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

特許第5667083号
(P5667083)

(45)発行日　平成27年2月12日(2015.2.12)　(24)登録日　平成26年12月19日(2014.12.19)

(51)Int.Cl.　F I
G06F 9/44 (2006.01)　G06F 9/06 620A

請求項の数 14 (全 30 頁)

(21)出願番号	特願2011-548292 (P2011-548292)
(86)(22)出願日	平成22年1月28日 (2010.1.28)
(65)公表番号	特表2012-516515 (P2012-516515A)
(43)公表日	平成24年7月19日 (2012.7.19)
(86)国際出願番号	PCT/US2010/022396
(87)国際公開番号	WO2010/088380
(87)国際公開日	平成22年8月5日 (2010.8.5)
審査請求日	平成25年1月22日 (2013.1.22)
(31)優先権主張番号	12/361,591
(32)優先日	平成21年1月29日 (2009.1.29)
(33)優先権主張国	米国(US)

(73)特許権者　501439828
シュナイダー　エレクトリック　アイティー　コーポレーション
アメリカ合衆国　ロードアイランド州　02892　ウエスト　キングストン　フェアグラウンズ　ロード　132
(74)代理人　100147485
弁理士　杉村　憲司
(74)代理人　100173794
弁理士　色部　暁義
(74)代理人　100153017
弁理士　大倉　昭人
(74)代理人　100164471
弁理士　岡野　大和

最終頁に続く

(54)【発明の名称】通信システム及びプロトコル

(57)【特許請求の範囲】

【請求項１】

第１のデバイスのステータス及び設定情報を伝える複数のメッセージを、前記第１のデバイスのユーザに決まった順序で伝える方法であって、前記複数のメッセージ及び前記複数のメッセージの決まった順序は第２のデバイスに記憶され、前記複数のメッセージは最初のメッセージ（メッセージ１）、２番目のメッセージ（メッセージ２）、最後から２番目のメッセージ（メッセージＮ－１）及び最後のメッセージ（メッセージＮ）を前記決まった順序で含み、前記方法は、

前記第１のデバイスに、前記複数のメッセージのうちの、前記最初のメッセージ（メッセージ１）と前記最後のメッセージ（メッセージＮ）を格納する行為と、

前記第１のデバイスのユーザに、前記最初のメッセージ（メッセージ１）、前記２番目のメッセージ（メッセージ２）、前記最後から２番目のメッセージ（メッセージＮ－１）及び前記最後のメッセージ（メッセージＮ）以外の３番目のメッセージを提示する行為と、

前記第１のデバイスのユーザが入力するナビゲーションコマンドを受け取る行為と、

第１の方向を示す前記ナビゲーションコマンドに応答して、前記第１のデバイスのユーザに前記最初のメッセージ（メッセージ１）を提示し、前記第２のデバイスに前記２番目のメッセージ（メッセージ２）を要求するとともに、該２番目のメッセージ（メッセージ２）を前記第１のデバイスに格納する行為と、

前記第１の方向とは反対の第２の方向を示す前記ナビゲーションコマンドに応答して

、前記最後のメッセージ（メッセージＮ）を前記第１のデバイスのユーザに提示し、第２のデバイスに前記最後から２番目のメッセージ（メッセージＮ－１）を要求するとともに、前記最後から２番目のメッセージ（メッセージＮ－１）を前記第１のデバイスに格納する行為とを含む通信方法。

【請求項２】

前記複数のメッセージの各々は、これら複数のメッセージの各々が前記第１のデバイスに格納されるまでは、前記第１のデバイスにではなく、前記第２のデバイスに対して前もって同定される、請求項１に記載の方法。

【請求項３】

前記第２のデバイスに前記最初のメッセージ（メッセージ１）及び前記最後のメッセージ（メッセージＮ）を要求する行為をさらに含む、請求項２に記載の方法。

【請求項４】

前記２番目のメッセージ（メッセージ２）を格納する行為は、前記最後のメッセージ（メッセージＮ）を以前に格納した前記第１のデバイスのメモリ位置に、２番目のメッセージ（メッセージ２）を格納する行為を含む、請求項１または３に記載の方法。

【請求項５】

前記最後から２番目のメッセージ（メッセージＮ－１）を格納する行為は、前記最初のメッセージ（メッセージ１）を以前に格納した前記第１のデバイスのメモリ位置に、前記最後から２番目のメッセージ（メッセージＮ－１）を格納する行為を含む、請求項４に記載の方法。

【請求項６】

いずれの時点においても、前記第１のデバイスのメモリに、前記複数のメッセージのうち、わずか３つのメッセージを格納する行為をさらに含む、請求項１に記載の方法。

【請求項７】

前記第１のデバイスに、前記複数のメッセージの各々を格納するために所定量を越えないメモリを割り当てる行為であって、前記所定量のメモリは、前記複数のメッセージのうち、わずか３つのメッセージを格納するのに十分である、メモリ割り当て行為をさらに含む請求項１に記載の方法。

【請求項８】

複数のメッセージを格納するメモリを備える第１のデバイスであって、前記複数のメッセージは、決められた順序を有し、且つ最初のメッセージ（メッセージ１）、２番目のメッセージ（メッセージ２）、最後から２番目のメッセージ（メッセージＮ－１）、及び最後のメッセージ（メッセージＮ）を前記決められた順序で含んでいる、第１のデバイスと、

前記複数のメッセージを、第２のデバイスのユーザに、前記決められた順序で提示するように構成された第２のデバイスであって、前記複数のメッセージは前記第２のデバイスのステータス及び設定情報を前記ユーザに伝え、該第２のデバイスは、バスによって相互接続されたプロセッサ、メモリ及び提示デバイスを含み、該第２のデバイスのプロセッサは、一連の命令を実行するようにプログラムされている第２のデバイスで、

前記命令によって前記プロセッサは、

前記第２のデバイスのメモリに、前記複数のメッセージのうち、前記最初のメッセージ（メッセージ１）と前記最後のメッセージ（メッセージＮ）を格納させ、

前記最初のメッセージ（メッセージ１）、前記２番目のメッセージ（メッセージ２）、前記最後から２番目のメッセージ（メッセージＮ－１）及び前記最後のメッセージ（メッセージＮ）以外の３番目のメッセージを、前記提示デバイスで前記第２のデバイスのユーザに提示し、且つ

前記第２のデバイスのユーザが入力するナビゲーションコマンドに応答して、

第１の方向を示す前記ナビゲーションコマンドに応答して、前記第２のデバイスのユーザに、前記最初のメッセージ（メッセージ１）を前記提示デバイスで提示し、前記第１のデバイスに、前記２番目のメッセージ（メッセージ２）を要求し、前記第２のデバ

イスのメモリに、前記２番目のメッセージ（メッセージ２）を格納し、
　　前記第１の方向とは反対の第２の方向を示す前記ナビゲーションコマンドに応答して、前記第２のデバイスのユーザに、前記最後のメッセージ（メッセージＮ）を前記提示デバイスで提示し、前記第１のデバイスに、前記最後から２番目のメッセージ（メッセージＮ－１）を要求し、前記第２のデバイスのメモリに、前記最後から２番目のメッセージ（メッセージＮ－１）を格納する、第２のデバイスとを備えている通信システム。
【請求項９】
　前記複数のメッセージのいずれも、前記第２のデバイスのメモリに格納されるまでは、前記第２のデバイスに対して同定されない、請求項８に記載の通信システム。
【請求項１０】
　前記一連の命令は前記第２のデバイスのプロセッサをして、前記第１のデバイスに、前記最初のメッセージ（メッセージ１）と、前記最後のメッセージ（メッセージＮ）を要求する命令をさらに含んでいる、請求項９に記載の通信システム。
【請求項１１】
　前記一連の命令は、前記第２のデバイスのプロセッサをして、以前に前記最後のメッセージ（メッセージＮ）を格納した前記第２のデバイスのメモリの第１の部分に、前記２番目のメッセージ（メッセージ２）を格納させる命令をさらに含んでいる、請求項８または１０に記載の通信システム。
【請求項１２】
　前記一連の命令は前記第２のデバイスのプロセッサをして、以前に前記最初のメッセージ（メッセージ１）を格納した前記第２のデバイスのメモリの第２の部分に、前記最後から２番目のメッセージ（メッセージＮ－１）を格納させる命令をさらに含んでいる、請求項１１に記載の通信システム。
【請求項１３】
　前記第１のデバイスは、前記第２のデバイスとは物理的に別個のものである、請求項８に記載の通信システム。
【請求項１４】
　前記第２のデバイスのメモリは、いずれの時点にも、前記複数のメッセージのわずか３つのメッセージのみを格納し、
　前記一連の命令は、前記第２デバイスの前記プロセッサをして、前記第２のデバイスに、前記複数のメッセージの各々を格納するために、前記第２のデバイスに所定量を超えないメモリを割り当てる命令をさらに含み、前記第２のデバイスのメモリの所定量は、前記複数のメッセージのうちの、わずか３つのメッセージを格納するのに十分である、請求項８に記載の通信システム。

【発明の詳細な説明】
【技術分野】
【０００１】
　本発明は、概して通信システム及びプロトコルに関し、より詳細には、デバイスがメッセージを当該デバイスのユーザに伝えることができ、メッセージの内容、順序、及び数をランタイム前に当該デバイスが知らなくて済む、通信システム及びプロトコルに関する。
【背景技術】
【０００２】
　多くの産業において、異なる分野の専門知識に属する企業同士が互いに共同し、各企業の能力を組み合わせて新製品を開発することがある。例えば、データセンタ用の無停電電源装置の設計で知られる企業が、メディアシステムサーバ及び制御デバイスの設計で知られる他の企業と協力し、他の企業のメディアシステムサーバ及び制御デバイスと統合した

オーディオ及び／またはビデオシステム用の、電源の管理・保護のソリューションを提供することができる。このような共同製品の設計段階において、該製品の設計、特徴、及び性能等を反映した仕様をドラフトし、修正し、そして仕上げる。一度設計が仕上がると、共同製品のスケジュールに影響を与えずに、新規の特徴または機能性を製品に追加したり、または設計上の見落としを修正したりすることが困難になる。

【０００３】

通常、設計過程において早い段階で特定される（また、非常に頻繁に変更されたり、いずれ進展させたりする）共同製品の一部位は、ユーザインターフェースである。例えば、設計過程の早期段階において、共同製品が、ボタン、キー、ダイヤル、提示要素（例えば提示される行数、ステータスＬＥＤ、ブザー等）、その他ユーザが利用可能な機能要素をいくつ有するか、及びそれらの機能性はどうするかについて、共同している企業は合意する必要がある。したがって、例えば共同製品が１行だけ提示するディスプレイや１つだけのボタンを有する場合、何らかのメッセージが存在する場合、例えば、製品に電源が投入された場合に、また、ユーザがボタンを１回押した場合に、さらにボタンが再度押された場合に、ユーザに対してディスプレイ上にどんなメッセージを提示するかについて、共同している企業は合意する必要がある。

【発明の概要】

【発明が解決しようとする課題】

【０００４】

通常の製品であれば、これらのメッセージ（すなわち、メッセージの内容、順番、及び典型的にはその数）は、製品のファームウェアに格納または反映され、該ファームウェアは通常の製品設計過程において、生産に十分余裕を持って凍結される。これは不都合なことである。なぜならば、このコードの凍結は、エラーが発見されるかもしれない、または新規或いは異なる機能性を提供すべき決定がなされるかもしれない最終確認試験のずっと前に生じることがよくあるためである。

【課題を解決するための手段】

【０００５】

本発明の一態様によれば、デバイスが実質上無制限の数のメッセージを当該デバイスのユーザに伝える必要があることができ、ランタイム前に、メッセージの数、順序及び内容をデバイスが知る必要はない、柔軟で、効率的で、メモリに依存するメッセージングシステム及びメッセージ提示プロトコルを提供する必要があることがわかる。好適には、メッセージングシステム及びメッセージングプロトコルは、メッセージをユーザストアに使えるか、またはＲＯＭベースのメモリ（ＲＯＭ、ＰＲＯＭ、ＥＥＰＲＯＭ、フラッシュメモリ等）のような、デバイス固有の静的メモリからのメッセージにアクセスするデバイスを必要としないか、または大量の動的メモリ（例えばＲＡＭ）及び大量のプロセス能力を必要としないのが有利である。ランタイム前にメッセージの数、順序及び内容を知っておく（すなわちデバイス内に格納する）必要がないため、メッセージの追加、並び替え、削除または変更を行うことができる。さらに、デバイスの電源を投入する度に、新たな機能性を提供することもできる。

【０００６】

本発明の一実施形態によれば、決まった順序に配列された複数のメッセージを、第１のデバイスのユーザに伝える方法が提供される。前記複数のメッセージには、決まった順序に並べられた最初のメッセージ、２番目のメッセージ、最後から２番目のメッセージ及び最後のメッセージが含まれる。本発明による方法は、前記第１のデバイスに、前記複数のメッセージのうち、最初のメッセージと最後のメッセージを格納する行為と、前記第１のデバイスのユーザに３番目のメッセージを伝える行為とを含む。前記第１のデバイスのユーザから受け取ったナビゲーションコマンドの方向に応答して、本発明の方法はさらに、前記ナビゲーションコマンドが第１の方向を示す場合には、前記第１のデバイスのユーザに前記最初のメッセージを伝え、第２のデバイスに前記２番目のメッセージを要求するとともに、前記２番目のメッセージを前記第１のデバイスに格納する行為と、前記ナビゲー

ションコマンドが前記第１の方向とは反対の第２の方向を示す場合には、前記最後のメッセージを前記第１のデバイスのユーザに伝え、第２のデバイスに前記最後から２番目のメッセージを要求するとともに、前記最後から２番目のメッセージを前記第１のデバイスに格納する行為とを含む。

【０００７】

本発明の他の実施形態によれば、決められた順序を有する複数のメッセージが格納される第１のデバイスと、前記複数のメッセージを、前記決められた順序でユーザに提示する第２のデバイスとを備える通信システムが提供される。前記複数のメッセージは、最初のメッセージ、２番目のメッセージ、最後から２番目のメッセージ及び、最後のメッセージを決められた順序で含んでいる。前記第２のデバイスは、バスによって相互接続されたプロセッサ、メモリ及びディスプレイを含む。前記プロセッサは、一連の命令を実行するようにプログラムされており、前記命令によって前記プロセッサは、前記第２のデバイスのメモリに、前記複数のメッセージのうち、前記最初のメッセージと前記最後のメッセージを格納させ、３番目のメッセージを、前記第２のデバイスのユーザに提示させ、且つ前記第２のデバイスのユーザから受け取ったナビゲーションコマンドの方向に応答して、前記ナビゲーションコマンドが、第１の方向を示す場合には、前記第２のデバイスのユーザに、前記最初のメッセージを提示し、前記第１のデバイスから前記２番目のメッセージを要求し、前記第２のデバイスのメモリに、前記２番目のメッセージを格納し、一方、前記ナビゲーションコマンドが、前記第１の方向とは反対の第２の方向を示す場合には、前記第２のデバイスのユーザに、前記最後のメッセージを提示し、前記第１のデバイスに、前記最後から２番目のメッセージを要求し、前記第２のデバイスのメモリに、前記最後から２番目のメッセージを格納する。

【０００８】

本発明の他の実施形態によれば、第１のデバイスのユーザに複数のメッセージを伝える方法が提供される。前記方法には、前記第１のデバイスに、前記複数のメッセージのうち最初のメッセージと２番目のメッセージを格納する行為と、前記第１のデバイスのユーザに、３番目のメッセージを伝える行為と、前記第１のデバイスのユーザから受け取ったナビゲーションコマンドの方向に応答して、前記ナビゲーションコマンドが第１の方向を示す場合に、前記第１のデバイスのユーザに、前記最初のメッセージを伝え、第２のデバイスに４番目のメッセージを要求し、前記第１のデバイスに前記４番目のメッセージを格納する行為と、前記ナビゲーションコマンドが第２の方向を示す場合には、前記２番目のメッセージを、前記第１のデバイスのユーザに伝え、前記第２のデバイスに５番目のメッセージを要求し、前記第１のデバイスに前記第５のメッセージを格納する行為が含まれる。

【図面の簡単な説明】

【０００９】

【図１】第１のシステムが知っているメッセージを第２のシステムに提示し得る、本発明による通信システムを示す図である。

【図２】本発明により実施し得るメッセージングループを機能的に示す図である。

【図３】ユーザにメッセージを伝えるために用いることができる本発明によるプロセスを示す図である。

【図４】本発明の様々な態様を実施し得る無停電電源装置の正面図である。

【図５】本発明の様々な態様を実施し得るプロセッサベースシステムの機能ブロック図である。

【発明を実施するための形態】

【００１０】

本発明のさらなる特徴及び利点と、本発明の様々な実施形態の構造及び動作を、添付された図面を参照して以下詳述する。図面において、同様の参照番号は、同様または機能的に類似する構成要素を示す。さらに、参照番号の左から１、２番目の数字は、該参照番号が最初に現れる図面の番号を示す。

【００１１】

添付の図面は、正しい縮尺で記載することを意図していない。図面中において、様々な図面に描かれている同一または略同一の要素には、同様な参照番号が付されている。明瞭化のため、すべての図のすべての要素に参照符号を付してはいない。

【００１２】

図１は、本発明の様々な態様を実施し得る通信システムを示す。特に、通信システム１００は、１つ以上のシステム１１０及びシステム１２０を含む。これらのシステム１１０及びシステム１２０は、メッセージ通信プロトコルに従って、通信媒体１３０を介して互いに通信する。システム１１０及びシステム１２０の各々は、完全に独立した、及び／または、個別のプロセッサベースシステムまたはデバイスとする、あるいは、単一のシステムまたはデバイス内に備えられる、例えば、無停電電源装置や、環境制御システム等のような、異なるサブシステムまたはサブデバイスとすることもできる。通信媒体１３０は、情報を送ることができる、例えばポイント・ツー・ポイント接続や、バスや、ネットワーク等のような、従来既知の、任意の有線または無線通信媒体とすることができる。システム１００を、メディアエンターテインメントシステムとする本発明の一実施形態によれば、システム１１０は、プロセッサベースのメディアシステムコントローラとすることができ、またシステム１２０は、プロセッサベースの無停電電源システムとすることができる。

【００１３】

図１に示すように、システム１２０は、複数の機能要素１２５、１２７を含み、これらの機能要素１２５、１２７を介して、情報をシステム１２０のユーザに伝え、また情報をシステム１２０のユーザから受け取ることができ、ユーザはシステム１２０と対話してすることができる。複数の機能要素は、情報をユーザに伝えることができる少なくとも１つの提示デバイス１２５と、ユーザがシステム１２０と対話して、少なくとも１つの提示デバイス１２５を介してユーザに伝えられる情報を選択できるようにする少なくとも１つの入力デバイス１２７とを備える。本発明の一実施形態によれば、少なくとも１つの提示デバイス１２５は、情報を視覚的にユーザに提示するディスプレイデバイスとすることができる。しかしながら、他の種類の提示デバイスを用いることもできることは明らかである。例えば別の実施形態では、提示デバイス１２５を、オーディオスピーカとすることができ、さらに他の実施形態では、複数の種類の提示デバイスを用いることもでき、例えばビジュアルディスプレイ及びオーディオスピーカを設けることもできる。本発明の実施形態によれば、少なくとも１つの入力デバイス１２７は、少なくとも１つの押しボタンまたはキーを備え、これによってユーザは、彼らに伝えられる情報を選択することができる。しかしながら、他の種類の入力デバイス（例えば、トラックボール、マウス、又はキーボード等）を代わりに用いることもできる。

【００１４】

図１には示されていないが、システム１１０も同様に、システム１２０につき上述したものに類似する機能要素を含み、これらの機能要素により、情報をシステム１１０のユーザに伝え、またシステム１１０のユーザから情報を受信することができる。システム１００は本発明の様々な実施形態による通信プロトコルの１つ以上の態様を実装することができるシステムの例示的な実施形態に過ぎず、システムの他の実装（例えばシステム１００よりも多くの、又はより少ないシステムを有する変形例）も可能であり、それらは本発明の範疇にあることは明らかである。

【００１５】

本発明によれば、システム１１０を「メッセージ提供システム」と称し、これは、システム１１０は、システム１２０のユーザに伝えるべき複数のメッセージを格納可能なメモリを含む。ここで、「メッセージ提示システム」と称するシステム１２０は、ユーザがとった所定の行為に応答して、提示デバイス１２５を介してユーザに複数のメッセージを伝えるか、または提示する。本発明の態様によれば、メッセージ提示システム１２０のユーザに伝えられる情報は、ランタイム前にメッセージ提示システム１２０に知らせる、即ち、メッセージ提示システム１２０に格納する必要がない。その代わりに、メッセージ提示

システム１２０のユーザに伝えられる情報は、メッセージ提供システム１１０によってメッセージ提示システム１２０に伝達する。このようにすれば、メッセージ提示システム１２０は、例えばＣＰＵ電源、ユーセージ（usage）及びメモリ等のような、メッセージ提示システムのリソースをほとんど必要としないで、可変の内容を持つ無制限の数のメッセージをユーザに提示することができるので有利である。実際、本発明の態様によれば、メッセージ提示システム１２０は、２つまたは３つのメッセージを格納するのに十分な量のメモリと、他のメッセージを要求する能力とを必要とするだけである。本発明の様々な実施形態に関わるこれらの態様及び利点を、図２～５を参照して以下詳細に説明する。

【００１６】

図２は、本発明の実施形態に従って実行しうるメッセージループを機能的に示す図である。メッセージループ２００は、図１のメッセージ提供システム１１０のようなメッセージ提供システムにより提供されるメッセージに基づいて、図１のメッセージ提示システム１２０のようなメッセージ提示システムにより実行される。メッセージの数及びその内容はランタイムに先立ってメッセージ提示システムは知らなくて良く、また、メッセージは、聴覚的、視覚的、またはその他の方法により提示することができる。

【００１７】

図２に示すように、メッセージループ２００は、複数のフレキシブルメッセージ２１０を含む。メッセージ提示システムはランタイム前に、複数のフレキシブルメッセージ２１０の内容及び数を知る必要がない（従って、これらのメッセージを、ここでは「フレキシブル」メッセージと称している）。本発明の一実施形態によれば、これらのフレキシブルメッセージ２１０は、メッセージ提示システム中の、ダイナミックＲＡＭのような揮発性メモリに格納され、常に、限られた数のメッセージだけが格納される。例えば、一実施形態では、常に、２つのメッセージだけを格納し、また、他の実施形態では、常に、３つのメッセージだけを格納する。フレキシブルメッセージ２１０のサイズ（すなわちメッセージ長）は、メッセージ提示システムの能力次第で変えることができる。本明細書にて用いる「メッセージ」とは、ユーザに提示できる任意形式の情報であり、英数字のデータ及び画像コンテンツ等を含む。一実施形態によれば、メッセージは視覚的に提示可能な文字ベースの英数字情報を含むが、他の種類の視覚的に提示可能な情報を用いることもできることは明らかである。メッセージは完全な単語や考えやリフを伝えるために、多数のメッセージを必要とするような、完全な構成とする必要はない。

【００１８】

図２に示されるように、メッセージループ２００は、１つまたは複数の静的メッセージ２２０を含むことができ、これらのメッセージ２２０は、ランタイム前にメッセージ提示システムが知っていても、知らなくてもよく、静的メッセージ２００は例えばメッセージ提示システムの不揮発性メモリ（ＲＯＭ等）に格納することができる。例えば、メッセージ提示システムが、規格品をカスタマイズしたバージョンに相当する場合、これらの静的メッセージ２２０は規格品の中心的な機能性に関連するメッセージを反映することができるのに対し、フレキシブルメッセージ２１０は、規格品をカスタマイズしたバージョンに固有な追加の能力を反映するメッセージを反映することができる。一実施形態によれば、ユーザは、例えばキー又はボタンを押すことによって、ループの静的メッセージ及びフレキシブルメッセージをインクリメントまたは進めることができる。例えば、電源投入後の初期設定時に、メッセージ提示システムは第１の静的メッセージである、静的メッセージ１を提示することができる。ユーザがダウンキーまたはダウンボタンを押すことに応えて、静的メッセージ２が提示される。ユーザが同じ方向（例えば下方）に押し続ける場合には、ある時点にて最後の静的メッセージである静的メッセージｍが提示されるようになり、続いて最初のフレキシブルメッセージである、フレキシブルメッセージ１が提示され、次にフレキシブルメッセージ２等が提示されるようになる。ユーザが同じ方向に押し続けている場合には、最後のフレキシブルメッセージである、フレキシブルメッセージＮが提示され、続いて静的メッセージ１が提示され、次に静的メッセージ２等が提示され、このようにして、メッセージループが実行される。

【００１９】
代わりに、本発明の実施形態によれば、メッセージループ２００を反対方向にトラバース、即ち進めることもできる。例えば、初期設定して、静的メッセージ１を提示した後にユーザが繰り返しアップキーまたはアップボタンを押した場合には、フレキシブルメッセージＮが提示され、続いてフレキシブルメッセージＮ－１等が提示され、メッセージループ２００を他の方向（例えば上方）にトラバースするようになる。ユーザがメッセージループ２００を同じ方向にトラバースさせ続けるものとすれば、或る時点に、フレキシブルメッセージ１が提示され、続いて静的メッセージｍ、ｍ－１等が提示されることになる。特定の実施形態では、単一の静的メッセージしかなく、最初の静的メッセージと最後の静的メッセージが事実上同一のメッセージとなるようにすることができ、他の実施形態では静的メッセージを全く存在させずに、全てのメッセージをフレキシブルメッセージとすることができる。

【００２０】
本発明の態様によれば、メッセージ提示システムがユーザにメッセージのループを提示することができる通信プロトコルが提供される。メッセージのループは反対方向にトラバースさせることができ、ループ内のメッセージの数、順序及び内容を（すなわち、メッセージループはフレキシブルである）は、ランタイム前にメッセージ提示システムに知らせる必要はない。これらのメッセージをメッセージ提示システムに提供するメッセージ提供システムは、メッセージ提示システムとやりとりする前に、メッセージの数、内容及び順序を知っているが、他にメッセージ提供システムに要求されることはほとんど無い。ある実施形態によれば、通信プロトコルはコマンドのセットを含み、これらのコマンドによりメッセージ提示システムは、メッセージ提供システムに対し、ループの最初及び最後のメッセージを要求することができ、また、ループをいずれかの方向にトラバースする場合に、メッセージ提供システムにループ内の次のメッセージを要求することができる。このような通信プロトコルを用いることで、メッセージ提示システムは現行のメッセージを提示し、次のメッセージを要求し、且つ前のメッセージを保有して、ループをトラバースする方向に無関係に、次にユーザに提示するメッセージを、メッセージ提示システム内にて局所的に見つけ（例えば格納し）、ユーザにほぼ即時に提示することができる。

【００２１】
メッセージ提示システムが、１つ以上の静的メッセージも提示できる一実施形態では、通信プロトコルは、フレキシブルメッセージループのターミネータも含み、このターミネータは、フレキシブルメッセージループが終了する方向に関係なく、メッセージ提示システムに、フレキシブルメッセージループの終了を警告する。メッセージ提示システムは、フレキシブルメッセージループのターミネータを受け取ると、再び静的メッセージを提示したり、その他の動作を行ったりする。このようなメッセージ通信プロトコルを以下、下記の表１および表２につき説明する。

【００２２】
表１は、メッセージ提示システムがメッセージ提供システムに送信するフレキシブルメッセージループに対するコマンドの典型例の一式と、メッセージ提供システムが該コマンドを受け取った際にとる動作を示す。

【００２３】
模範的なフレキシブルメッセージループのコマンド

【表１】

コマンド名	10進値	説明	利用法
DEC_LEFT（シンボル‘<<’）	174	メッセージを１つデクリメントする	メッセージ提供システムは、このコマンドをメッセージ提示システムから受け取ると、即座にメッセージ(カレントメッセージインデックス－１）を返答する。
INC_RIGHT（シンボル‘>>’）	175	メッセージを１つインクリメントする	メッセージ提供システムは、このコマンドをメッセージ提示システムから受け取ると、即座にメッセージ(カレントメッセージインデックス＋１）を返答する。
MSG_N	176	最後のメッセージを要求する	メッセージ提供システムは、このコマンドをメッセージ提示システムから受け取ると、即座に最後のメッセージ(メッセージN)を返答する。また、カレントメッセージインデックスをNに設定する。
MSG_1	177	最初のメッセージを要求する	メッセージ提供システムは、このコマンドをメッセージ提示システムから受け取ると、即座に最初のメッセージ(メッセージ１)を返答する。また、カレントメッセージインデックスを１に設定する。

【００２４】

この模範的なメッセージの通信プロトコルによると、メッセージ提供システムは、直近にメッセージ提示システムに送信したメッセージを同定するインデックス（カレントメッセージインデックス）を格納する。メッセージ提供システムはまた、最初のフレキシブルメッセージ（すなわちメッセージ１）及び最後のフレキシブルメッセージ（すなわちメッセージＮ）と考えられるフレキシブルメッセージを同定する識別子を格納する。どのメッセージを最初のメッセージ及び最後のメッセージと考えられるかは、メッセージ提示システムと対話するまでは、メッセージ提供システムだけが知って必要があることは明らかである。従って、最初のメッセージ及び最後のメッセージ、さらにメッセージの数、その内容及びその順序を時とともに変えることができる。

【００２５】

表１に示すように、フレキシブルメッセージループのコマンドには、DEC_LEFTコマンド、INC_RIGHTコマンド、MSG_Nコマンド、及びMSG_1コマンドが含まれる。メッセージ提示システムは、ループをある方向（例えば上方または逆方向）にトラバースする場合に、DEC_LEFTコマンドをメッセージ提供システムに送信して、ループの次のメッセージを要求する。メッセージ提供システムは、DEC_LEFTコマンドに応答し、メッセージ（カレントメッセージインデックス－１）を返答し、カレントメッセージインデックスの値を１つデクリメントする。メッセージ提示システムは、ループの反対方向（例えば下方または順方向）にトラバースする場合に、INC_RIGHTコマンドをメッセージ提供システムに送信し、次のメッセージを要求する。メッセージ提供システムは、INC_RIGHTコマンドに応答し、メッセージ（カレントメッセージインデックス＋１）を返答し、カレントメッセージインデックスの値を１つインクリメントする。

【００２６】

メッセージ提示システムは、MSG_Nコマンドをメッセージ提供システムに送信し、ループの最後のメッセージ（例えばメッセージＮ）を要求する。メッセージ提供システムは、

MSG_Nコマンドに応答し、ループの最後のフレキシブルメッセージ（すなわちメッセージＮ）を返答し、カレントメッセージインデックスの値をＮに設定する。メッセージ提示システムは、MSG_1コマンドをメッセージ提供システムに送信し、ループの最初のメッセージ（例えばメッセージ１）を要求する。メッセージ提供システムは、MSG_1コマンドに応答し、ループの最初のフレキシブルメッセージ（すなわちメッセージ１）を返答し、カレントメッセージインデックスの値を１に設定する。

【００２７】

表２に示すように、メッセージ提示システムが、１つ以上の静的メッセージを提示することができる本発明の一実施形態によれば、フレキシブルメッセージループのコマンドには、フレキシブルメッセージのターミネータを含むこともできる。

【００２８】

典型的なフレキシブルメッセージループの端部を示すコマンド

【表２】

コマンド名	内容	説明	利用法
EOMSG	“APC”	フレキシブルメッセージループのターミネータ	メッセージ提供システムは、この文字列をメッセージ提示システムに送信する。

【００２９】

一実施形態によれば、フレキシブルメッセージのターミネータは、”APC”の値を持つが、他の値を代わりに用いることもできることは明らかである。一般に、フレキシブルメッセージの値は、実際のフレキシブルメッセージにはあり得ないものとすべきである。メッセージ提供システムは、コマンドDEF_LEFTまたはコマンドINC_RIGHTのいずれかに応答して、フレキシブルメッセージループの端部に到達し、従ってメッセージ１またはメッセージＮのいずれかが直近に送信されたことを、メッセージ提示システムに示す、フレキシブルメッセージターミネータのEOMSG（端部のメッセージ）を送信する。メッセージ提示システムは、フレキシブルメッセージループのメッセージの数を知ることを必要としていないため、このメッセージを受け取ることで、ループの端部に到達したことを知る。フレキシブルメッセージループのターミネータに応答して、メッセージ提示システムは、静的メッセージを提示することができ、他の行為をとることもできる。

【００３０】

本発明の一実施形態によれば、フレキシブルメッセージループのターミネータを受け取った後に、どの静的メッセージが提示されるかを、現在ループを移動している方向によって変えることができる。ある実施形態では、ユーザが順方向または下方に進んでフレキシブルメッセージループの端部に到達した時（すなわち、フレキシブルメッセージループを、フレキシブルメッセージ１からフレキシブルメッセージＮまで移動した時）には、最初の静的メッセージ（すなわち静的メッセージ１）を提示し、ユーザが逆方向または上方に進んでフレキシブルメッセージループの端部に到達した時（すなわち、フレキシブルメッセージループを、フレキシブルメッセージＮからフレキシブルメッセージ１まで移動した時）には、最後の静的メッセージ（すなわち静的メッセージｍ）を提示する。

【００３１】

本発明の態様によれば、通信プロトコルには最小限の数の規則が含まれ、メッセージ提示デバイスが送信するメッセージは該規則に従う。例えば、一実施形態において、各フレキシブルメッセージには、最初のキャラクタであるACKキャラクタ（ASCII十進数値６）及び最後のNULLキャラクタ（ASCII十進数値０）が含まれて、メッセージ提示システムがメッセージの最初と最後を認識することができる。この実施形態において、最小のメッセージ長（ACK及びNULLキャラクタを除く）は１であるが、他の制約はほとんど課されることがない。例えば、この実施形態において、フレキシブルメッセージの数またはその内容の

数に上限は無く、フレキシブルメッセージの長さは、提示デバイスの能力（例えばディスプレイスクリーンのサイズ等）のみに制約される。

【００３２】

表１及び表２に記述されるフレキシブルメッセージループのコマンドの一式は、事実上例示であって、フレキシブルメッセージ及び静的メッセージの両方を用いる本発明の特定の実施形態のみに関連することが明らかである。代替的な実施形態において、メッセージループのコマンドの一式を変えることができる。例えば、メッセージの提示システムは、ループを一方向のみに動かすことも可能であるため、ユーザがメッセージループをナビゲート（すなわち、順方向または逆方向のいずれかであって、両方ではない）させる単一のボタンのみを備え、次のメッセージを要求する１つのコマンドのみ（例えばDEC_LEFTまたはINC_RIGHT）が提供／サポートされる必要がある。そのような一方向のループの移動のみが許容される実施形態では、メッセージ提示システムは、３つではなく２つのみのメッセージ（すなわち現行のメッセージ及び次のメッセージ）を格納することができる。あるいは、メッセージ提示システムの他の実施形態では、静的メッセージを持たず、全てのメッセージがフレキシブルメッセージであり、それゆえにメッセージのターミネータを提供する必要がない。かかる実施形態において、メッセージ提示システムは単に、ユーザによって選択された移動方向に基づいてフレキシブルメッセージループを移動し続けるのみである。

【００３３】

図３は、本発明の一実施形態により、ユーザと通信し、ユーザにメッセージを提示するプロセスを示す図である。該プロセスは、図５を参照して以下に詳細が説明されるメッセージ提示システムのプロセッサ等によって実行される。図３に示されるメッセージ通信プロセスの間、多数の変数が初期化され、プロセッサが、現行のメッセージをユーザに提示し、ユーザに提示する準備ができている他のメッセージ（ループの次のメッセージまたは、ループの１つ前のメッセージ）を有することができる状態を維持する。ユーザがメッセージループを順方向に進むのに応答し、プロセッサは次のメッセージを提示し、それからメッセージ提供システムに、メッセージループの順方向の次のメッセージを提供するように要求する。ユーザがメッセージループを両方向に動かすことが許容されている場合、ユーザが方向転換を決めた際に、逆方向または順方向のどちらにメッセージループの移動を開始する場合でも、プロセッサは、ユーザに１つ前のメッセージを提示する準備ができるようにしておく。ユーザが逆方向に移動するのに応答して、プロセッサは１つ前のメッセージを提示し、それからメッセージ提供システムに、メッセージループ逆方向の次のメッセージを提供するように要求する。以下、図３のメッセージ提示ルーチン300に関する、メッセージ通信プロセスを詳述する。

【００３４】

ブロック310で、プロセッサは多数の変数を初期化する。２つの反対の方向に動かす（すなわち、メッセージループを動かす）ための進行ボタンを備える本発明の一実施例において、ブロック３１０で初期化される変数には、ループの現行のメッセージ（CURR_MSG）の内容を格納する変数、ループの次のメッセージ（NXT_MSG）の内容を格納する変数、ループの１つ前のメッセージ（PREV_MSG）の内容を格納する変数が含むことができる。一実施形態によれば、現行のメッセージ（CURR_MSG）を格納する変数を初期化して、メッセージ提示システムの電源が投入された後に提示されるメッセージの内容にすることができる。このメッセージを、メッセージ提示システムのプロセッサがアクセス可能な不揮発性のメモリ内に前もって格納される静的メッセージとすることができ、また代わりに、メッセージ提供システムが提供するフレキシブルメッセージとすることもできる。次のメッセージ（NXT_MSG）を格納する変数を、最初のフレキシブルメッセージであるメッセージ１の内容に初期化することができ、１つ前のメッセージ（PREV_MSV）を、最後のフレキシブルメッセージであるメッセージＮの内容に初期化することができる。次のメッセージ（NXT_MSG）の変数の内容と、前のメッセージ（PREV_MSG）の変数の内容をローカルで格納することによって、ユーザが進行するメッセージループの方向に関わらず、メッセージ提示シ

ステムは、ループ中の次のメッセージに素早くアクセスしてユーザに提示することができる。

【００３５】

例えばある実施形態では、メッセージ提示システムはブロック310において、メッセージ提供システムに、MSG_Nコマンドの次にMSG_1コマンドを送信することができる。それぞれのコマンドに応答し、メッセージ提供システムは、メッセージ提示システムに、メッセージＮに続いてメッセージ１を送信し、メッセージ提供システムのカレントメッセージインデックスがメッセージ１を指すようにする。メッセージ提示システムはそれから、変数CURR_MSGを静的メッセージの内容に、PREV_MSGの内容をメッセージＮに、NXT_MSGの内容をメッセージ１に初期化し、メッセージ提示システムが、ユーザが進行する方向に関わらず、次のメッセージを遅延無くユーザに提示できるように、次のメッセージをローカルで利用可能とすることができる。

【００３６】

メッセージ提示ルーチン300はブロック320で、変数CURR_MSGの内容をユーザに提示した後に、ブロック330に進み、ユーザからのナビゲーションコマンドを待つ。ブロック330において、ユーザからナビゲーションコマンドを受け取っていない場合には、単に変数CURR_MSGの内容がユーザに対して提示（表示）される。あるいは、メッセージ提示ルーチンがユーザから受け取ったナビゲーションコマンドに応答して、ルーチンはブロック340に進み、次のメッセージを予測する。ブロック340では、ブロック330で受け取ったナビゲーションコマンドの方向に基づいて次のメッセージが予測される。例えば、ブロック330でのユーザから受け取ったナビゲーションコマンドが、順方向移動コマンド（例えば下、右またはインクリメント進行ボタンの押下）に一致した場合は、ルーチンはループ順方向の次のメッセージ（例えばメッセージ２）を次のメッセージと予測する。あるいは、ブロック330でユーザから受け取ったナビゲーションコマンドが、逆方向移動コマンド（例えば上、左またはデクリメント進行ボタンの押下）に一致した場合は、ルーチンはループ逆方向の次のメッセージ（例えばメッセージＮ－１）を次のメッセージと予測する。次のメッセージを予測した後、メッセージ提示ルーチンはブロック350に進む。

【００３７】

ブロック350で、メッセージ提示ルーチン300は、メッセージ提供システムに対してループの次のメッセージを要求する。例えば、ブロック340でループの次の予測メッセージが順方向であると判断された場合、メッセージ提示システムは、INC_RIGHTコマンドをメッセージ提供システムに送信する。このコマンドに応答して、メッセージ提供システムは、「カレントインデックス＋１」のメッセージ（例えばメッセージ２）を送信することができる。ブロック360で、メッセージ提示ルーチンは、現行のメッセージに対する変数（すなわちCURR_MSG）、次のメッセージに対する変数（NXT_MSG）及び１つ前のメッセージに対する変数（PREV_MSG）を更新する。従って、例えばユーザがループ順方向に移動した場合、現行のメッセージの変数（CURR_MSG）を、次のメッセージの変数（NXT_MSG）が以前格納していたものに更新し、１つ前のメッセージの変数（PREV_MSG）を、現行のメッセージの変数（CURR_MSG）が以前格納していたものに更新し、次のメッセージの変数（NXT_MSG）を、メッセージ提供システムから直近に受け取った内容（この例ではメッセージ２）に更新することができる。本発明の一実施形態によれば、ブロック360における変数の更新は単に、変数CURR_MSG、NXT_MSG及びPREV_MSGに対応する、メッセージ提示システムのメモリ位置のポインタを設定することだけで実行できる。ブロック360で変数を更新した後、メッセージ提示ルーチンはブロック320に進み、現行のメッセージの変数（CURR_MSG）の内容がユーザに対して提示される。その後はブロック320からブロック360までのプロセスが繰り返される。

【００３８】

下記に示す表３及び表４は、メッセージループを順方向及び逆方向に移動する時に、各々図３のメッセージ提示ルーチンに従い、現行のメッセージ（すなわちCURR_MSG）、次のメッセージ（NXT_MSG）及び１つ前のメッセージ（PREV_MSG）を表す変数の内容の典型例

を示す。表３及び表４において、値「Message0」は、メッセージ提示システムの電源投入時の初期化後に提示されるメッセージを示し、例えば静的メッセージとすることができる。表３及び表４に示されるように、電源投入時の初期化後、現行のメッセージの変数（すなわちCURR_MSG）、次のメッセージの変数（すなわちNXT_MSG）、及び１つ前のメッセージの変数（すなわちPREV_MSG）は、表の列「現在の値」に表される値を持つ。順方向の移動に対応するナビゲーションコマンドをユーザから受け取った場合に、これらの変数は表の列「インクリメント受取時」に示される値に更新される。一方、逆方向の移動に対応するナビゲーションコマンドをユーザから受け取った場合に、これらの変数は表の列「デクリメント受取時」に示される値に更新される。

【００３９】

ループの順方向における典型的な変数の状態

【表３】

状態	変数	現在の値	ｲﾝｸﾘﾒﾝﾄ受取時	ﾃﾞｸﾘﾒﾝﾄ受取時
初期化	CURR_MSG	メッセージ０	メッセージ１	メッセージN
	PREV_MSG	メッセージN	メッセージ０	メッセージ０
	NXT_MSG	メッセージ１	メッセージ２	メッセージN－１
状態１	CURR_MSG	メッセージ１	メッセージ２	メッセージ０
	PREV_MSG	メッセージ０	メッセージ１	メッセージ１
	NXT_MSG	メッセージ２	メッセージ３	メッセージN
状態２	CURR_MSG	メッセージ２	メッセージ３	メッセージ１
	PREV_MSG	メッセージ１	メッセージ２	メッセージ２
	NXT_MSG	メッセージ３	メッセージ４	メッセージ０
状態N－１	CURR_MSG	メッセージN－１	メッセージN	メッセージN－２
	PREV_MSG	メッセージN－２	メッセージN－１	メッセージN－１
	NXT_MSG	メッセージN－３	メッセージ０	メッセージN－３

【００４０】

ループの逆方向における典型的な変数の状態

【表４】

状態	変数	現在の値	ｲﾝｸﾘﾒﾝﾄ受取時	ﾃﾞｸﾘﾒﾝﾄ受取時
初期化	CURR_MSG	メッセージ０	メッセージ１	メッセージＮ
	PREV_MSG	メッセージＮ	メッセージ０	メッセージ０
	NXT_MSG	メッセージ１	メッセージ２	メッセージＮ－１
状態１	CURR_MSG	メッセージ１	メッセージ２	メッセージ０
	PREV_MSG	メッセージ０	メッセージ１	メッセージ１
	NXT_MSG	メッセージ２	メッセージ３	メッセージＮ
状態２	CURR_MSG	メッセージ２	メッセージ３	メッセージ１
	PREV_MSG	メッセージ１	メッセージ２	メッセージ２
	NXT_MSG	メッセージ３	メッセージ４	メッセージ０
状態Ｎ－１	CURR_MSG	メッセージＮ－１	メッセージＮ	メッセージＮ－２
	PREV_MSG	メッセージＮ－２	メッセージＮ－１	メッセージＮ－１
	NXT_MSG	メッセージＮ－３	メッセージ０	メッセージＮ－３

【００４１】

表３及び４を見ると、ある状態から次の状態に遷移する際に、メッセージの内容の２つは依然同じものであることから、ポインタを動かすだけで、現行のメッセージ（すなわちCURR_MSG）、次のメッセージ（すなわちNXT_MSG）及び１つ前のメッセージ（すなわちPREV_MSG）を効率的に更新することができる。現行のメッセージ、次の予測メッセージ及び１つ前のメッセージのコピーを、メッセージ提示システムのローカルのメモリに維持することで、メッセージ提示システムは、ユーザからのあらゆる入力に対して素早く応答することができるとともに、利用するシステムリソースの量を最小限とすることができることが明らかである。

【００４２】

図４は、本発明の一実施形態におけるラックマウント可能な無停電電力装置の前面図を示すものであり、本発明の様々な態様を実施することができ、図１のメッセージ提示システム120に対応し、図２、３を参照して上述したフレキシブルメッセージループを実行する。図４に示すように、無停電電力装置（ＵＰＳ）システム400は、様々なユーザがアクセス可能な機能要素を備え、ＵＰＳシステムのユーザに情報を伝え、ユーザとＵＰＳシステム400との対話を可能とする。これらの機能要素には、ディスプレイスクリーン410、進行ボタン420及び430、電源スイッチ440、セットアップボタン450、ステータスボタン460及び複数のステータスインジケータ470が含まれる。ある実施形態によれば、ディスプレイスクリーン410には、各行２０文字表示可能な、２行表示の真空蛍光灯ディスプレイを含むことができるが、異なるサイズのディスプレイまたは異なる種類のディスプレイを代わりに用いることができ、本発明は特定のサイズや特定の種類のディスプレイに限定されるものではないことは明らかである。

【００４３】

本発明の態様によれば、ディスプレイ410は、ステータス及び設定情報をユーザに伝えるために用いられ、ボタン420、430、450、460は、様々な機器設定や動作ステータスを決定したり、様々な機器設定をセットしたり、変更したり、調節したりするために用いられる。例えば、ユーザは進行ボタン420及び430によって、ＵＰＳシステムと対話することが可能となり、例えば、ＵＰＳシステムの動作状態または設定情報を反映する一連のメッセ

ージまたは他の情報を、上方または下方にスクロールさせたり、特定のモードにおいて、設定値を上下させたり、設定の有効、無効を切り替えたりすることができる。好ましくは、２つの対立する方向（例えば上下、入出、左右）を表す２つの進行ボタンが設けられ、ユーザは、情報をいずれかの方向にもスクロールさせることができる。しかしながら、以下に詳述するように、本発明は、異なる２つの対立する進行ボタンを使用するものに限定されるものではなく、例えば１つのみの進行ボタンを設けることもできる。本発明の態様によれば、進行ボタン420及び430によって、ディスプレイ410によってユーザに提示される可変の内容のメッセージの量を、実質的に無制限とすることができる。さらに、ＵＰＳシステム400は、ＵＰＳシステム400のディスプレイ410に提示されるメッセージの内容、メッセージの数及び表示されるメッセージの順序を、ランタイム前に知っている必要はない。

【００４４】

ＵＰＳシステムがラックにマウントされて、ＵＰＳシステムの前面のみがアクセス可能な時でも、ユーザはＵＰＳシステム400のフロントパネルに配置された電源スイッチ440に容易にアクセスすることができる。ユーザはセットアップボタン450によって、様々な機器設定及び動作を決定することができる。また、ユーザはステータスボタン460によって、様々な電気的測定値、環境的測定値、動作ステータスメッセージ、バッテリー状態メッセージ及び他のシステム情報を順次参照することができる。複数のステータスインジケータ470は、ユーザへの優先順位がより高い特定の情報を提示するために用いられ、その情報は例えば、電源がバッテリーから供給されているか、ＵＰＳシステムがオンラインであるか、ＵＰＳシステムのバッテリーの残寿命がある量を下回っているか、ＵＰＳシステムに供給されるＡＣ電源が許容できる動作範囲内にあるか、ノイズフィルタリングが動作しているか等の情報である。

【００４５】

本発明の態様によれば、ＵＰＳシステム400は、複数のフレキシブルメッセージをディスプレイ410に提示させることができ、ＵＰＳシステムはランタイム前に、フレキシブルメッセージの数、順序、内容を知らず、すなわち、フレキシブルメッセージはＵＰＳシステムの固定記憶に格納されない。一実施形態において、フレキシブルメッセージは、ＤＲＡＭ等の動的メモリに格納され、ＵＰＳシステムは、現行のディスプレイスクリーン、次のディスプレイスクリーン、１つ前のディスプレイスクリーンに対応する３つのメモリ位置のみを常に動的に格納する。フレキシブルメッセージを提示している間、ＵＰＳシステム400は、図３を参照して上述した、メッセージ提示ルーチンを実行する。例えば、ＵＰＳシステム400は、ユーザがダウン進行ボタン430を押下したことに反応し、ループ下方または順方向に存在する、次のメッセージを提示して、メッセージ提供システムにINC_RIGHTコマンドを送信することができ、また、ＵＰＳシステム400は、ユーザがアップ進行ボタン420を押下したことに対応し、ループの上方または逆方向に存在する、１つ前のメッセージを提示して、メッセージ提供システムにDEC_LEFTコマンドを送信することができる。

【００４６】

本発明の様々な実施形態では、１つ以上のプロセッサベースシステムを実装することができ、例えば１つのプロセッサベースＵＰＳシステムを実装することができる。これらのプロセッサベースシステムでは、例えば、インテル社のPentium型のプロセッサ、Motorola社のPowerPCプロセッサ、サン・マイクロシステムズ社のUltraSPARCプロセッサ、Hewlett-Packard社のPA-RISCプロセッサ、または他のあらゆる種類のプロセッサ等をベースとする汎用プロセッサを利用することができる。あるいは、これらのプロセッサベースシステムは、専用プロセッサまたはＡＳＩＰ（例えば、ＳＴマイクロエレクトロニクス社の８ビットのST72F321プロセッサ、または１６ビットのSTM32F102プロセッサ）をベースとすることができる。そのようなプロセッサベースシステムで実行されるソフトウェアは、ＵＰＳシステム等のシステムの動作を、用途の範囲内で制御するコントローラとして振る舞う。そのようなソフトウェアには、本発明によるメッセージングシステム及びメッセージングプロトコルを実装する、特殊化されたソフトウェアのコードが含まれる。本発明の実施

形態を実施し、図４のＵＰＳシステムの一部を構成する、典型的なプロセッサベースシステムについて、以下図５を参照して説明する。

【００４７】

プロセッサベースシステム500は一般に、バス560によって相互接続される、プロセッサ510、メモリ520、通信インタフェース530、入出力サブシステム540及び、ディスプレイサブシステム550を備える。上述したように、プロセッサ510は汎用プロセッサ、または他のあらゆる種類の既知のプロセッサとすることができ、本発明は特定の種類のプロセッサのいずれにも限定されるものではない。一実施形態では、プロセッサはＳＴマイクロエレクトロニクス社の８ビットのST72F321プロセッサである。メモリ520を用いて、プロセッサベースシステム500の動作中に、プログラム及びデータを格納しておくことができる。概してメモリ520には、比較的高性能で揮発性のＲＡＭ（例えばダイナミック・ランダム・アクセス・メモリ（ＤＲＡＭ））や、より固定型のメモリ（ＲＯＭ、ＰＲＯＭ、ＥＥＰＲＯＭ等）が含まれるが、ディスクドライブまたは他の不揮発性のストレージデバイス等のあらゆるデータ格納デバイスもメモリ520に含まれ得る。

【００４８】

プロセッサベースシステム500は、通信インタフェース530を使用して、図１のメッセージ提供システム110等の他のデバイスと、無線または有線の任意の通信媒体で通信を行うことができる。例えば、通信インタフェースを、無線または有線のイーサネット通信インタフェース、ユニバーサル・シリアル・バス（ＵＳＢ）インタフェース、ＲＳ－２３２、ＲＳ－４２２または、ＲＳ－４８５シリアルインタフェース、ＩＥＥ－１２８４パラレルインタフェース、またはデータを送受信できる任意の他の種類の通信インタフェースとすることができる。

【００４９】

プロセッサベースシステム500は、入出力サブシステム540を使用して、ユーザからの入力を受け取り、及び/またはユーザへの出力を提供することができる。様々な入力デバイス（図４の進行ボタン420、430等）から入力が行われ、様々な出力デバイス（図４のインジケータ470等）に出力される。トラックボール、マウス、キーボード等の他の種類の入力デバイス及び出力デバイスを使用することも可能であり、本発明は特定の入力デバイスまたは出力デバイスに限定されるものではないことが明らかである。

【００５０】

プロセッサベースシステムは、図４に示されるディスプレイ410等のディスプレイサブシステム550によって、ユーザにメッセージを伝える。動作時にプロセッサ410は、ユーザに対する提示情報をディスプレイサブシステムに供給する。一実施形態によれば、提示情報には、各行最大２０文字の２行の情報を含むことができる。他の種類の提示デバイスを代わりに使用することができ、他の種類のディスプレイ（例えばビットマップディスプレイ等）を代わりに使用することもできることが明らかである。

【００５１】

バス560には、システム要素間のあらゆる通信の組合わせが含まれ、専用または標準のコンピュータのバス技術（例えばＩＤＥ、ＳＣＳＩ、ＰＣＩ及び、ＩｎｆｉｎｉＢａｎｄ）が含まれる。従って、バス560は、プロセッサベースシステム400のシステム要素間で通信（例えばデータ及び命令）のやりとりを行うことを可能とする。

【００５２】

動作中、プロセッサ510は、他の作業の合間に、図３を参照して上述したようなメッセージ提示ルーチンを実施することができる。例えばプロセッサ510は、現行のメッセージ、次のメッセージ及び、１つ前のメッセージの内容をメモリ520に格納することができ、現行のメッセージの内容をディスプレイサブシステム550に送信し、ユーザに対し提示することができる。ユーザの進行ボタン押下に応答し、入出力システム540はプロセッサ510に進行ボタン押下のイベントを通知することができる。この通知に応答して、プロセッサ510は次のメッセージの内容をディスプレイサブシステム550に送信し、ユーザに対して表示することができ、通信インタフェース530に対してコマンドを送信してメッセージ提供

システムにループの次のメッセージを要求することができる。
【００５３】
本発明の様々な態様及び機能が実施される、ある種類のプロセッサベースシステムを例として、プロセッサベースシステムが示されるが、本発明の態様は、図５に示されるシステムで実施されるものに限定されるものではない。本発明の様々な態様及び機能を、図５に示されるシステムとは異なるアーキテクチャ及びコンポーネントを持つ１つ以上のプロセッサベースシステムで実施することができる。
【実施例１】
【００５４】
典型的なフレキシブルメッセージングアルゴリズム
下記に、本発明のフレキシブルメッセージングを実行するための、Ｃプログラミング言語で記述されたアルゴリズムの典型例が示される。本アルゴリズムは特に、各行２０文字の、２行のディスプレイスクリーンに使用される。このアルゴリズムでは、２行のメッセージが同時に提示され、現行のディスプレイスクリーン、１つ前のディスプレイスクリーン、次のディスプレイスクリーンを格納する記憶アレイが設けられ、各々の記憶アレイは２行のメッセージを持つことができる。このアルゴリズムは最小限のメモリを使用し、あらゆるプロセッサで効率的に実行することができる。例えば、８ビットのST72F321プロセッサにおいて、このアルゴリズムが要求するメモリの量は134バイトのみで、各42ビットの３つ（現行のメッセージ、次のメッセージ及び１つ前のメッセージ）の表示情報を格納し、適切な変数及びポインタを維持し、両方向の移動を可能とするためのものである。この典型的なアルゴリズムには、一方向（例えば下方または順方向）のみの動作を行う機能性も含まれており、その場合、２つのみの表示情報が格納される。以下、典型的なアルゴリズムに定義される様々な関数の概要を説明する。
【００５５】
関数：FlxScrnCircle
この関数は、ユーザが順方向または下方に移動する際に、物理メモリの正しい領域の位置にポインタを操作するものである。より詳しくは、この関数はポインタを操作し、１つ前の表示を示すポインタ（pPrevScrn）を、以前「現行の提示情報」を格納していたメモリ位置とし、現行の表示を示すポインタ（pCurrScrn）は、以前「次の表示情報」を格納しているメモリ位置の位置を示し、次の提示を示すポインタ（pCurrLine）は、以前「１つ前の表示情報」のメモリ領域の１番目のメッセージラインを示す。
【００５６】
関数：FlxScrnCircle
この関数は、FlxScrnCircleとは正反対の振る舞いをし、ユーザが逆方向または上方に移動する際に、物理メモリの正しい領域の位置にポインタを操作するものである。より詳しくは、この関数はポインタを操作し、１つ前の表示を示すポインタ（pPrevScrn）を、以前「現行の提示情報」を格納していたメモリ位置とし、現行の表示を示すポインタ（pCurrScrn）は、以前「次の表示情報」を格納しているメモリ位置の位置を示し、次の提示を示すポインタ（pCurrLine）は、以前「１つ前の表示情報」のメモリ領域の２番目のメッセージラインを示す。
【００５７】
関数：Flx_IsItEndLoop
この関数は、「次の表示情報」を格納するメモリ位置を示すポインタ（pCurrLine）を操作し、移動の方向（順方向または逆方向）に基づき、正しい行（メッセージ）を指すようにする。従って、順方向に移動している場合には、ポインタは第１の行（メッセージ）sTextを指し、一方、逆方向に移動している場合には、ポインタは第２の行（メッセージ）sText2を指す。ポインタを操作した後、この関数は、フレキシブルメッセージのターミネータを、受け取っているかをチェックする。フレキシブルメッセージのターミネータを受け取ると、関数FlxEndLoopProcessが呼び出され、ポインタをNULL状態に設定する。
【００５８】

関数：FlexStartFirstLine
この関数は、ルーチンが最初のフレキシブルメッセージから開始された場合に、メッセージ提供システムから、最初のフレキシブルメッセージ（フレキシブルメッセージ１）を取得する。すなわち、静的メッセージのブロックを順方向に抜け出ることにより、このルーチンが開始された場合である。
【００５９】
関数：FlexStartFromLastLine
この関数は、ルーチンが最後のフレキシブルメッセージＮから開始された場合に、メッセージ提供システムから、最後のフレキシブルメッセージ（フレキシブルメッセージＮ）を取得する。すなわち、静的メッセージのブロックを逆方向に抜け出ることにより、このルーチンが開始された場合である。
【００６０】
関数：DefaultStatusPrevFlexible
この関数は、ユーザが静的メッセージングに移行する際に、静的メッセージを扱うために従来のレガシーコードを使用できるようにする。この関数は、関数InitialTheLastFlexibleMessageによってシステムの電源投入後一度だけ呼び出されて、フレキシブルメッセージングに使用される変数を初期化する。従って、静的メッセージを扱うために従来のレガシーコードが使用され、一方、フレキシブルメッセージに対してはフレキシブルメッセージングのアルゴリズムが用いられる。
【００６１】
関数：InitialTheLastFlexibleMessage
この関数の目的は、システムの起動を円滑にすることである。システムの電源投入後、この関数は予め設定された期間待ち、それから、変数の初期化を開始し、その後関数DefaultStatusPrevFlexibleを呼び出し、フレキシブルメッセージの提示が、最初のフレキシブルメッセージと最後のフレキシブルメッセージのいずれから開始されているかを判断する。
【００６２】
関数：FlexibleInnerLoopNormal
この関数は、関数FlexibleStringInnerLoopによって呼び出され、標準的な内部ループの条件をプロセスし、すなわち、ユーザがフレキシブルメッセージループの中間にいる場合に、順方向に移動している場合及び逆方向に移動している場合に関わらずフレキシブルメッセージループから抜け出し、静的メッセージループに入る。この関数は、移動の方向に基づき、INC_RIGHT_CMDコマンドまたはDEC_LEFT_CMDコマンドを送信し、メッセージ提供システムに次のメッセージを要求する。またこの関数は、関数FlexibleStringInnerLoopによって呼び出される。
【００６３】
関数：FlexibleStringInnerLoop
この関数は、必要に応じて他のすべてのコードセグメントを呼び出す、主要なコードセグメントである。この関数は、ユーザが「上」ボタンを押した場合にはIncFxnPtr、「下」ボタンを押し場合にはDecFxnPtrが呼び出す別セクションのコードによって、呼び出される。
【００６４】

```
//============================================================
//============================================================

#define FLEXIBLE_BUFFERS 1 // フレキシブル表示機能
                           // フレキシブル表示機能の無効時には
                           //０とする

#if FLEXIBLE_BUFFERS
```

```
#define FULL_LINE_SIZE_BUFF 21 // ディスプレイの物理的サイズに１を加えて
                               //決定される

#define BOTH_UPDOWN_DIRECTION 1 // 両方向の移動を可能とする
                                // 下方／右方のみの移動を行う場合は０を設定する

// 構造体内のバッファsText_nの数は、提示されるメッセージラインの数で決定される。
// FULL_LINE_SIZE_BUFFは、１つのメッセージ゛の最大サイズに１を加えた値とすべきで
ある。
//
// 下記の構造体は、ディスプレイに２行提示する構造体FLEXBUFFの定義の典型例である
。
typedef struct FLEX_BUFF {

 char sText[FULL_LINE_SIZE_BUFF];  // 第１の表示メッセージラインを格納する
 char sText2[FULL_LINE_SIZE_BUFF]; // 第２の表示メッセージラインを格納する

} FLEXBUFF;

// 構造体FLEXSCNを定義する

typedef struct FLEXIBLE_SCREEN {

 char *pCurrLine; // 次に来るメッセージを格納するバッファのポインタ

 FLEXBUFF *pCurrScrn; // 現行のディスプレイ表示のポインタ
 FLEXBUFF *pPrevScrn; //  １つ前のディスプレイ表示のポインタ
 FLEXBUFF sLine[3];   // 本機能を扱うためには３つのFLEXBUFFが必要となる。

 char bScreenIndex; // 表示制御パラメータ
 // ０、１、２のいずれかの値を持つ、配列sLine[]のインデックスである

 // メッセージ制御用の、メッセージ及び表示の状態変数
 UBYTE bLoopCnt;
 UBYTE bStatus;

 // メッセージ要求のトラッキングの記録
 UBYTE bIncMsgReq; // メッセージのインクリメント要求をカウントする

#if BOTH_UPDOWN_DIRECTION
 UBYTE bDecMsgReq; // メッセージのデクリメント要求をカウントする
#endif


} FLEXSCN;

// ************ bStatus用の定義 ***************

#define fBackEndLoop 0x80 // 第７ビット  1: セットされている場合、逆方向に移動し
てループの端部に到達したことを示す
```

```
#define fAcknowledged 0x40 //第６ビット  1: ACKキャラクタの受信、0:未受信
#define fNotInitialState 0x20 //第５ビット  1: 初期状態でない, 0: 初期状態
#define fEndLoop 0x10 //第４ビット  1: フレキシブル内部ループの端部 (EOMSG)
#define fScrnReady 0x08 //第３ビット  1: 次のメッセージを表示する準備ができてい
る
#define f1stLine 0x04 //第２ビット  1: １行目, 0: ２行目; 下記の備考を参照
#define fDefaultScrn 0x02 //第１ビット  1: 負／デクリメント／逆方向
#define fIncfunc 0x01 //第０ビット  1: IncFxn, 0: DecFxn;
// 備考：１ビットの代わりに１バイトが使用された場合、256行を収容できる。

FLEXSCN FlxScn; // 物理メモリをここに割り当てる

//============================================================
//
// 関数: FlxScrnCircle ( )
//
// 説明: 表示のインクリメント／右回りの移動
//
// 入力:
// char bScrnIndex - 表示インデックス番号 ( 0, 1, 2 )
//
//============================================================
//
void FlxScrnCircle ( char bScrnIndex )
{
 FlxScn.pCurrScrn = &(FlxScn.sLine[ bScrnIndex ]);
 FlxScn.pPrevScrn = FlxScn.pCurrScrn++;

 if ( bScrnIndex == 2 ) // この条件が満たされたときのみ、次の命令文を実行する
  FlxScn.pCurrScrn = &(FlxScn.sLine[0]);

 // 新しいメッセージを受け取るバッファのポインタ
 FlxScn.pCurrLine =
 FlxScn.pCurrScrn->sText;
}

//============================================================
//
// 関数: FlxScrnCounterCircle ( )
//
// 説明: 表示のデクリメント／左回りの移動
//
// 入力:
// char bScrnIndex - 表示インデックス番号 ( 0, 1, 2 )
//
//============================================================
//
void FlxScrnCounterCircle ( char bScrnIndex )
{
 FlxScn.pCurrScrn = &( FlxScn.sLine[ (2 - bScrnIndex) ] );
```

```
 FlxScn.pPrevScrn = FlxScn.pCurrScrn--;
 if ( bScrnIndex >= 2 ) // この条件が満たされたときのみ、次の命令文を実行する
  FlxScn.pCurrScrn = &(FlxScn.sLine[2]);

 //新しいメッセージを受け取るバッファのポインタ
 FlxScn.pCurrLine = FlxScn.pCurrScrn->sText2;
}

//=========================================================
//
// 関数: Flx_IsItEndLoop ( )
//
// 説明: フレキシブルメッセージループの端部かどうか確かめる
//
// 入力:
// char bScrnIndex - 表示インデックス番号 ( 0, 1, 2 )
// UBYTE fIncFxn - 0の場合は上（左）に移動し、その他の場合は下（右）に動かす
//
//=========================================================
//
void Flx_IsItEndLoop ( char bScrnIndex, UBYTE fIncFxn )
{
 FLEXBUFF *pScrn; // 適切な表示ポインタ
 char *pStr;

 if ( !fIncFxn ) //上（左）に移動している場合
 {
  pScrn = &( FlxScn.sLine[ (2 - bScrnIndex) ] );
  pStr = pScrn->sText2; }
 else //下（右）に移動している場合
 {
  pScrn = &( FlxScn.sLine[ bScrnIndex ] );
  pStr = pScrn->sText;
 }

 if ( !*pStr || !strcmp( pStr, "APC" ) )
  FlxEndLoopProcess ( );
}

//=================================================
//
// 関数: FlxEndLoopProcess ( )
//
// 説明: フレキシブルメッセージループの端部のプロセス
//
// 入力:なし
//
//=================================================
//
void FlxEndLoopProcess ( void )
```

```
{
 if ( ValTheBit(FlxScn.bStatus, f1stLine) )
 {
  if ( FlxScn.pCurrLine == FlxScn.pCurrScrn->sText )
   FlxScn.pCurrScrn->sText2[0] = 0;
  else
   FlxScn.pCurrScrn->sText[0] = 0;
  }
#if NO_TERMINATER_DISPLAY
 *FlxScn.pCurrLine = 0;
#endif
 // 端部を表す文字列を受け取る
 SetTheBit( FlxScn.bStatus, (fEndLoop | fScrnReady) );
}

//=============================================================
//
// 関数: FlexStartFirstLine ( )
//
// 説明: 最初のフレキシブルメッセージラインの要求を設定する
//
// 入力: なし
//
//=============================================================
//
void FlexStartFirstLine ( void )
{
 ClrTheBit( FlxScn.bStatus, fScrnReady );
 FlxScrnCircle ( FlxScn.bScreenIndex );
#if BOTH_UPDOWN_DIRECTION
 SetTheBit(FlxScn.bStatus, (fIncfunc | f1stLine | fBackEndLoop) );
#else
 SetTheBit(FlxScn.bStatus, f1stLine );
#endif

 SendInsertCmd ( INC_MSG_1_CMD );
}

//=============================================================
//
// 関数: FlexStartFromLastLine ( )
//
// 説明: 最後のフレキシブルメッセージラインの要求を設定する
//
// 入力: なし
//
//=============================================================
//
void FlexStartFromLastLine ( void )
{
```

```
 FlxScrnCounterCircle ( FlxScn.bScreenIndex );
#if BOTH_UPDOWN_DIRECTION
 SetTheBit( FlxScn.bStatus, (flstLine | fBackEndLoop) );
 ClrTheBit(FlxScn.bStatus, fIncfunc);
#else
 SetTheBit( FlxScn.bStatus, flstLine );
#endif

 SendInsertCmd ( DEC_MSG_N_CMD );
 }

//============================================================
//
// 関数: DefaultStatusPrevFlexible ( )
//
// 説明: 最後のフレキシブルメッセージラインを、既定状態に設定する
//
// 入力：なし
//
//============================================================
//
void DefaultStatusPrevFlexible ( void )
{
 if ( !ValTheBit( FlxScn.bStatus, fDefaultScrn ) )
 {
 if ( !ValTheBit( bbmCommFlags, fUI_uLinkMode ) ) // レガシーコードの条件
                                                  //が満たされる場合、
                                                  //付加的な作業を行う
 {

#if BOTH_UPDOWN_DIRECTION
 if ( EE_default_screen < 4 )
  FlexStartFromLastLine ( );
 else
#endif
 FlexStartFirstLine ( );
 }

 SetTheBit( FlxScn.bStatus, fDefaultScrn );
 }
}

//============================================================
//
// 関数: InitialTheLastFlexibleMessage ( )
//
// 説明: 最後のフレキシブルメッセージ用の初期化
//
// 入力：なし
//
```

```
//============================================================
// void InitialTheLastFlexibleMessage ( void )
{
 if ( !ValTheBit( FlxScn.bStatus, fNotInitialState ) )
 {
  if ( FlxScn.bLoopCnt++ >= 126 ) // 下記のプロセスを遅延させ、最小化させる
                                  //タイミング制御文
  {
   SetTheBit( FlxScn.bStatus, fNotInitialState );
   ClrTheBit( FlxScn.bStatus, fDefaultScrn ); DefaultStatusPrevFlexible ( );
  }
 }
}

//============================================================
//
// 関数: FlexibleInnerLoopNormal ( )
//
// 説明: 標準の内部ループの条件プロセス
//
// 入力：
// UBYTE max_funcs - 関数メニューの項目数
// UBYTE fIncFxn - 1の場合、IncFxnPtr( )から呼び出されている
//                 その他の場合、DecFxnPtr( )から呼び出されている
// char bScrnIndex - 表示インデックス番号( 0, 1, 2 )
//
//============================================================
//
void FlexibleInnerLoopNormal ( UBYTE max_funcs,
                               UBYTE fIncFxn,
                               char bScrnIndex )
{
 SetTheBit( FlxScn.bStatus, f1stLine ); // 第一の表示ラインを満たす処理の開始

#if BOTH_UPDOWN_DIRECTION
 if ( fIncFxn )
 {
  SetTheBit( FlxScn.bStatus, fIncfunc );

  FlxScrnCircle ( bScrnIndex );

  bFuncIndex = (max_funcs-4) + (bScrnIndex << 1);

  SendInsertCmd ( INC_RIGHT_CMD );
 }
 else
 {
 ClrTheBit( FlxScn.bStatus, fIncfunc );

 FlxScrnCounterCircle ( bScrnIndex );
```

```
 bFuncIndex = max_funcs - (bScrnIndex << 1);

 SendInsertCmd ( DEC_LEFT_CMD );
 }

 // 即座に静的メッセージの状況を返答するように制御する
 if ( ValTheBit( FlxScn.bStatus, fDefaultScrn ) )
  ClrTheBit( FlxScn.bStatus, fDefaultScrn );

#else

 FlxScrnCircle ( bScrnIndex );

 bFuncIndex = (max_funcs-4) + (bScrnIndex << 1);

 SendInsertCmd ( INC_RIGHT_CMD );

#endif // BOTH_UPDOWN_DIRECTIONのブロック

}

//=============================================================
//
// 関数: FlexibleStringInnerLoop ( )
//
// 説明: インクリメントまたはデクリメントのファンクションキーが押された場合に、
//       終了シグナルまで、ループ中のフレキシブルメッセージをプロセスする
//
// Input:
// UBYTE max_funcs - 機能メニューの項目数
// UBYTE fIncFxn -  1の場合、IncFxnPtr( )から呼び出されている
//                  その他の場合、DecFxnPtr( )から呼び出されている
//
//=============================================================
void FlexibleStringInnerLoop ( UBYTE max_funcs, UBYTE fIncFxn )
{
 char bScrnIndex; // より柔軟にするため、ローカル変数を使用する

 // 新しいスクリーンメッセージを更新するため、フラグをクリアする
 ClrTheBit( FlxScn.bStatus, fScrnReady );

 bScrnIndex = FlxScn.bScreenIndex + 1;
 if ( bScrnIndex > 2 )
  bScrnIndex = 0;

#if BOTH_UPDOWN_DIRECTION

 // 内部ループの中にいる場合、方向転換のプロセスを行う
if ( fIncFxn != ValTheBit( FlxScn.bStatus, fIncfunc ) )
```

```
 {
  if ( bScrnIndex < 2 )
   bScrnIndex {circumflex over ( )}= 0x01; // 高速に切り替える技術を使用している

  // ここでは、押されるキーの方向が変更されたかチェックし、
  // 現在の表示にはターミネータが含まれるかをチェックする

  if ( ValTheBit( FlxScn.bStatus, fBackEndLoop ) )
  {
   SetTheBit( FlxScn.bStatus, fEndLoop );
   }
   else
   {
    // 方向が変更されたため、以前のEndLoopはもはや有効でない
    if ( ValTheBit( FlxScn.bStatus, fEndLoop ) )
    {
     ClrTheBit( FlxScn.bStatus, fEndLoop );
    }
    if ( ( !fIncFxn && (FlxScn.bIncMsgReq == 5) )
      || ( fIncFxn && (FlxScn.bDecMsgReq == 5) ) )
     Flx_IsItEndLoop ( bScrnIndex, fIncFxn );
    }
   }
#endif // BOTH_UPDOWN_DIRECTIONのブロック

 FlxScn.bScreenIndex = bScrnIndex; // 現行の表示を更新する

 if ( !ValTheBit( FlxScn.bStatus, fEndLoop ) )
 {
  FlexibleInnerLoopNormal ( max_funcs, fIncFxn, bScrnIndex );
 }
 else // if ( ValTheBit( FlxScn.bStatus, fEndLoop ) )
 {
 // フレキシブルメッセージループから静的メッセージに移動した場合、
 // フレキシブルメッセージを開始する準備をする必要はなく、
 // 固定メニューからフレキシブルメニューに移動するキーが押されたときに、
 //表示の準備はできている

 ClrTheBit( FlxScn.bStatus, fEndLoop );

#if BOTH_UPDOWN_DIRECTION

 if ( !fIncFxn )
 {
  bFuncIndex = max_funcs - 6;
  ClrTheBit( FlxScn.bStatus, fIncfunc );
 }
 else
 {
 bFuncIndex = 0;
```

```
 SetTheBit( FlxScn.bStatus, fIncfunc );
 }

#else

 bFuncIndex = 0;

#endif // BOTH_UPDOWN_DIRECTIONのブロック
 if ( FlxScn.bIncMsgReq & 1 )
 SetTheBit( FlxScn.bStatus, fScrnReady );
 }
 }

#endif // FLEXIBLE_BUFFERSのブロック
```

【００６５】

上記のアルゴリズムは、メッセージ提供システムに次のフレキシブルメッセージ（例えばメッセージｉ）を要求し、該メッセージを次のディスプレイスクリーンを示すメモリ位置に設定し、ポインタ*pCurrLineを設定する。例示する実施形態では、各々のディスプレイスクリーンは２つのメッセージを含むことができるため、現在の移動方向の次のメッセージ（例えばメッセージｉ＋１または例えばメッセージｉ－１）も要求される。一実施形態では、この次のメッセージは下記のコードを含むProcess_FlexibleStringUpdateと呼ばれる関数により要求される。

【００６６】

```
#if BOTH_UPDOWN_DIRECTION

 if ( ValTheBit(FlxScn.bStatus, fIncfunc) )
 {
  FlxScn.pCurrLine = FlxScn.pCurrScrn->sText2;
  SendInsertCmd ( INC_RIGHT_CMD ); // メッセージ提供システムからの
                                   // フレキシブル文字列の更新要求

 else
 {
  FlxScn.pCurrLine = FlxScn.pCurrScrn->sText;
  SendInsertCmd ( DEC_LEFT_CMD ); // メッセージ提供システムからの
                                  // フレキシブル文字列の更新要求
 }
#else

 FlxScn.pCurrLine = FlxScn.pCurrScrn->sText2;
 SendInsertCmd ( INC_RIGHT_CMD ); // メッセージ提供システムからの
                                  // フレキシブル文字列の更新要求
#endif
```

【００６７】

この例示的な実施形態では、下記のコードのセグメントにより、DEC_LEFT_CMDまたはINC_RIGHT_CMDが通信インタフェースに提供され、メッセージ提供システムに送信される際に、関数Process_FlexibleStringUpdateが呼び出される。

【００６８】

```
#if FLEXIBLE_BUFFERS
```

```
case DEC_LEFT_CMD:
case INC_RIGHT_CMD:
case DEC_MSG_N_CMD:
case INC_MSG_1_CMD:
Process_FlexibleStringUpdate ( bRxChar );
break;

#endif
```

【００６９】

本発明の少なくとも１つの実施形態の各々の態様を記載したが、当業者は様々な置換、変更及び改善をたやすく行うことができることは明らかである。そのような置換、変更及び改善は本開示の一部と解釈され、本発明の範囲に含まれることは明らかである。従って、上述の記載は単なる例示のみを目的とするものである。

【図１】

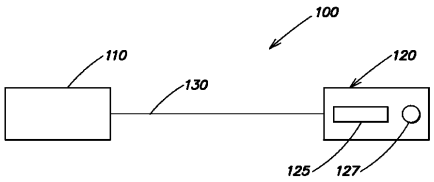
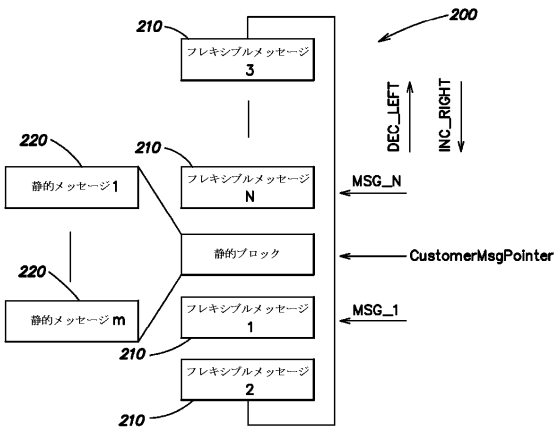
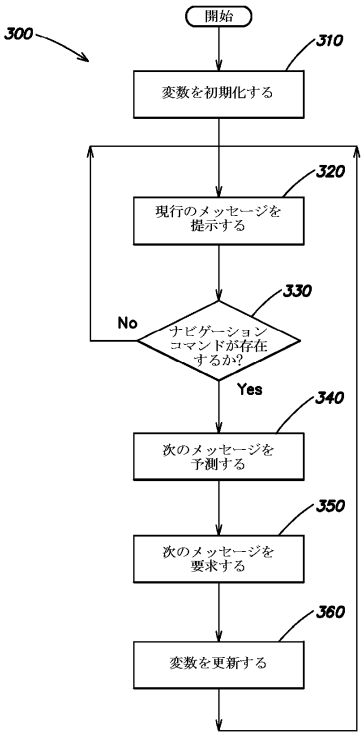


FIG. 1

【図２】



【図３】



【図４】

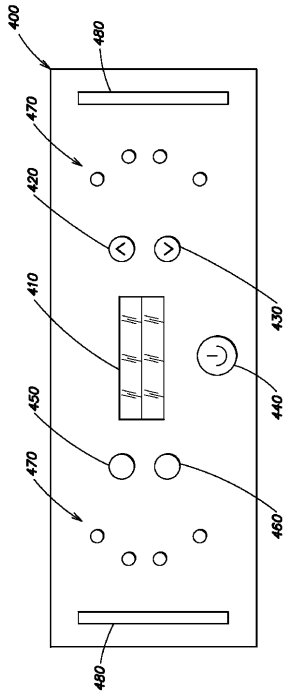
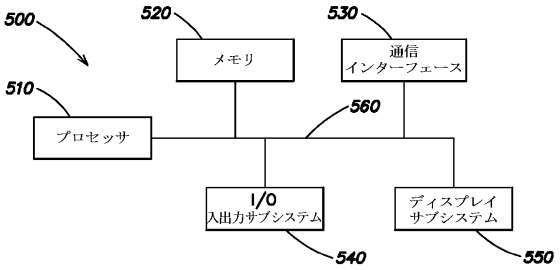

400
480
470
420
410
430
440
450
470
460
480

FIG. 4

【図５】


500
520
メモリ
530
通信
インターフェース
510
プロセッサ
560
I/O
入出力サブシステム
ディスプレイ
サブシステム
540
550

フロントページの続き

(72)発明者　ダオ　イー　ズー
アメリカ合衆国　マサチューセッツ州　０１７２０　アクトン　アップル　ヴァレー　ドライヴ　３

審査官　塚田　肇

(56)参考文献　特表２０１２－５１６５１５（ＪＰ，Ａ）
特開２０００－０２９４４８（ＪＰ，Ａ）
特開平１０－３２２７３９（ＪＰ，Ａ）
特開平４－４８３２６（ＪＰ，Ａ）
特開平４－２５７１２７（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　　９／４４