子及びアプリケーション識別子により参照される。アプリケーション識別子（すなわち、Ｘｉｄ）は、既に実行中のアプリケーションを起動する際の迅速なマッチング用のユニークキーとして使用される。アプリケーション識別子は、ベンダ識別子及びカード識別子（カード識別子はアーキテクチャ２００を参照して先に説明したサービス固有識別子と同等である）と共にカード上に格納されるものと、開始されるときにシステムによりアプリケーションに割り当てられるものとの２つが

存在する（後者のアプリケーション識別子は、ここではコンポーネント識別子又はＸｉｄとも呼ばれ、前者のアプリケーション識別子は上述のようにサービス識別子に関連する）。

【０４１８】

各アプリケーションは、識別のためにＸｉｄを使用することで必要に応じて機能の作成者又は消費者として登録しても良い。アプリケーションは、ユーザインタラクションを介して特定の時点で何が必要かを認識している。例えば、ユーザはアプリケーション内を「写真追加」画面まで移動する。ここで、アプリケーションはフォト消費者として登録しても良い。この点に関して、サービスグループアプローチは実際には大まか過ぎるので、登録はサービスグループではなく機能に基づいて行なわれるのが好ましい。更に、このアプローチでは、サービスグループ中の全てのアプリケーションがそのサービスグループにより提供される全ての機能をサポートしない限り、作成者が必要とする特定のサービスを提供することができない可能性があるときに、アプリケーションは消費者／作成者関係において別のアプリケーションにリンクすることができない。

【０４１９】

アプリケーションは他のアプリケーションから２つ以上の機能を要求する可能性があるので、このようなモデルは実現用の２つのオプションを提示する。

【０４２０】

１．グラフ中の各ノードは、他のどのノードに対しても接続を１つだけ有する。これは、接続が消費者／作成者関係に含まれるサービスのリストも含まなければならないことを意味する。消費者がサービスに対する登録を解除するたびにリストエントリが削除される。接続用のサービスのリストが空になると、接続は削除される。接続が削除されると、その接続によりリンクされる作成者もチェックを受ける。作成者ノードが既に他のノードに接続していない場合、そのノードは削除されるであろう。

【０４２１】

２．このオプションは、サービスのリストを保持する代わりに、各サービスが消費者／作成者ノード間における別々の接続である点を除いては（１）と同様である。このため、２つのアプリケーション間には複数の接続が存在するであろう。消費者がサービスに対する登録を解除するとき、その接続は削除される。作成者が既に接続されていない場合、消費者は終了する。

【０４２２】

この提案は、それぞれが現在リーダ１に挿入されているスマートカードと関連付けられるアプリケーションを特定のユーザ動作以外のイベントにより終了させることができる点において問題である。これは、ユーザの観点から混乱を招く恐れがある。従って、アプリケーションの終了に対する代替のアプローチが望まれる。

【０４２３】

このような代替のアプローチでは、アーキテクチャ４９００は、上述のように特定のサービスグループのメンバである任意のアプリケーション４９２０に特に依存することなく動作しても良いが、「主」サービスグループと呼ばれるグループの一時情報を介して動作する。主サービスグループは、任意のアプリケーション４９２０が、作成者である場合、消費者である場合、作成者兼消費者である場合、及び作成者でも消費者でもない場合のいずれに分類されるかによって判定される２つ以上の現在のアプリケーション間の一時機能的関係から生じる。

【０４２４】

アプリケーション４９２０に対するこのような管理システムは、同じサービスグループ中のアプリケーションの作成者／消費者対又はアプリケーションが双方の基準を満たす場合には単一のアプリケーションが登録されるときに形成される「主」サービスグループの概念を中心とする。例えば、アプリケーションＡｃ及びＡｐが同時に動作するときにはサービスグループＡが主となり、作成者－消費者対に適合する。一方、作成者－消費者対に適合するＡｃＢｐ又はＡｐＢｃは主サービスグループを作成しない。この管理システムによ

ると、主サービスグループが形成されるとき、そのグループを共有しない全てのアプリケーションが終了させられる。双方が同時に登録される場合、主サービスグループは、第２の主サービスグループと共に存在しても良い。例えば、アプリケーション＃１が開始されてＡｐＢｐを登録し、アプリケーション＃２が開始されてＡｃＢｃを登録する場合、Ａ及びＢは主になる。２つ以上の主サービスグループが存在するためには、開始される新規のアプリケーションが作成者－消費者対を確立する各グループに登録するときにそれらのサービスグループが形成されなければならない。サービスグループを形成するアプリケーションの作成者／消費者対は、主サービスグループが主グループの「従属者」になった後に登録される。従属グループの従属グループが形成されても良い。従属者の従属者は、従属者の作成者が既に第２の従属者に対する消費者として登録されているときに形成される。

【０４２５】

このような管理構造の最終結果は、ユーザが望む最終結果を達成するためにデータの受け渡しを行なう相互作用アプリケーションのツリー又はグラフの作成及び解体である。具体的には、このような結果は、ピザ注文サービスに対する上述の例Ｂとは対照的に、利用されるアプリケーションのフェースから容易に明らかにならない可能性がある。このアプリケーション管理構造は、以下の例を参照することで最も良く説明される。

【０４２６】

以下の例は、複数のアプリケーションを参照する。その詳細が以下の表４に記載される。

【０４２７】

【表４】

表4

カードアプリケーション名	記述	サービスグループメンバ（ｐ＝作成者、ｃ＝消費者、ｎ＝どちらでもない）
ＩＤ１	識別詳細カード	Ｚｐ　Ｃｐ
ＩＤ２	識別詳細カード	Ｚｐ　Ｑｐ
ＰｈｏｔｏＩＤ	写真識別カード	Ｚｐ　Ｑｐ　Ａｐ
Ｐｈｏｔｏ１	写真カード	Ａｐ　Ｆｐ
Ｐｈｏｔｏ２	写真カード	Ｍｐ　Ａｐ
ＰＩＮ	個人識別番号カード	Ｐｐ
Ｂａｎｋ	エレクトロニックバンキングカード	Ｂｎ
Ｐｉｚｚａ	ピザ注文カード	Ｒｎ
Ｔ－ｓｈｉｒｔ	Ｔシャツ製造	Ｔｐ
ＣａｒｄＭａｋｅｒ	他のカード作成に使用されるカード	Ｓｐ

【０４２８】

例Ｆ：

この例では、ユーザはカード上に受信者の名前、定番のメッセージ、及び写真を有するグリーティングカードを作成することを望む。表４のカードを使用する第１のステップは、ユーザがＣａｒｄＭａｋｅｒアプリケーションカードをリーダ１へと挿入することであろう。このような動作によりアプリケーションが開始され、サービスグループＡ及びＺの消費者として登録される。アプリケーションは、そのサービスグループへの所属を動的に変更しても良い。例えば、ＣａｒｄＭａｋｅｒが開始され、ユーザに以前に作成したカードと同一のカードの作成を希望するか否かを尋ねる画面表示を提示しても良い。「ＮＯ」と回答すると、新規のカードが作成されることになるので、ＣａｒｄＭａｋｅｒはＩＤ１及びＰｈｏｔｏ１に消費者として登録する。この段階におけるプロセスツリーは図５３Ａに示される。次に、ユーザは写真が要求されていることを認識し、ＣａｒｄＭａｋｅｒアプリケーションを抜き取り、Ｐｈｏｔｏ１アプリケーションを挿入することによってその写真を提供する。ＣａｒｄＭａｋｅｒアプリケーションはまだ機能を実行しなければならないので、リーダ１から抜き取っても動作を継続する。Ｐｈｏｔｏ１アプリケーションの挿

入により、図示するようにＡｃ及びＡｐに主サービスグループをクレートに詰めるが、これは、ＣａｒｄＭａｋｅｒアプリケーションが写真を要求し、Ｐｈｏｔｏ１アプリケーションがその写真を供給することができることを意味する。Ｐｈｏｔｏ１アプリケーションは、写真にアクセスするためにＰＩＮを必要とするので、構成は図５３Ｂに表すようになる。Ｐｈｏｔｏ１上の全ての写真が使用のためのロック解除にＰＩＮを必要とする訳ではないので、考慮中の場合のように、処理を進めるのにＰＩＮを必要とするときにはＰｈｏｔｏ１はＰｃとしてのみ登録する。図５３Ｃによると、続いてＰＩＮカードが提供される。図５３Ｃから明らかなように、第２の作成者－消費者対が形成され、この場合、ＰＩＮの提供によりＰｈｏｔｏ１カードはユーザにより選択された写真をＣａｒｄＭａｋｅｒアプリケーションに供給することができる。これらのタスクが完了すると、プロセスツリーの左側の枝が消去され、図５３Ｄに示すように、対応する「実行済」アプリケーションがイベントマネージャ４９０４から登録を解除する。プロセスを完了する次のステップは、所望の名前を有するカードを挿入することである。ここでは、この名前は、図５３Ｅに示すアプリケーションＩＤ１から得られる。このアプリケーションが所望の名前を供給するので、ＣａｒｄＭａｋｅｒアプリケーションは満足し、それにより、他の全てのアプリケーションは登録を解除し、終了できるようになる。ＣａｒｄＭａｋｅｒアプリケーションは他のアプリケーションとのインタラクションなしに所要のカードを出力することができる。

【０４２９】

代替のアプローチでは、写真及び名前の双方にアクセスするのにＰＩＮアプリケーションが必要とされても良い。双方の写真カードに対するＰＩＮが同じである場合に限り、ＰＩＮアプリケーションカードは１度だけの挿入で良い。図５４に示すようなプロセスツリーが形成されるであろう。この例では、ＰｈｏｔｏＩＤは作成されるカードの受信者の写真を有しても良く、Ｐｈｏｔｏ１は写真に重ねる魅力的な背景写真を有しても良いので、ＰｈｏｔｏＩＤ及びＰｈｏｔｏ１が使用される。

【０４３０】

図５４は、プロセスツリー中のノードに対する複数のリンクが許容され、到達不可能なノード（リンクなしのノード）上のアプリケーションが終了させられることを例証する。

【０４３１】

実行中のアプリケーションに対する上限は７に設定されるのが好ましい。この数を超える場合、アプリケーションの終了によりプロセスツリー中の最古の葉アプリケーションが開始される。

【０４３２】

以上の記述では、本発明の幾つかの構成及びこれらの構成に対する変形を説明しているに過ぎず、本発明の趣旨から逸脱することなく、変形及び／又は変更を行なうことができる。各実施形態は例示のためのものであり、限定することを意図しない。

【図面の簡単な説明】

【図１】　図１は、読取り装置及び関連するカードの斜視図である。

【図２】　図２は、図１に示すカードの反対側の斜視図である。

【図３】　図３は、図１に示すカードを線III－IIIに沿って切断したときの長手方向の横断面図である。

【図４】、

【図５】　図４及び５は、図１に示すカードの代替のカード構成の裏面の斜視図である。

【図６（ａ）】　図６（ａ）は、カードインタフェースシステムのハードウェアアーキテクチャを示す図である。

【図６（ｂ）】　図６（ｂ）は、カードインタフェースシステムの別のハードウェアアーキテクチャを示す図である。

【図７】　図７は、図６（ａ）及び図６（ｂ）の汎用コンピュータの概略ブロック図である。

【図８】　図８は、カードインタフェースシステムアーキテクチャの概略ブロック図であ

る。
【図９】　図９は、カードインタフェースシステムの概略ブロック図である。
【図１０】　図１０は、図１のリーダの内部構成を示す概略ブロック図である。
【図１１】　図１１は、図１のカードに格納されたカードヘッダのデータ構造を示す図である。
【図１２】　図１２は、図１１のヘッダの各フィールドの記述を示す図である。
【図１３】　図１３は、図１１のヘッダに含まれる各フラグの記述を示す図である。
【図１４】　図１４は、図１のカードに対するオブジェクト構造の各フィールドの記述を示す図である。
【図１５】　図１５は、図１４のオブジェクト構造に対するフラグの記述を示す図である。
【図１６】　図１６は、図１４のオブジェクト構造に対する各オブジェクト種別の記述を示す図である。
【図１７】　図１７は、図１４のオブジェクト構造によるユーザインタフェースオブジェクト構造に対する各フィールドの記述を示す図である。
【図１８】　図１８は、図１４のオブジェクト構造による各ユーザインタフェースオブジェクトフラグに対する記述を示す図である。
【図１９】　図１９は、図１のリーダから送信されるメッセージヘッダの形式を示す図である。
【図２０】　図２０は、図１９のヘッダに対するメッセージイベント種別を列挙するテーブルを示す図である。
【図２１】　図２１は、単純なメッセージの形式を示す図である。
【図２１（ａ）】　図２１（ａ）は、ＩＮＳＥＲＴメッセージの形式を示す図である。
【図２２】　図２２は、ＭＯＶＥメッセージの形式を示す図である。
【図２３】　図２３は、ＰＲＥＳＳメッセージ及びＲＥＬＥＡＳＥメッセージの形式を示す図である。
【図２４】　図２４は、図６のシステム内のメッセージの流れを示すデータフロー図である。
【図２５】　図２５は、図１のリーダにより実行される読取り方法を示すフローチャートである。
【図２６】　図２６は、図２５の方法の間に実行される図６のシステムを初期化する方法を示すフローチャートである。
【図２７】　図２７は、図２５の方法の間に実行される図１のカードをチェックする方法を示すフローチャートである。
【図２８】　図２８は、図２５の方法の間に実行される図１のリーダのタッチパネルを走査する方法を示すフローチャートである。
【図２９】　図２９は、図２５の方法の間に実行される１０ミリ秒待機方法を示すフローチャートである。
【図３０】　図３０は、図６のシステムのイベントの概要を示すフローチャートである。
【図３１】　図３１は、図３０のプロセスの間にイベントマネージャにより実行されるプロセスを示すフローチャートである。
【図３２】　図３２は、図３０のプロセスの間に実行される新規のアプリケーションを開始するための方法を示すフローチャートである。
【図３３】　図３３は、図３０のプロセスの間に実行されるアプリケーションを終了する方法を示すフローチャートである。
【図３４】　図３４は、永続的アプリケーションに対する現在のセッションを終了する方法を示すフローチャートである。
【図３５】　図３５は、フォーカス変更を実行する方法を示すフローチャートである。
【図３６】　図３６は、ランチャにより実行される方法の概要を示すフローチャートである。

【図３７】　図３７は、図３６の方法の間に実行されるアプリケーションを変更する方法を示すフローチャートである。
【図３８】　図３８は、図３６の方法の間に実行される新規のアプリケーションを登録する方法を示すフローチャートである。
【図３９】　図３９は、ランチャからイベントを受信するときにアプリケーションにより実行される方法を示すフローチャートである。
【図４０】　図４０は、ランチャからイベントを受信するときにブラウザコントローラアプリケーションにより実行される方法を示すフローチャートである。
【図４１】　図４１は、ブラウザアプリケーション方法を示すフローチャートである。
【図４２】　図４２は、システム６００のセットトップボックスをより詳細に示す概略ブロック図である。
【図４３】　図４３は、「ボトムエントリ」型のリーダの斜視図である。
【図４４】　図４４は、図４３のリーダの平面図である。
【図４５】　図４５は、図４３のリーダにカードを挿入するユーザを示す図である。
【図４６】　図４６は、カードが完全に挿入された後に図４３のリーダを操作するユーザを示す図である。
【図４７（ａ）】　図４７（ａ）は、図４４の線Ｖ－Ｖに沿って切断したときの長手方向の横断面図である。
【図４７（ｂ）】　図４７（ｂ）は、カードがリーダのレセプタクルに部分的に挿入された場合の図４７（ａ）と同様の図である。
【図４７（ｃ）】　図４７（ｃ）は、カードがリーダのテンプレートレセプタクルに完全に挿入された場合の図４７（ａ）と同様の図である。
【図４８】　図４８は、更なるカードインタフェースシステムアーキテクチャの概略ブロック図である。
【図４９】　図４９は、カードとアプリケーションとの関係を示す概略ブロック図である。
【図５０】　図５０は、アプリケーションとサービスグループとの関係を示す図である。
【図５１Ａ】、
【図５１Ｂ】、
【図５１Ｃ】　図５１Ａから図５１Ｃは、図４８のアーキテクチャ内の多様なカード配列を示す図である。
【図５２】　図５２は、図４８のアーキテクチャに対するスマートカードに格納される制御テンプレートデータを示す図である。
【図５３Ａ】、
【図５３Ｂ】、
【図５３Ｃ】、
【図５３Ｄ】、
【図５３Ｅ】　図５３Ａから図５３Ｅは、マルチカードアプリケーション構造の一例を示す図である。
【図５４】　図５４は、図５３Ａから図５３Ｅの目的を達成するための代替のアプローチを示す図である。
【図５５】　図５５は、マルチアプリケーション方法の有向グラフ表現を示す図である。
【図５６（ａ）】、
【図５６（ｂ）】　図５６は、アプリケーションを開始する方法を示す図である。
【図５７】　図５７は、図１１のカードヘッダに続く１つ以上のオブジェクト構造を示す図である。
【図５８】　図５８は、図８のディレクトリサービスにより実行されるプロセスの概要を示すフローチャートである。

【図１】

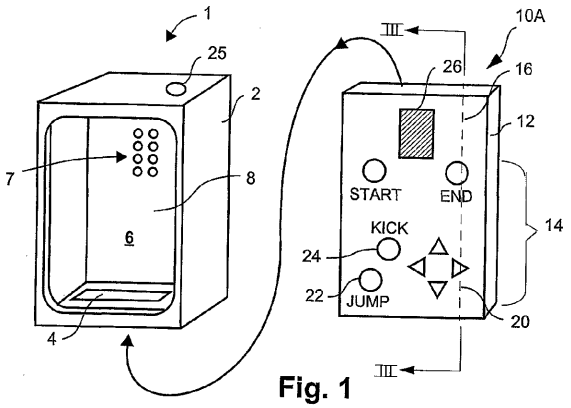


Fig. 1

【図２】

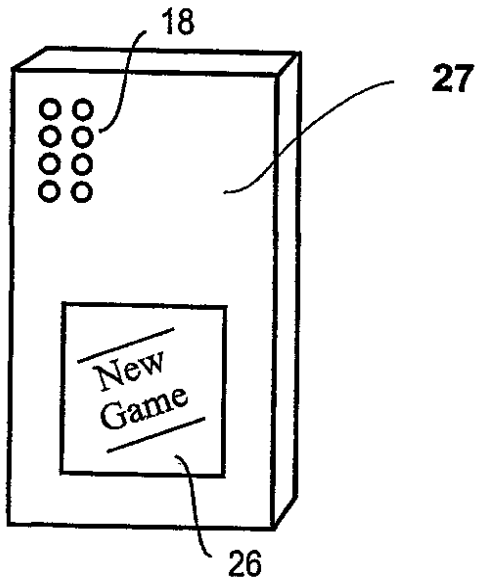


Fig. 2

【図３】

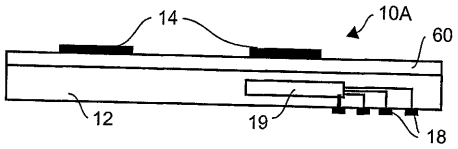


Fig. 3

【図４】

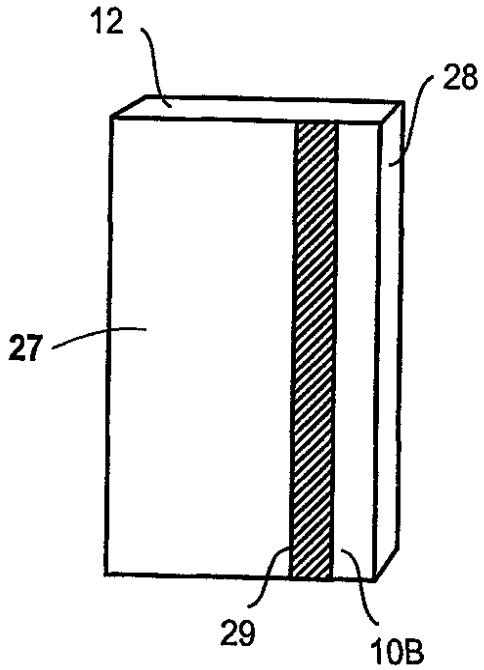


Fig. 4

【図５】

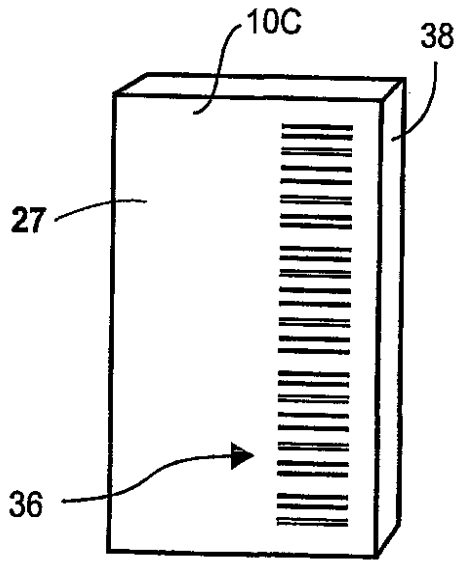


Fig. 5

【図６（ａ）】

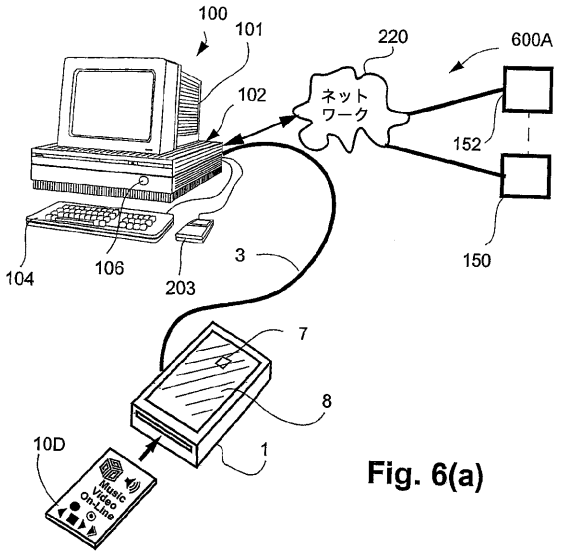


Fig. 6(a)

【図６（ｂ）】

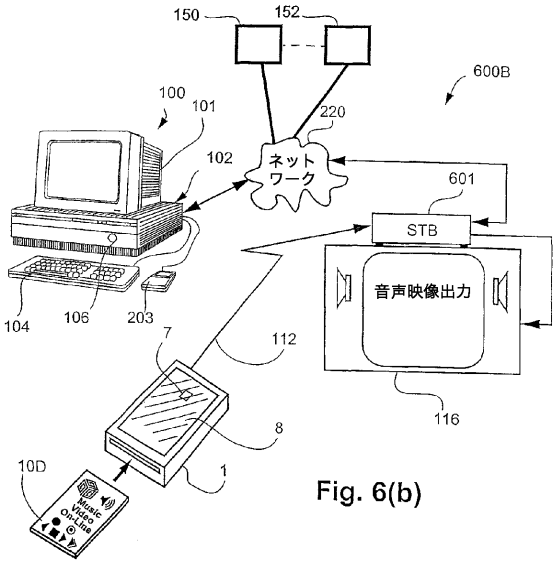


Fig. 6(b)

【図７】

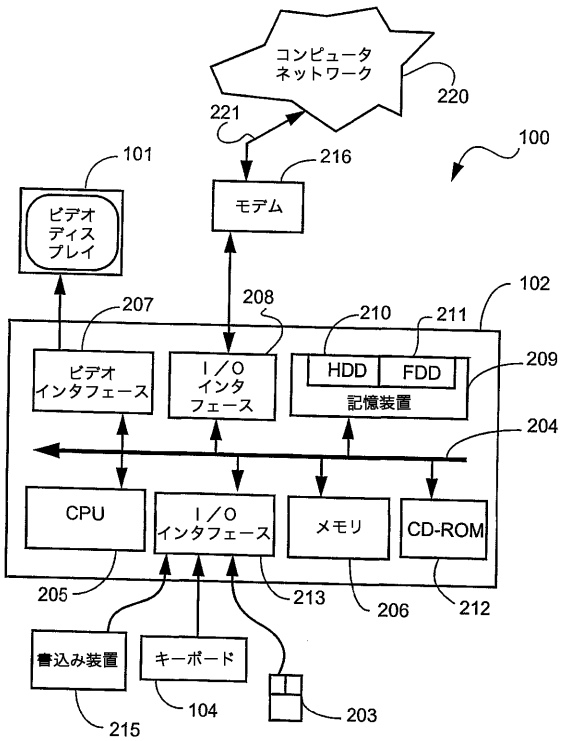


Fig. 7

【図８】

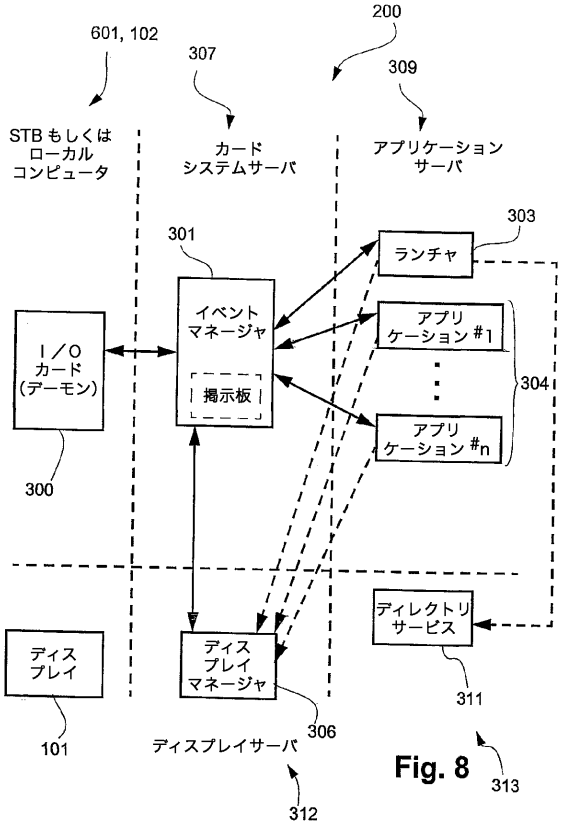


Fig. 8

【図９】

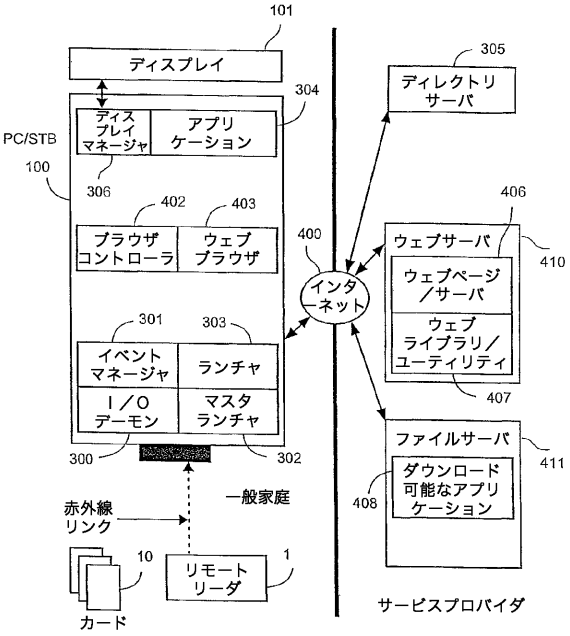


Fig. 9

【図１０】

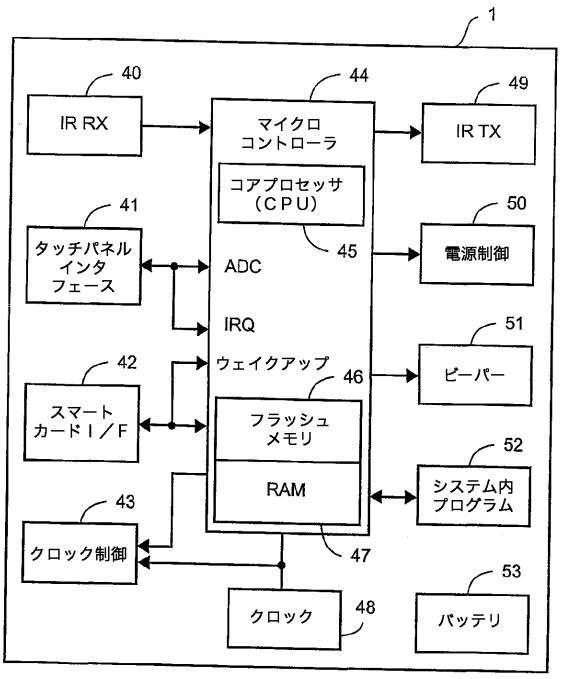


Fig. 10

【図１１】

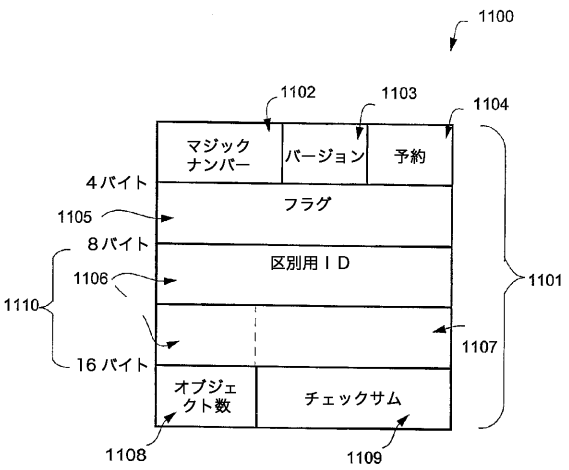


Fig. 11

【図１２】

フィールド番号	記述（カードヘッダ）
マジックナンバー	２バイトのマジックナンバー。 これを有効なカードとして指定する定数。 現在、「ｉ」のＡＳＣＩＩ値＋ 「C」のＡＳＣＩＩ値として定義される
バージョン	１バイトのバージョン番号。 各バージョン増加分はカードレイアウトの 下位のバージョンと互換性のあるリーダにより 読むことができないレイアウトの変更を指定する。 この文書ではカード形式のバージョン１ （０ｘ０１）を記述する
予約	このデータは今後の使用のために予約される。 その値は０に設定されなければならない
フラグ	このカードに対する４バイトのフラグ。（図１３参照。） 全ての未使用ビットは０でなければならない
区別用ＩＤ	８バイトの区別用識別子。 区別用識別子はサービス識別子及び サービス固有識別子の２つのフィールドを含む。 サービス識別子は５バイトであり、 カードと関連付けられたサービスを識別する。 サービス固有識別子は３バイトのサービス 固有値である
オブジェクト数	１バイト。このヘッダに続くオブジェクトの数。 ０の可能性もある
チェックサム	カードチェックサムであり、２バイト。 カードチェックサムは１６ビットであり、 チェックサムを除くカード上の全てのデータ バイトの符号無し整数の合計である

Fig. 12

【図１３】

名前	記述（プリカードフラグ値）	値（１６進）
ビープ停止	リーダユニットがデフォルトで音声 フィードバックを供給するのを停止 する。このビットが設定されている 場合、ＵＩ要素オブジェクトにおいて ＵＩ要素の「ビープ反転」フラグが 設定されていない限り、ＵＩ要素が 押下されたときでもリーダは音声 フィードバックを発行しない	0x0000 0001
ＭＯＶＥ イベントなし	ユーザが指をリーダの表面で 動かしたときにリーダユニットが マウスとして機能するのを停止する	0x0000 0002
イベント 座標なし	リーダがＰＲＥＳＳイベント、 ＲＥＬＥＡＳＥイベント、 及びＭＯＶＥイベントに対する座標を 送信するのを停止する。Ｘ値及び Ｙ値は値０を伴って送信される	0x0000 0004

Fig. 13

【図１４】

名前	記述（オブジェクト構造）	長さ
種別	オブジェクトの種別（図１６参照）	１バイト
オブジェクト フラグ	このオブジェクトと関連付けられる 一般的なオブジェクトフラグ （図１５参照）。注：オブジェクト種別に 固有の追加フラグはオブジェクトの データフィールド内で指定される	１バイト
長さ	このオブジェクトに続くデータの長さ。 この値は０の可能性がある	２バイト
データ	このオブジェクトと関連付けられる データ。このデータの構造は オブジェクトの種別によって決まる	可変長

Fig. 14

【図１５】

名前	記述（プリオブジェクトフラグ値）	値（１６進）
非アクティブ	リーダに対してオブジェクトが有効で あるが、その種別に関わらず 無視されることを示す	0x01

Fig. 15

【図１６】

名前	記述（オブジェクト種別）	値（１６進）
ＵＩオブジェクト	ＵＩカードボタン	0x10
カードデータ	このカードのみに関連するデータを含む	0x20
固定長データ	カード上に固定長のデータブロックを 格納するのに使用することができる オブジェクト	0x30
リーダ挿入	カードが挿入されたときにリーダに指示を 与えるのに使用することができる オブジェクト	0x40
操作なし	カード上の空きスペースのブロックを 埋めるのに使用されるオブジェクト	0x01
操作なし （１バイト）	標準オブジェクトヘッダをもたない １バイトのオブジェクト。 通常のオブジェクトヘッダには小さすぎる カード上のスペースを埋めるのに使用される	0x00

Fig. 16

【図１７】

フィールド	記述 （ユーザインタフェースオブジェクト構造）	サイズ
フラグ	カード上のこのＵＩ要素に固有のフラグ	１バイト
X1	このオブジェクトの矩形の左下隅の 座標のＸ値	１バイト
Y1	このオブジェクトの矩形の左下隅の 座標のＹ値	１バイト
X2	このオブジェクトの矩形の 右上隅の座標のＸ値	１バイト
Y2	このオブジェクトの矩形の 右上隅の座標のＹ値	１バイト
データ	このオブジェクトと関連付けられる ０以上のバイトのデータ。 このフィールドのサイズはオブジェクト データサイズ－上述のフィールドの 合計サイズにより判定される	可変長

Fig. 17

【図１８】

名前	記述（ＵＩオブジェクト用のフラグ）	値
ビープ イネーブル反転	このフラグによりボタンはカードヘッダ中の ビープ停止フラグの逆の値を有する。 ヘッダにおいてビープ停止フラグが設定 されていない場合、このフラグにより ボタンはビープ音を発生しない。逆の場合、 ボタンはビープ音を発生する	0x01
オート リピート	ボタンの上で押下状態が保たれるとき、 ボタンと関連付けられたメッセージが 自動的に繰り返される	0x02
押下時データ 送信禁止	このフラグによりボタンはＰＲＥＳＳ イベントの際にボタンと関連付けられた データを送信しない。デフォルトは ＰＲＥＳＳイベントの際にボタンと 関連付けられたデータを送信することである	0x04
リリース時 データ送信禁止	このフラグによりボタンはＲＥＬＥＡＳＥ イベントの際にボタンと関連付けられた データを送信しない。デフォルトは ＲＥＬＥＡＳＥイベントの際にボタンと 関連付けられたデータを送信することである	0x0a

Fig. 18

【図１９】

フィールド	記述（メッセージヘッダ形式）	バイト
プリアンブル	メッセージのプリアンブル。 値は常に０ｘＡＡ ０ｘ５５ （ビットシーケンス１０１０１０１００１０ １０１０１）。これは、イベント マネージャがメッセージの始めを容易に 探し出せるようにするためである	2
バージョン	このメッセージが使用するＵＩカード ＩＲメッセージプロトコルのバージョン。 今回のプロトコルのバージョンは バージョン１である（バージョン フィールドは０ｘ０１）	1
種別	メッセージ種別。これは図２０において 与えられる値のうちの１つである	1
リーダＩＤ	メッセージを送信したリーダの１６ビット ＩＤ。この番号はリーダのバッテリが交換 されるときに変更される疑似ランダムに 生成される数である。 これはアプリケーションと共に複数の リーダが使用されているときにリーダを 区別するのに必要とされる	2
サービス	カード上に格納されたサービス識別子	5
サービス固有	カード上に格納されたサービス固有識別子	3

Fig. 19

【図２０】

名前	記述（メッセージ種別コード）	コード
INSERT	カードがリーダに挿入された	'I'
REMOVE	カードがリーダから抜き取られた	'E'
PRESS	タッチパネルが押下された	'P'
RELEASE	タッチパネルの押下が解除された	'R'
MOVE	押下位置が移動したが押下は 解除されていない	'M'
BADCARD	カードが挿入されたが検証に 合格しなかった	'B'
LOW_BATT	リーダのバッテリの電力が低下している	'L'

Fig. 20

【図２１】

フィールド	記述（単純なメッセージ形式）	バイト
ヘッダ	図１９により定義されるような メッセージヘッダ	14
チェックサム	メッセージチェックサム。これは メッセージ中の全バイトの合計である	1
チェックサム'	チェックサムの１の補数	1

Fig. 21

【図２１（ａ）】

フィールド	記述（ＩＮＳＥＲＴメッセージ形式）	バイト
ヘッダ	図１９により定義されるようなメッセージヘッダ	14
長さ	データのバイト数。０の可能性がある	2
データ	カード上のカードデータオブジェクトからのデータ	長さ
チェックサム	メッセージチェックサム。これはメッセージ中の全バイトの合計である	1
チェックサム'	チェックサムの１の補数	1

Fig. 21(a)

【図２２】

フィールド	記述（ＭＯＶＥメッセージ形式）	バイト
ヘッダ	図１９により定義されるようなメッセージヘッダ	14
X	接触位置のＸ座標	1
Y	接触位置のＹ座標	1
チェックサム	メッセージチェックサム。これはメッセージ中の全バイトの合計である	1
チェックサム'	チェックサムの１の補数	1

Fig. 22

【図２３】

フィールド	記述（ＰＲＥＳＳメッセージ形式／ＲＥＬＥＡＳＥメッセージ形式）	バイト
ヘッダ	図１９により定義されるようなメッセージヘッダ	14
X	接触位置のＸ座標	1
Y	接触位置のＹ座標	1
長さ	データのバイト数。０の可能性がある	2
データ	ユーザインタフェース要素と関連付けられるデータ	長さ
チェックサム	メッセージチェックサム。これはメッセージ中の全バイトの合計である	1
	チェックサムの１の補数	1

Fig. 23

【図２４】

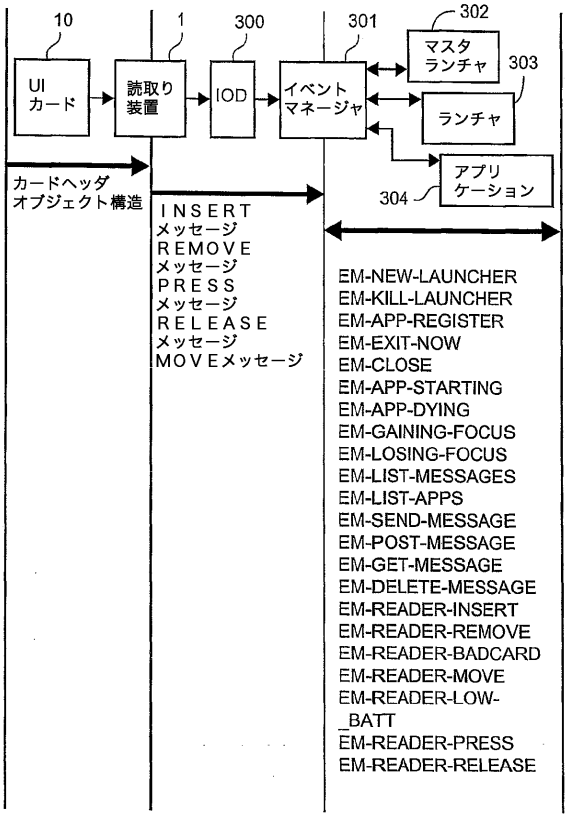


Fig. 24

【図２５】

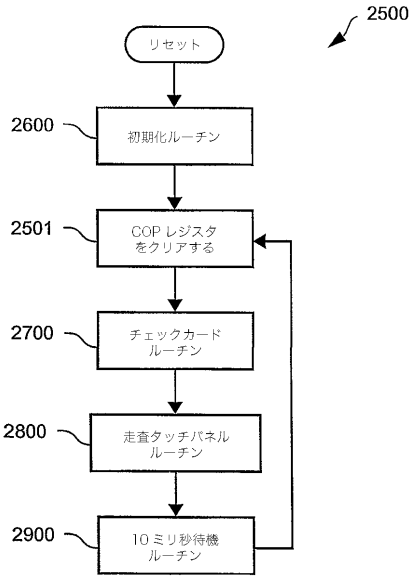


Fig. 25

【図２６】

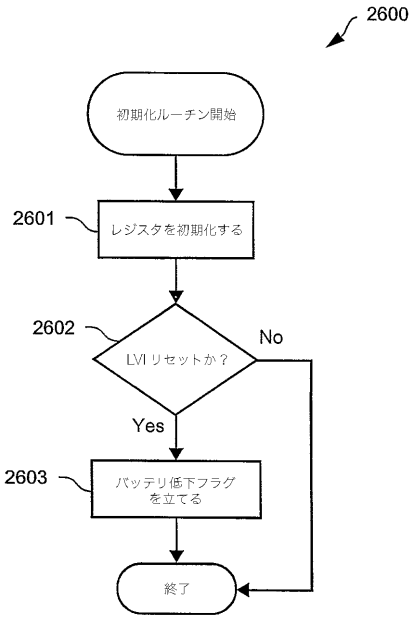


Fig. 26

【図２７】

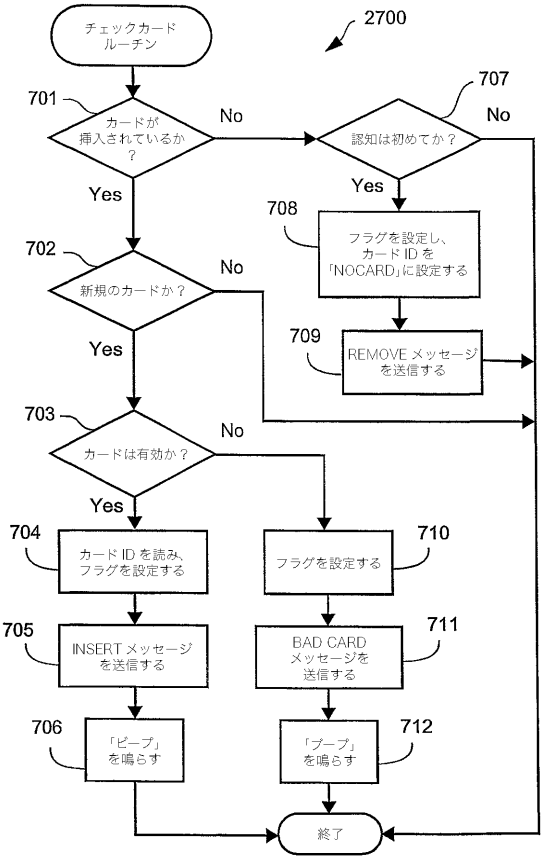


Fig. 27

【図２８】

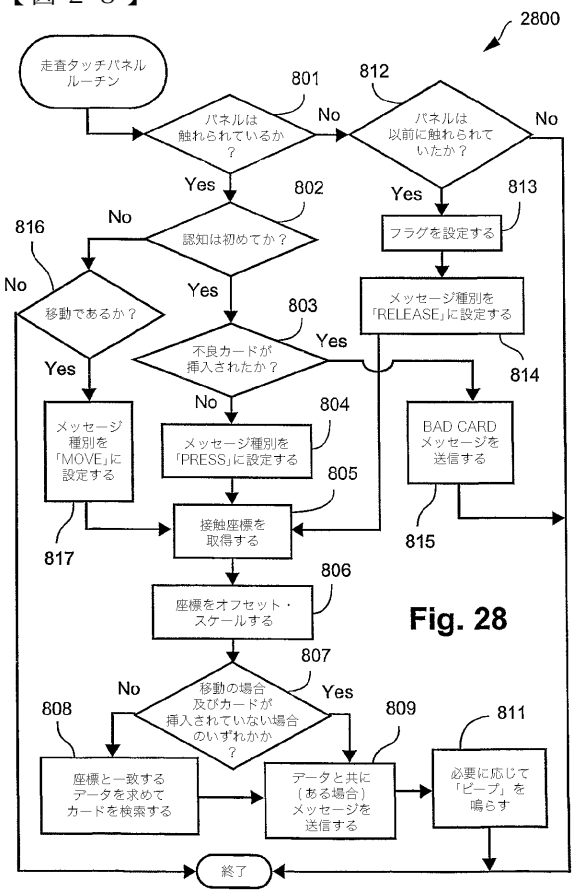


Fig. 28

【図２９】

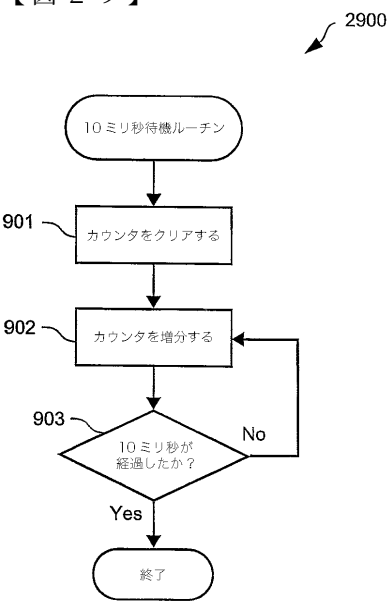


Fig. 29

【図３０】

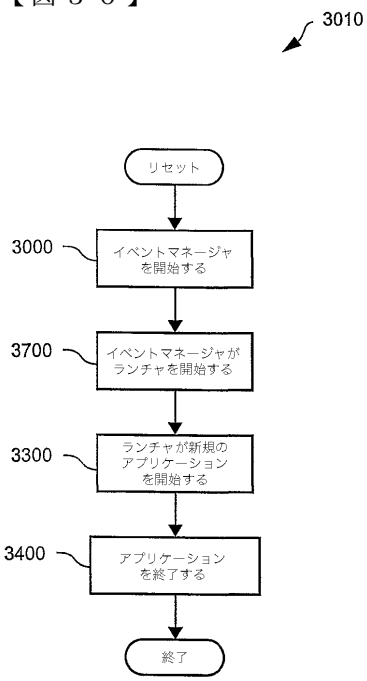


Fig. 30

【図３１】

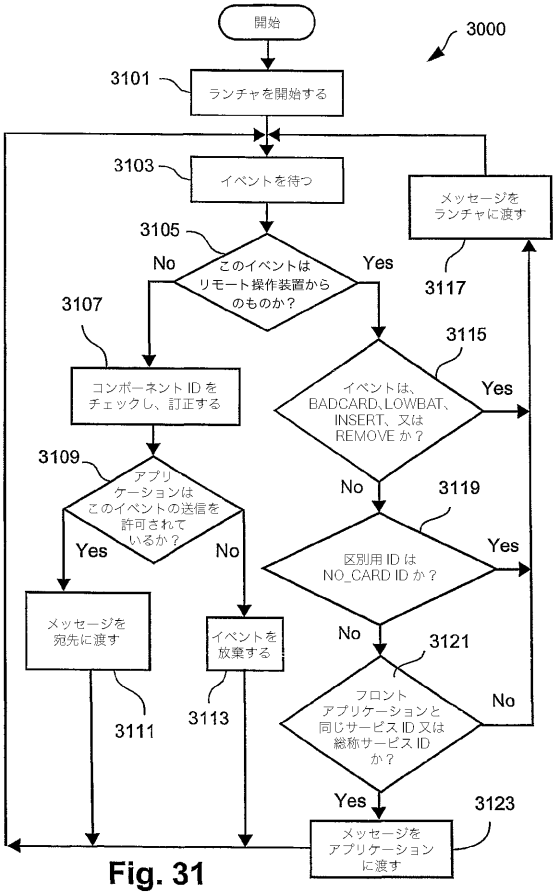


Fig. 31

【図３２】

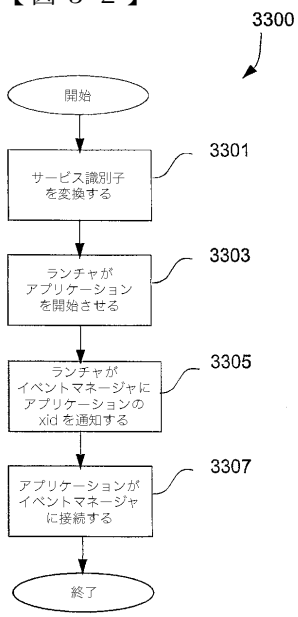


Fig. 32

【図３３】

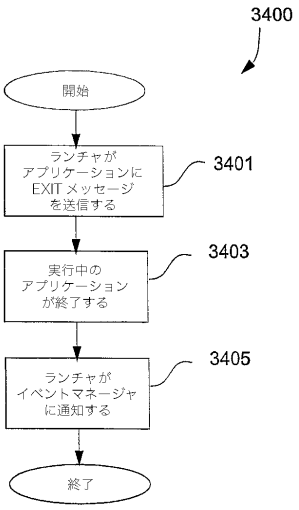


Fig. 33

【図３４】

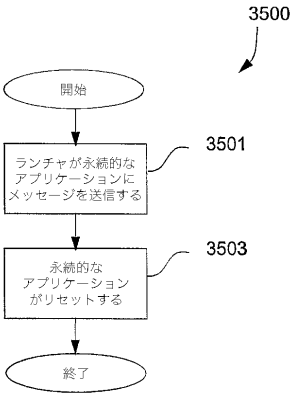


Fig. 34

【図３５】

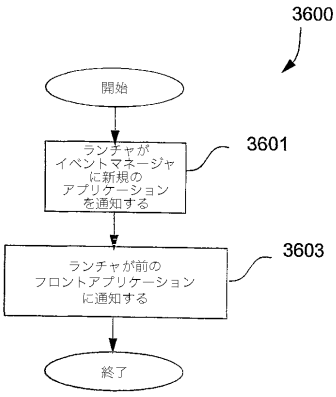


Fig. 35

【図３６】

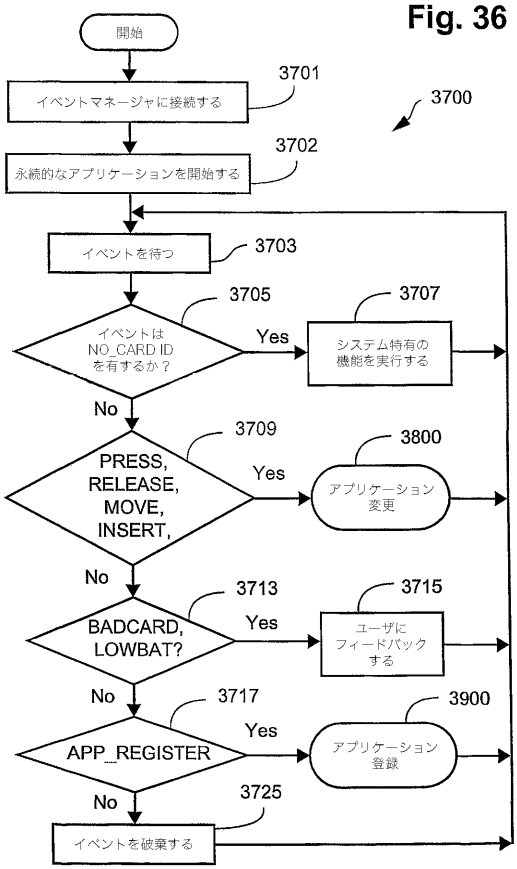


Fig. 36

【図３７】

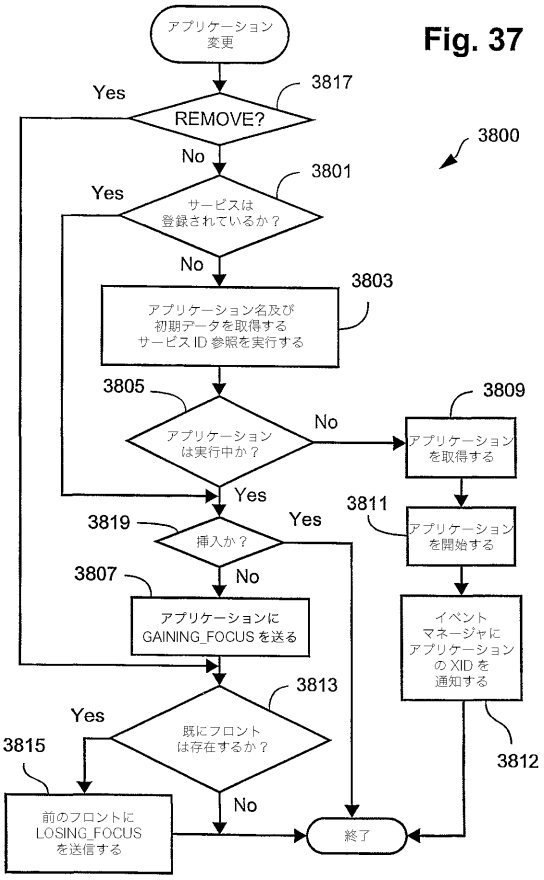


Fig. 37

【図３８】

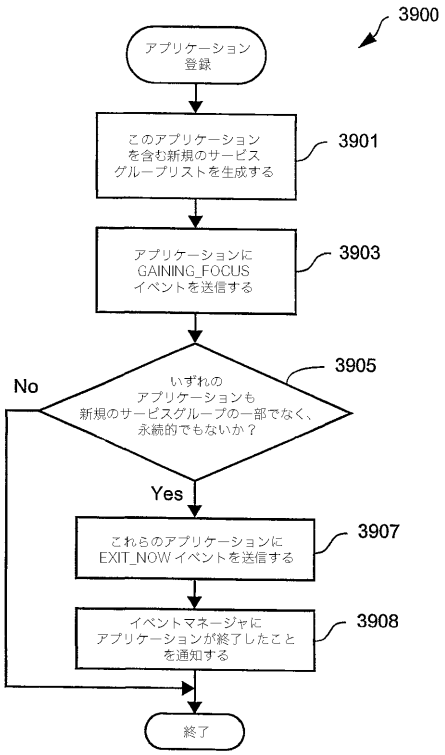


Fig. 38

【図３９】

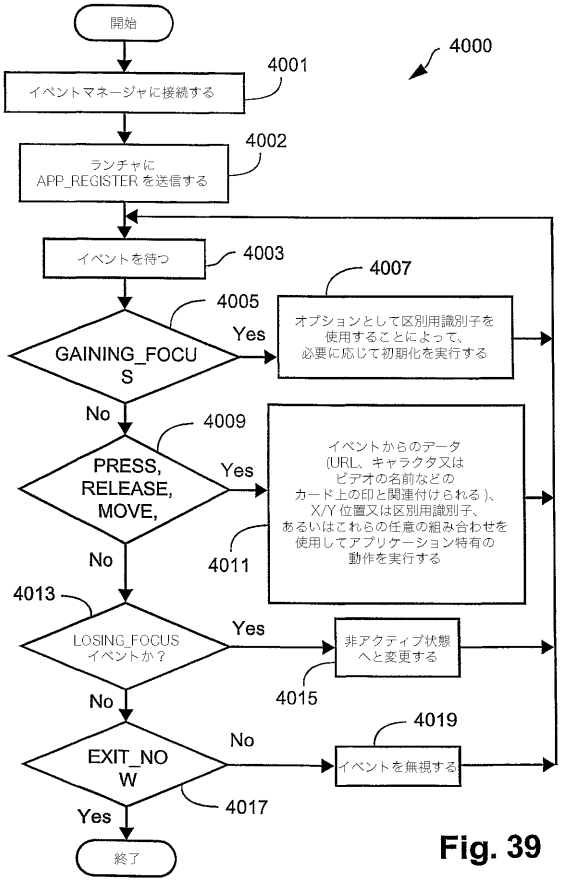


Fig. 39

【図４０】

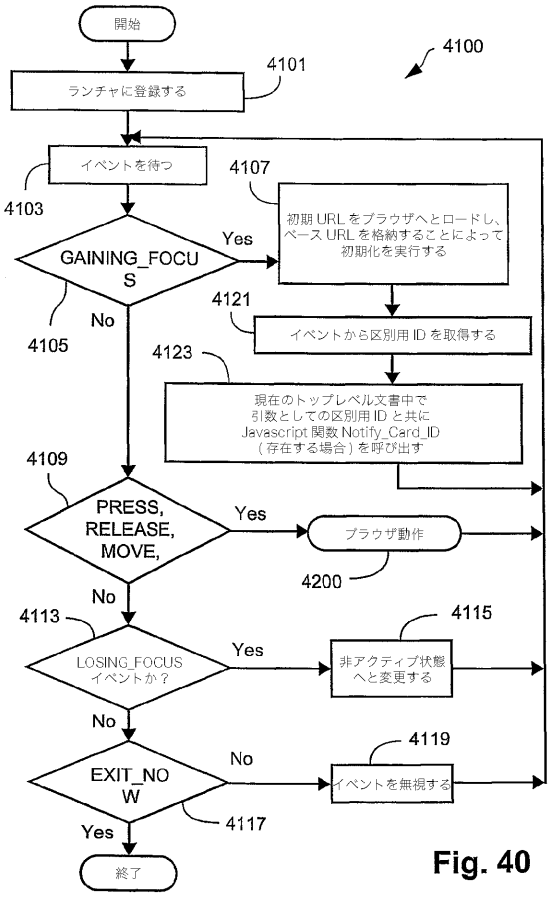
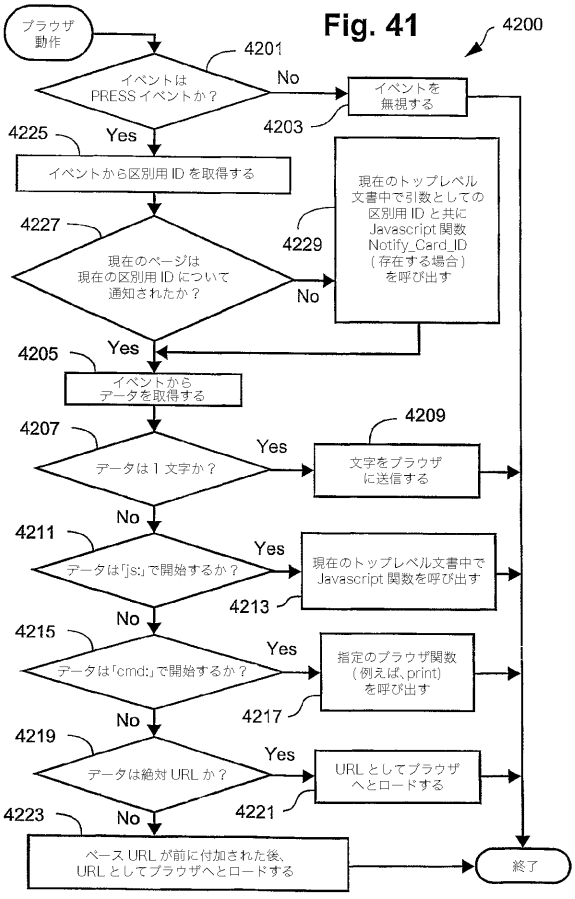


Fig. 40

【図４１】

Fig. 41



【図４２】

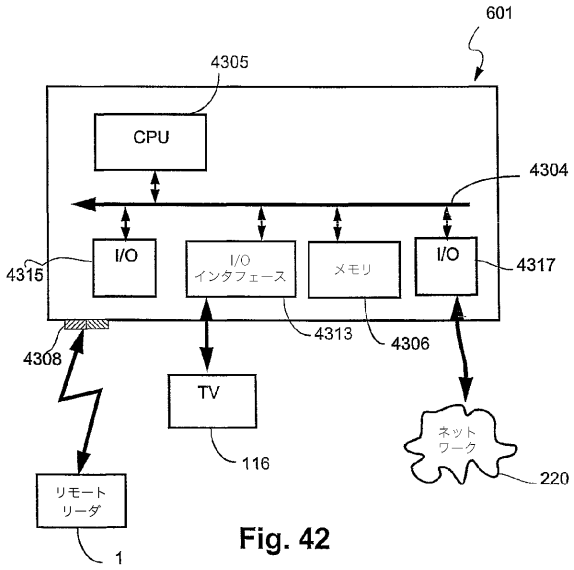


Fig. 42

【図４３】

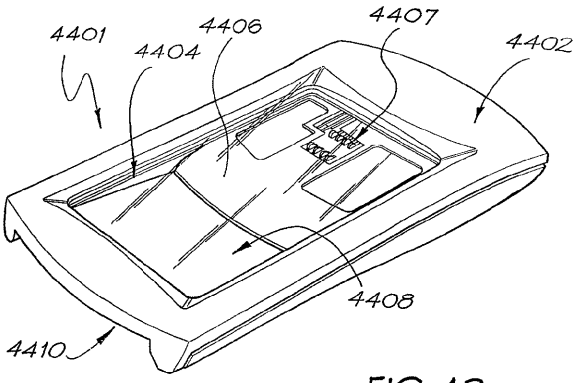


FIG. 43

【図４４】

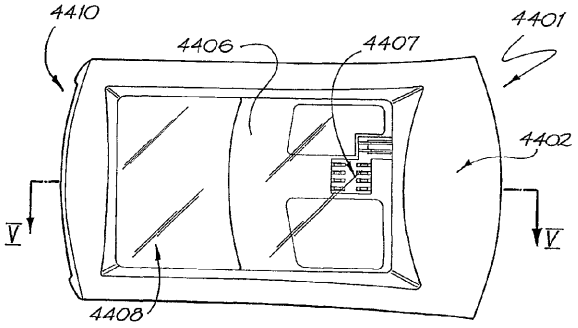


FIG. 44

【図４５】

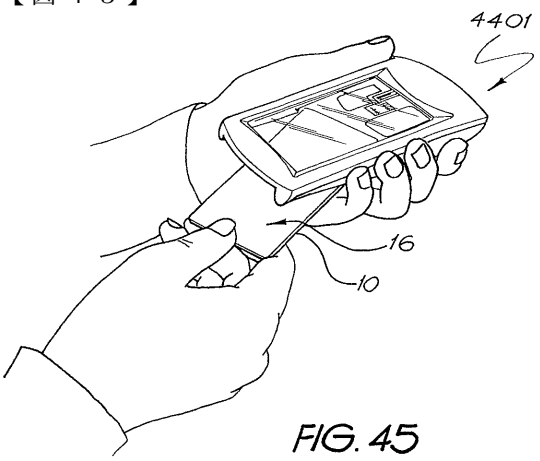


FIG. 45

【図４６】

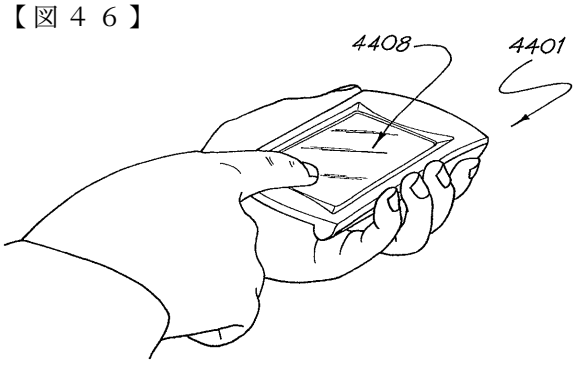


FIG. 46

【図４７（ａ）】

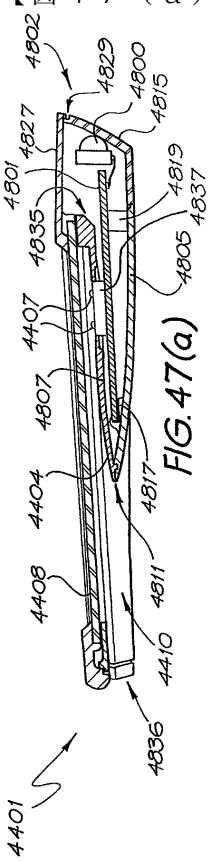


FIG. 47(a)

【図４７（ｂ）】

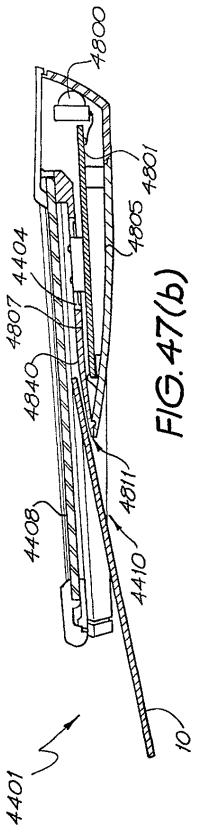


FIG. 47(b)

【図４７（ｃ）】

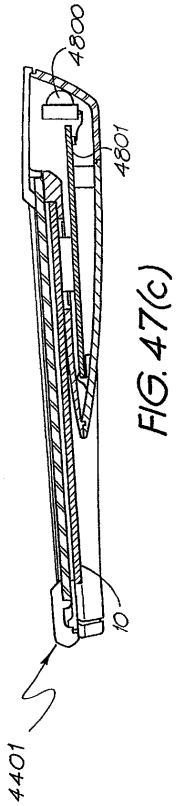


FIG. 47(c)

【図４８】

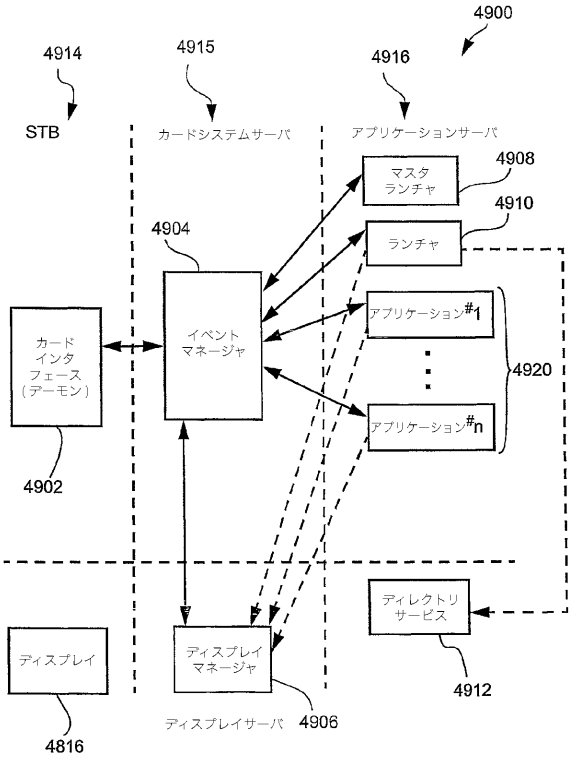


Fig. 48

【図４９】

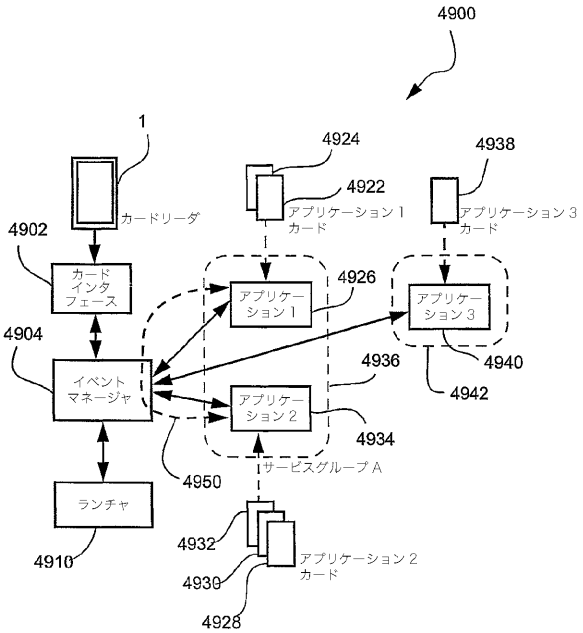


Fig. 49

【図５０】

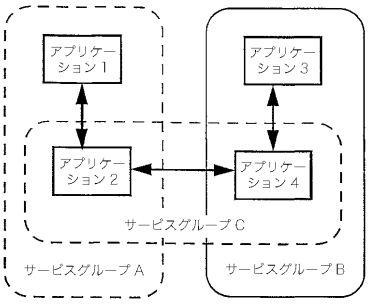


Fig. 50

【図５１Ａ】

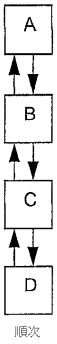
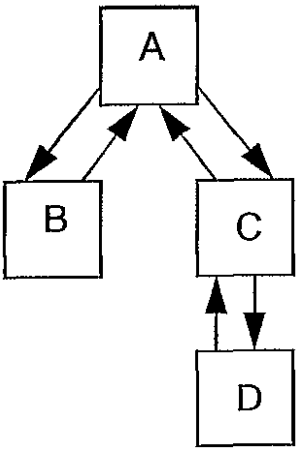


Fig. 51A

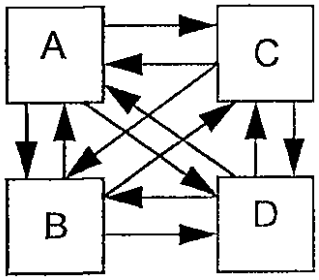
【図５１Ｂ】



階層

Fig. 51B

【図５１Ｃ】



完全網目状

Fig. 51C

【図５２】

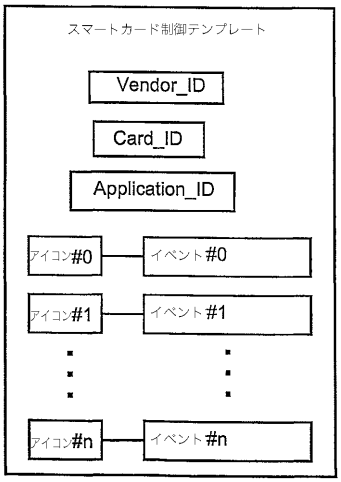


Fig. 52

【図５３Ａ】

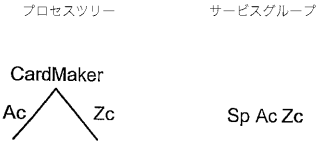


Fig. 53A

【図５３Ｂ】

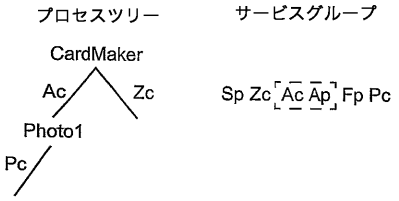


Fig. 53B

【図５３Ｃ】

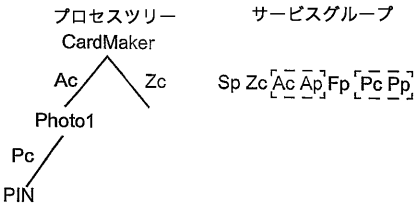


Fig. 53C

【図５３Ｄ】

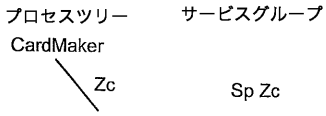


Fig. 53D

【図５３Ｅ】

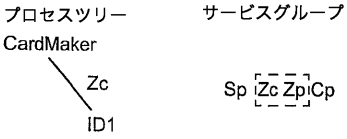


Fig. 53E

【図５４】

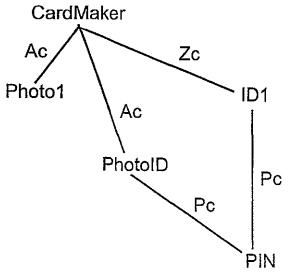


Fig. 54

【図５５】

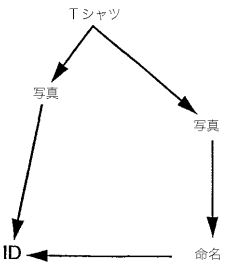


Fig. 55

【図５６（ａ）】

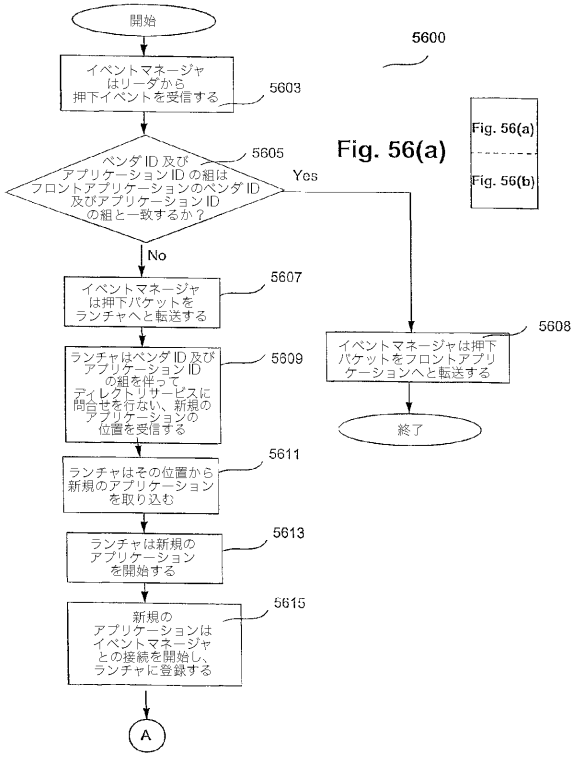


Fig. 56(a)

【図５６（ｂ）】

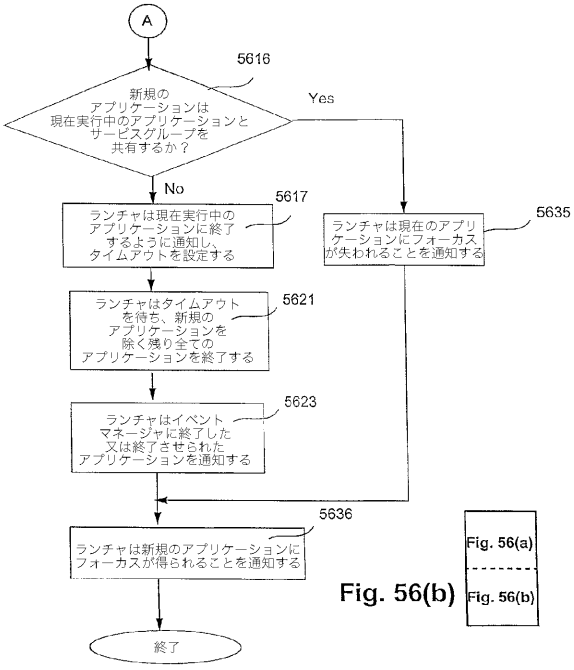


Fig. 56(b)

【図５７】

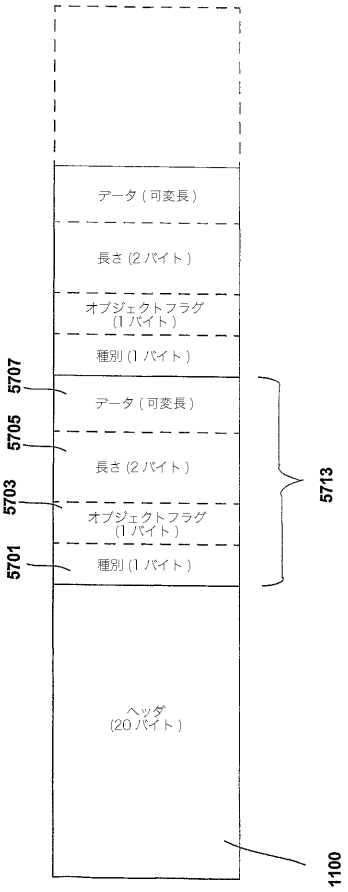


Fig. 57

【図５８】

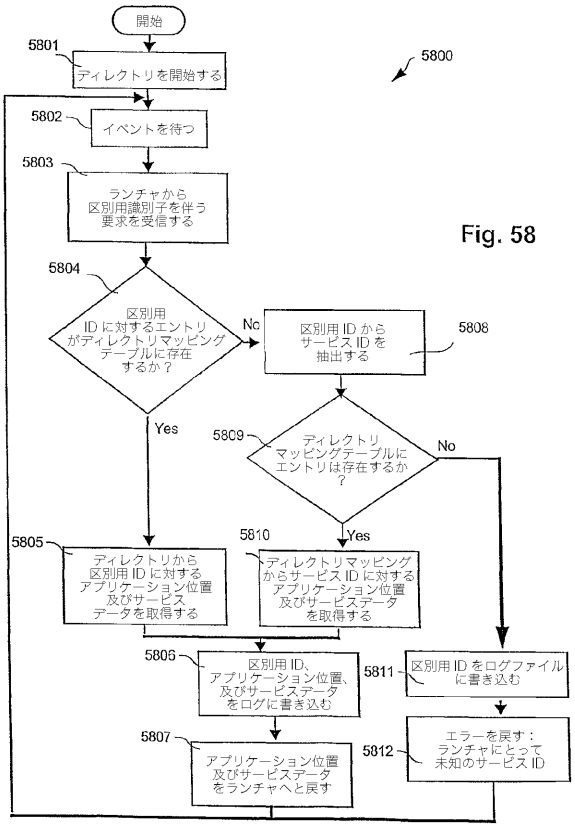


Fig. 58

フロントページの続き

(72)発明者 ヤップ, スーーケン
オーストラリア国 ２１１３ ニュー サウス ウェールズ州, ノース ライド, トーマス ホルト ドライブ １ キヤノン インフォメーション システムズ リサーチ オーストラリア プロプライエタリー リミテツド 内
(72)発明者 スミーリー, ロバート
オーストラリア国 ２１１３ ニュー サウス ウェールズ州, ノース ライド, トーマス ホルト ドライブ １ キヤノン インフォメーション システムズ リサーチ オーストラリア プロプライエタリー リミテツド 内
(72)発明者 フレミング, ハイドン, グラハム
オーストラリア国 ２１１３ ニュー サウス ウェールズ州, ノース ライド, トーマス ホルト ドライブ １ キヤノン インフォメーション システムズ リサーチ オーストラリア プロプライエタリー リミテツド 内
(72)発明者 シンプソンーヤング, ウイリアム
オーストラリア国 ２１１３ ニュー サウス ウェールズ州, ノース ライド, トーマス ホルト ドライブ １ キヤノン インフォメーション システムズ リサーチ オーストラリア プロプライエタリー リミテツド 内
(72)発明者 ニューマン, アンドリュー, ティモシー, ロバート
オーストラリア国 ２１１３ ニュー サウス ウェールズ州, ノース ライド, トーマス ホルト ドライブ １ キヤノン インフォメーション システムズ リサーチ オーストラリア プロプライエタリー リミテツド 内
(72)発明者 ユールゥロー, ツェンヤ, アレクサンダー
オーストラリア国 ２１１３ ニュー サウス ウェールズ州, ノース ライド, トーマス ホルト ドライブ １ キヤノン インフォメーション システムズ リサーチ オーストラリア プロプライエタリー リミテツド 内

審査官 西脇 博志

(56)参考文献 豪州特許出願公開第５３５２７／９９号明細書

(58)調査した分野(Int.Cl., ＤＢ名)
H04Q 9/00-9/10