【公報種別】特許法第１７条の２の規定による補正の掲載
【部門区分】第６部門第３区分
【発行日】平成23年12月22日(2011.12.22)

【公表番号】特表2008-509483(P2008-509483A)
【公表日】平成20年3月27日(2008.3.27)
【年通号数】公開・登録公報2008-012
【出願番号】特願2007-525004(P2007-525004)
【国際特許分類】
Ｇ０６Ｆ　9/445　(2006.01)
Ｇ０６Ｆ　9/44　(2006.01)
【ＦＩ】
Ｇ０６Ｆ　9/06　６５０Ｂ
Ｇ０６Ｆ　9/06　６２０Ｋ

【誤訳訂正書】
【提出日】平成23年10月28日(2011.10.28)
【誤訳訂正１】
【訂正対象書類名】特許請求の範囲
【訂正対象項目名】全文
【訂正方法】変更
【訂正の内容】
【特許請求の範囲】
【請求項１】
ハードウェアデバイスのための固定サイズのファームウェアのイメージを生成する方法であって、
前記ハードウェアデバイスは、前記ハードウェアデバイスに関する複数のコンポーネント環境に対して、ビット数を含む引数の関数としてパラメータ表示され、
前記方法は、
前記複数のコンポーネント環境の複数の態様を、ビットブロックを使用する論理記述として表現することと、
前記複数のコンポーネント環境の複数の物理態様を、ビットブロックを使用する物理記述として表現することと、
前記論理記述と前記物理記述とを一対一マッピングを使用してペアとして関連づけることと、
前記論理記述と前記物理記述との前記ペアを含む記述ブロックを、前記複数のコンポーネント環境の各コンポーネント環境において前記ハードウェアデバイスをサポートする複数のパラメータ表示関数として前記ファームウェアのイメージに追加することと、
前記パラメータ表示関数が追加された前記ファームウェアのイメージを前記ハードウェアデバイスのホストの記憶部にロードすることと
を含む方法。
【請求項２】
前記ハードウェアデバイスは、電源、バス、ファン、ディスクドライブ、センサー、フラッシュ部からなるグループから選択される、請求項１に記載の方法。
【請求項３】
前記パラメータ表示関数に渡される複数の実引数が実行時にバインドされる、請求項１に記載の方法。
【請求項４】
前記パラメータ表示関数に渡される複数の実引数は各々が、コンテクストに依存して解釈される一の固定サイズのビットブロックである、請求項１に記載の方法。

【請求項５】
各ビットブロックが同じ固定サイズである、請求項４に記載の方法。
【請求項６】
各ビットブロックが６４ビット長から２５６６ビット長の範囲にある、請求項５に記載の方法。
【請求項７】
前記ファームウェアのイメージは、
前記ハードウェアデバイスのための抽象デバイスドライバーインターフェイスを含む動作ブロックをさらに含む、請求項１に記載の方法。
【請求項８】
前記パラメータ表示関数に渡される複数の実引数は各々が、実行時に、実行時コンテクスト決定型にキャストされる、請求項４に記載の方法。
【請求項９】
前記型のサイズは、前記ビットブロックの前記固定サイズより小さいか又は等しい、請求項８に記載の方法。
【請求項１０】
前記ホストに接続されたハードウェアを監視するべく、かつ、発見されたハードウェアの変化に応じて前記ファームウェアの複数の態様をアクティベートするべく、動的発見モニターを前記ホストに与えることをさらに含む、請求項１に記載の方法。
【請求項１１】
前記ホストは、特定用途向け集積回路（ＡＳＩＣ）である、請求項１０に記載の方法。
【請求項１２】
前記物理記述は、チップ上の出力ピンの数、その割り当て、特定のデータの記憶場所、多様なクロック及びタイミングパラメータ、温度パラメータのうちの１つ以上を含み得る、請求項１に記載の方法。
【請求項１３】
複数のハードウェアデバイスのための固定サイズのファームウェアのイメージを生成する方法であって、
前記ハードウェアデバイスは各々が、前記ハードウェアデバイスの各々に関する複数のコンポーネント環境に対して、ビット数を含む引数の関数としてパラメータ表示され、
前記方法は、
（Ａ）前記複数のハードウェアデバイスの各々に対して、
（ａ１）前記複数のコンポーネント環境の各々の複数の態様を、ビットブロックを使用する論理記述として表現することと、
（ａ２）前記複数のコンポーネント環境の各々の複数の物理態様を、ビットブロックを使用する物理記述として表現することと、
（ａ３）前記論理記述と前記物理記述とを一対一マッピングを使用してペアとして関連づけることと、
（Ｂ）前記論理記述と前記物理記述との前記ペアを含む少なくとも一つの記述ブロックを、前記複数のコンポーネント環境の各コンポーネント環境において前記ハードウェアデバイスの各々をサポートする複数のパラメータ表示関数として前記ファームウェアのイメージに追加することと、
（Ｃ）前記パラメータ表示関数が追加された前記ファームウェアのイメージを前記ハードウェアデバイスのホストの記憶部にロードすることと
を含む方法。
【請求項１４】
前記ハードウェアデバイスの各々は、電源、バス、ファン、ディスクドライブ、センサー、フラッシュ部を含むグループから選択される、請求項１３に記載の方法。
【請求項１５】
前記パラメータ表示関数に渡される複数の実引数が実行時にバインドされる、請求項１

３に記載の方法。
【請求項１６】
前記パラメータ表示関数に渡される複数の実引数が、型なしビットブロックとして与えられる、請求項１３に記載の方法。
【請求項１７】
各ビットブロックが同じ固定サイズである、請求項１６に記載の方法。
【請求項１８】
各ビットブロックが６４ビット長から２５６ビット長の範囲にある、請求項１７に記載の方法。
【請求項１９】
前記ファームウェアのイメージは、前記ハードウェアデバイスの各々の抽象デバイスドライバーインターフェイスを含む動作ブロックをさらに含む、請求項１３に記載の方法。
【請求項２０】
前記ホストに接続されたハードウェアを監視するべく、かつ、発見されたハードウェアの変化に応じて前記ファームウェアの複数の態様をアクティベート又は変更するべく、動的発見モニターを前記ホストに与えることをさらに含む、請求項１９に記載の方法。
【請求項２１】
前記ホストは、特定用途向け集積回路（ＡＳＩＣ）である、請求項２０に記載の方法。
【誤訳訂正２】
【訂正対象書類名】明細書
【訂正対象項目名】全文
【訂正方法】変更
【訂正の内容】
【発明の詳細な説明】
【発明の名称】予期しない又は変化するハードウェア環境へのソフトウェアとファームウェアの適応
【技術分野】
【０００１】
本発明は一般的にコンピュータシステムに関する。特に、本発明は、予期しない又は変化するハードウェア環境に対応するファームウェアに関する。
【背景技術】
【０００２】
本発明は、「予期しない又は変化するハードウェア環境へのソフトウェアとファームウェアの適応」というタイトルの２００４年８月４日に出願された米国仮出願Ｎｏ．６０／５９９０８８に関し、その優先権を主張するものであり、該出願は参照により本願開示の一部を成す。
【発明を実施するための最良の形態】
【０００３】
本発明は、添付の図を参照し、以下の詳細な説明を読むことによってよく理解することができる。尚、図中では適宜、同じ参照番号と文字を使って異なる図ファセット間で同じか、対応するか、或いは、類似の要素を示している。
【０００４】
以下の詳細な説明では、構成要素の例が挙げられるが、本発明はそれらに限定されることはない。本発明の例示的一実施形態について説明されるが、本発明はその構成で利用されることに限定されることはなく、処理システム内の様々な構成で利用可能である。本願で特にことわりなき場合は、「含む」という用語は排他的なもの、或いは、限定的なものではない。
【０００５】
（背景と概観）
代表的なコンピュータシステムには多数のハードウェアデバイスが含まれており、その

各々を制御する必要がある。それらのデバイスは、通常、マイクロプログラム可能なプロセッサに組み込まれた低レベルの制御プログラム（ファームウェアと呼ばれることがある）によって制御される。本願で使われる「コンピュータシステム」という用語は、少なくとも一つのプロセッサを含むあらゆるシステムを意味する。従って、本発明の一実施形態のコンピュータシステムには、パーソナルコンピュータ、メインフレーム、ＰＤＡ（パーソナルデジタルアシスタント）、関連する周辺デバイス、システムのオペレーションに関連するハードウェアコンポーネントが含まれる。本発明の一実施形態のコンピュータシステムは、家庭用機器、ディスクトップ筐体、或いは、その他のデバイスに組み込むことができる。周辺デバイスには、メモリカード、ディスクなどの外部メモリデバイス；映像・音声入出力デバイス；プリンタなどが含まれる。ハードウェアコンポーネントには、電源、様々な型のバス；ファン、ディスクドライブ、センサー、フラッシュ部品などが含まれる。

【０００６】

ファームウェアは、典型的には、フラッシュメモリなどの不揮発性記憶装置に格納されるが、本発明の実施形態はそれに限定されることはない。その代わりに、ファームウェアを読み出し専用メモリ（ＲＯＭ）、不揮発性ＲＡＭ（ＮＶＲＡＭ）などに格納してもよい。

【０００７】

コンピュータシステムでは、ファームウェアはそれが動作し制御する特定のハードウェアデバイスに高度にカスタマイズされることが多い。従って、ハードウェアが変更されたとき、及び／又は、コンピュータシステムに追加されたときは、通常、ファームウェアを変更する必要がある。本発明の発明者らは、初めて、コンピュータシステムのハードウェアを修正（追加、変更、削除、アップグレード）する毎に新規の或いは更新されたファームウェアをインストールする代わりに、単一のファームウェアプログラムが、予期しない又は変化するハードウェア環境に対処し適応可能となるように開発できることを理解した。

【０００８】

本発明の一態様に係る本質的に一般的なファームウェアを提供するために、第１のハードウェアデバイスが、その全体の機能に基づいて分類され汎化される。有限個の一般型の周辺デバイス（例えば、電源、ファン、ディスクドライブ、センサー、フラッシュ部など）がある。また、デバイスの物理的な態様とそれらの論理的な態様とを区別することが好ましい。この様に、一般的なデバイスの物理記述と論理記述を得ることができる。論理記述とそれに対応する物理記述を使うことによって完全なハードウェアデバイスの記述を得ることができる。

【０００９】

様々なデバイスに対する物理的なデバイス記述と論理的なデバイス記述から、特定の種類のデバイスを汎化するためのサポートを行う共通のファームウェア・プロトタイプが得られる。次に、このプロトタイプをパラメータ表現することによって、それが使われる様々な実デバイスをサポートすることができる。まず、このプロトタイプの実装をせずに（当該型の）仮想の／実在しない周辺デバイスを実装することができる。

【００１０】

認知されている一般型の周辺ハードウェアデバイスは、所定の組み込みソフトウェア／ファームウェア部品の応用範囲を反映したものである。この部品は、例えば、ある形態のハードウェア制御可能ロジックをある種の組み込みマイクロコントローラに実装する部品である。この場合、周辺デバイスは（温度、電圧、電流、シャーシへの侵入（ｃｈａｓｓｉｓ－ｉｎｔｒｕｓｉｏｎ）などのための）センサー、ファン、ＬＥＤ（発光ダイオード）などである傾向がある。無論、本発明は、認知されている特定の複数の周辺デバイスの集合に限定されることはない。

【００１１】

上述したように、周辺ハードウェア環境の完全記述には論理記述と物理記述の双方が含

まれている（物理記述はそれに対応する論理記述のペアとして完全記述となる）。物理記述には、チップの出力ピンの番号と割り当て、所定データのメモリ位置、様々なクロックとタイミングパラメータ、温度パラメータなどが含まれていてもよい。

【００１２】

図１は、本発明の一実施形態のアーキテクチャの概念を示す。本図はコンポーネント・スキームを論理的な部分と物理的な部分に分割した構成を示す。また、本図は、共通のデフォルト／プロトタイプ・インターフェイスを利用し、補完的な物理レベルのスキームを利用することを示す。論理スキームと物理スキームを合わせることによって、コンポーネント・プロトタイプの所定のインスタンスの完全記述が得られる。

【００１３】

説明を簡単にするために、本図では、任意の副次的なコンポーネントのアセンブリから構成される所定のコンポーネントの性能については割愛する。

【００１４】

（電源の例）

本発明の様々な態様について特定の例として電源を挙げて説明する。当業者であれば、本例が本発明の範囲を限定するものではなく、その目的は説明することであることを理解できるであろう。

【００１５】

電源は、全てのコンピュータシステムと全ての周辺デバイスのコンポーネントである。本願で利用する電源の例は、様々な種類の（電気的）「電源」コンポーネントのために利用される一つの所定のデフォルト／プロトタイプスキームによってコンポーネントアセンブリがどのように影響を受けるかについて説明するものである。電源は多くの異なる役割を担っている。その中には自立型インテリジェント管理副次システムを備えるものがあるが、そうでないものもある。電源の中には、内部に複数の電源ユニットをもつ実電源システムがある。電源の種類は驚くほど多い。しかしながら、本発明の実施形態によれば、これらの全ての固有電源は、一般的な論理的な（電気的）電源として扱うことができる。例えば、周辺デバイスの電気的な電源について考えていただきたい。

【００１６】

一般的な方法では、論理的な電源記述と物理的な電源記述の両方を使ってパラメータ表現する。プロトタイプは、各ビヘイビア・ファセット／メソッド／スロットの実装については柔軟である。本願で使う「ファセット」とは、（実行時）動作のビヘイビアのことか、或いは、（それに対応する周辺ハードウェアデバイスの現在の状態に関する）特定の属性／特性／データのことのいずれか一方として定義される。ビヘイビア／メソッド／スロットは代替可能であることが望ましい。本発明の実施形態によれば、一つの論理インスタンスは、（当該周辺デバイスのためのスキームに基づく）副次的な論理インスタンスのグラフから構成可能である。

【００１７】

所定の一般化された型のインターフェイスは予期される特定数のソフトウェア／ファームウェアインターフェイス・ファセットを持つものである必要がある。上述したように、周辺ハードウェアデバイスは、（増々特殊化する様々な型の）ファンからセンサーまでの全てを動作させるものであって、それには、ＬＥＤ、ＬＣＤ（液晶ディスプレイ）、モデム、ＮＩＣ（ネットワークインターフェイスカード）（或いは、個別のＭＡＣ／ＰＨＹチップ）などが含まれる。

【００１８】

特定の電源デバイスの例に戻って、電気的な電源の全ては以下の論理的なファセットをサポート可能であると期待される。

１．パワーの初期化（何も要求されない場合は、ことによると「何もしない」ファセットであり得る）、

２．パワー情報（特定データフォーマットのパワー関連情報）、

３．パワーオン（アクティベート／開始／オンにする）、

４．パワーオフ（デアクティベート／停止／オフにする）、
５．パワーリセット、
６．パワーサイクル（通常、（ａ）パワーオン、（ｂ）一時停止、（ｃ）パワーオフ）、及び
７．パワーステータス（現在のパワーステータス）。

これらの論理ファセットの背後にある実装論理を駆動するための対応する物理記述／実装を、これらの論理ファセットの実装を最終的に物理的な電気信号送信インターフェイスにどのようにバインドさせるかに関する記述について以下で説明する。本例の目的のために、期待される以下の電源（副次システム）ファセットの選択リストに焦点を当てる。

【００１９】

本例の説明を簡単にするために、特定の及び／又は固有の型の電源が備える（或いは、備えていない）更なる／追加的インターフェイス・ファセット（即ち、特徴／機能）については説明しない。この制約によって、一連の電源の周辺デバイスをこの特定のスキームでモデル化することが可能になる。重要なことは、本発明によってあらゆるより固有の型の電源モデルが不必要になることである。

【００２０】

一般的に、本発明によれば、関連する周辺（及び／又はホスト）デバイスを、組み込みソフトウェア／ファームウェア部品の枠組みの中で（論理的にかつ物理的に）説明する必要がある。関連する電源（周辺デバイス）が存在する場合は、組み込みソフトウェア／ファームウェア部品では所定の論理的な電源記述を配置する必要がある。全ての（電気的な）電源を説明するために使われる特定の論理スキームの一例を以下に示す。

【００２１】

・論理パワーオン信号（パワーをオンにする）
・論理パワーオン信号幅（即ち、持続時間）
・論理パワーオフ信号（パワーをオフにする）
・論理パワーオフ信号幅（即ち、持続時間）
・論理パワーリセット信号（パワーのリセット）
・論理パワーリセット信号幅（即ち、持続時間）
・論理パワーステータス（現在のパワーステータス－真／偽状態を読む）
・論理割り込みリクエスト（ＩＲＱ）型（外部ボタン押下事象に対して）
・論理割り込みリクエスト番号（ＩＲＱ識別子）
・論理割り込みリクエストの解釈（アクティベート、デアクティベート、或いは、その両方）
・＜ビヘイビアの論理記述子＞（例えば、関数／メソッドの署名）

列挙した記述属性の各々によって＜ビヘイビアの論理記述子＞がパラメータ表現される。これらの列挙した記述属性の各々は所定のビットブロック（ｂｌｏｃｋ　ｏｆ　ｂｉｔｓ）（「ＢｏＢ」－以下で詳細に説明される）として表される（パックされる）。＜ビヘイビアの論理記述子＞の一実施形態を図２に示す。

【００２２】

本願で使われる各論理記述の構造規約（ｓｔｒｕｃｔｕｒａｌ　ｃｏｎｖｅｎｔｉｏｎｓ）は「スキーム（ｓｃｈｅｍｅ）」（単数）と呼ばれる。「スキーマ（ｓｃｈｅｍａ）」（複数）と「スキーマタ（ｓｃｈｅｍａｔａ）」（複数）という用語は、スキーマタ全体を参照するために使われる。（そのため、本例では電源スキームを説明する）本例の電源スキームとその他の周辺デバイスのために使われるスキーマタをまとめることによって、より包括的な周辺デバイス・スキーマを構成（或いは、ビルドアップ）することが可能になる。同様に、周辺デバイス・スキーマとホスト処理サポートコンポーネント・スキームをまとめると、単数で包括的な組み込みシステムスキーマになると考えられる。

【００２３】

（めったに発生しないが）論理記述ファセットのうちの一つが固有型のスキーマによってよくサポートされていない場合は、適当な無害の実装によってファセットをスタブ（除

去）することができる。典型的なスタブの処理は、常に実装として成功する（ｓｕｃｃｅｅｄｓ）（スタブは、役に立つデフォルトの実装である）。本発明の実施形態の様々なスキーマタによって、幾つかのファセットを必須なものとし、その他のものをオプションとすることができる。さらに、デフォルトのスタブは、オプションとしてのファセットのほとんどに対して提供される。所定の記述によってそれらが省略される場合はデフォルトが適用される。多くの場合、この技術によって、本発明が非コンパチブルな（特定の）型の電源をサポートすることが可能になる。スタブを賢明に使うことによって、様々な電源をスキームの期待値に合致させることができる。

【００２４】

（実行可能な固定のイメージと論理スキーム）

汎化されたファセットスキームに基づいて全周辺デバイスを説明する。論理型の周辺デバイスの各々は、これらの汎化されたファセット・テンプレート（本願ではスキームと呼ばれる）のうちの一つに一致する。この技術はいかなる背景でも利用可能であり、利用されるスキーマタの型を大きく変えることができ、それに対応して最適化されたスキーマタを集めることによって、各アプリケーションのコンテクストに最もよく対応することができる。例えば、電圧センサーが一つもないデバイス／製品の電圧センサーをサポートする理由はない。

【００２５】

その結果、所定の（アプリケーション）コンテクストに適用可能な（及び／又は、期待される）型のハードウェア周辺デバイスだけを記述するための準備として、限定されたスキーマを前もって定義しておくことができる。さらに別のスキーマタを備えることを選択して、（結果としてファームウェア／ソフトウェア「部品」の）適用可能性を上げることができる。選択されたスキーマの表現範囲は、それに対応する組み込みソフトウェア／ファームウェア／ホスト部品に対して期待される範囲の適用可能性にマッピングされる。

【００２６】

また、所定の最大個数の論理周辺デバイス記述インスタンスが利用される。選択された最大個数は任意の値であり、本発明は、選択した数の記述を利用することができる。実際上、これは、異なる組み込みソフトウェア／ファームウェア部品で既知の不揮発性（ＮＶ）記憶の容量を任意でかつ先験的に確保できることを意味する。この選択された一定の容量は、（所定のアプリケーションのコンテクストで）必要とされる論理記述（及び／又は物理記述）インスタンスの最大個数に対応する傾向がある。この最大個値は、ほとんどの、或いは全ての物理的に実現可能な周辺デバイス構成をカバーできる程十分大きな値であることが好ましい。本発明の実施形態の記憶サイズを固定にすることが望ましいが、記憶装置の動的割り当ても可能である。従って、組み込みファームウェア／ソフトウェア部品は、一定サイズであってもよいし、可変であってもよい。

【００２７】

当業者であれば理解できることであるが、利用可能な記述インスタンスの最大個数は、所定のファームウェア／ソフトウェア部品が組み込まれる製品に関するコスト関数であることが多い。さらに別の不揮発性記憶チップでは、（組み込みソフトウェア／ファームウェア部品の）動作するホストのハードウェア環境を拡張することができるが、そのチップはコストを増大させる。

【００２８】

本発明は、動的に割り当てられた任意の記憶容量を使って機能するが、当業者であれば、任意であるが固定個数の記述エントリが必要なときに本発明が機能することを理解することができる。本発明は、厳密に上限が定められた不揮発性記憶容量とともに機能する。記述インスタンスの最大個数が選択された場合、組み込みファームウェア／ソフトウェアの実行可能なイメージのレイアウトは変わらない。従って、この記述エントリの最大個数を選択するためのビルド時のオプションは本発明の実施形態の一部を構成する。

【００２９】

（自由形式記述のバインド）

本発明の実施形態では、組み込みソフトウェア／ファームウェア・コードが所定の周辺コンポーネントに対する適切な論理インターフェイスにバインドできるように、所謂「関数ポインタ」を使った実行時バインディングを利用する。一般的に次にくるものは、ビルド時のコンパイラの呼出し規約（ｃａｌｌｉｎｇ　ｃｏｎｖｅｎｔｉｏｎｓ）である。これによって、論理的なハードウェアコンポーネント記述のそれぞれによって、ハンドル（例えば、メモリアドレス）をデバイス型／スキーマ特定的なデータ構造に提供することになる。また、このデータ構造は、様々なデバイス記述に固有のビヘイビア（例えば、関数ポインタ）に対応する。

【００３０】

本発明の実施形態の論理インターフェイスがパラメータ表現される。これによって、電源などの周辺ハードウェアコンポーネントの論理的なカテゴリを汎化することが可能になり、これらの様々なデバイスの全てと通信する様々な多くのメソッドが利用可能になる。

【００３１】

本発明の好適な実施形態では、我々の論理的なビヘイビア・ファセットの各々（と全て）に対する引数／パラメータとして多数の曖昧なビットが引き渡される。発明者らは、ほとんどの引数／パラメータが６４ビットから２５６ビットまでの間のビット数で満足することが多いことを見出した（高密度にパック化ビットフィールドを積極的に使う場合）。無論、当業者であれば、正確なビット数が当該システムスキーマに基づくものであることを理解することができる。

【００３２】

このパラメータ引渡しのアプローチによって、本発明は可変個と可変型のビヘイビア（例えば、関数／メソッド）とパラメータ（例えば、関数／メソッドの引数）の両方を扱うことができる。電源のような非常に一般的な概念を説明するには、様々な個数のパラメータと様々な型のパラメータが電源の特定の副次型のために必要となる場合が多い。本発明では、再符号化／再プログラム化ではなく、プロトタイプに基づくアプローチを採用する。全てのプロトタイプ・ビヘイビアでは、交換可能な、及び／又は、標準のインターフェイスをもつ必要がある。

【００３３】

システムに変化が起こっても、その組み込みファームウェア／ソフトウェア部品は再ビルド（即ち、再コンパイル／解釈）が不要であることが望ましいことを思い出していただきたい。これらは可変個のインターフェイス引数を使って機能する必要がある。

【００３４】

一例として、特定の実装でのビットブロックが６４ビットであるとする。通常、これらの６４ビットは、（多くの場合、マシン語か、或いは、デュアルマシン語として）（実行時での）各論理インターフェイスの呼び出しに関連する関数コール（スタックフレーム）で強制されるものである。これらの６４ビットは異なる数のパラメータとして、及び／又は、異なる型のパラメータとして解釈される。それらを正確に解釈するメソッドは、当該論理スキームに基づくものである。これらの曖昧なビットの一部は、当該スキーマの特定の物理記述に基づいて解釈されることがある。図２は、本発明の実施形態に基づいてこれらのビットブロックをサポート可能な方法の一例を示す。当業者であれば理解できることであるが、本発明を実施するための多くの方法が存在する。異なるプログラミング言語の実行時のファシリティは、異なる実装にとって好ましい傾向がある。図２は、Ｃプログラミング言語を使って、（できる限り固定サイズの）動作ブロック内で共通のプロトタイプを実装する方法を示す。

【００３５】

マイクロプロセッサの中には６４ビットのマシン語を直接サポートするものもあるが、利用される曖昧ビットの正確な数は任意である。これは、本発明を適用するシステムスキーマに依存する。このアプローチは、いかなるビット数でも同じく機能する。必要に応じて、（あらゆる特定の組み込みソフトウェア／ファームウェア部品用に使われるスキーマでは全て記述されたファセットに）引き渡される曖昧ビットの数を拡張することができる

。当業者であれば理解できることであるが、ビットブロックの型を決めないことは好適である。

【００３６】

本発明の好適な実施形態では、利用されるビット数とは無関係に、その数は所定の部品に対して一定の値であるべきである。本発明の実施形態で好ましくないことは、可変数のビットを使うことであり、これは、システムの複雑度を増す。ビットブロック内で曖昧なビットの数が固定である場合、ビヘイビア・ファセットの各々は、常に単一の標準ビットブロックの引数を利用することができる。これは、ビヘイビア・ファセット（或いは、スロット）を実装するためのユニークなメカニズムとメソッドである。これによってある程度、記述ファセットを通常の関数や通常のオブジェクトメソッドなどから区別することができる。

【００３７】

電源の例に戻り、以下の表（表１）は、記述した電源の（論理的な）概念を（曖昧な）ビットブロックにマッピングした一例を示す。

【表１】

表1：論理電源の説明（ビットブロックのみ）

セクション／ビットオフセット	ビット数	用途	一例の値
1	16	パック化ビットフィールド	0x0404
15:8	8	論理パワーオン（GPIO）信号	0x04
7:0	8	論理パワーオフ（GPIO）信号	0x04
2	16	パック化ビットフィールド	0x0605
15:8	8	論理パワーステータス（GPIO）信号	0x06
7:0	8	論理パワーリセット（GPIO）信号	0x05
3	32	パック化ビットフィールド	0x053205FF
31:24	8	パワーオン信号パルス幅	0x05
23:16	8	パワーオフ信号パルス幅	0x32
15:8	8	パワーリセットパルス幅	0x05
7:7	1	パワーステータスの信号の代わりにIRQを使う	1b
6:6	1	外部（真）或いは内部（偽）IRQ型	1b
5:2	1	割り込みリクエスト（IRQ）番号	0x4
1:0	2	トリガエッジ（0＝フロント、1＝バック、2＝両方）	10b

【００３８】

（柔軟な記述のファセット）

既存の実装を最適化するために、多数の技術を利用して可変の個数で可変の型の引数／パラメータを、（高密度に／効率的に）ビットブロックにパックされたビットフィールドシーケンスに圧縮することができる。ユニークなインターフェイス・ファセット署名を完全に標準化するために、本発明の実施形態の中には、（「ステータス」値と呼ばれる）値を戻すインターフェイス・ファセットをユニバーサルに列挙することを採用するものがある。さらに、本発明の実施形態のシステムでは、ｃｏｎｆｏｒｍａｎｔ関数を使って、状態／属性／特性のアクセス・ファセット（例えば、所謂「ゲッター（ｇｅｔｔｅｒ）」及

び「セッター（ｓｅｔｔｅｒ）」）とビヘイビア・ファセットとの両方を実装することが好ましい。

【００３９】

これまで論理記述のアプローチ（特定のスキーマ規約に一致する特定のスキームを使った）について説明した。これは、様々なプロトタイプのインスタンスをサポートするものである。各インスタンスは一つの周辺ハードウェアデバイスを表すことができる。

【００４０】

この新規なメカニズム／メソッドはどのようなやり方にも適用可能であるが、この電源の例では、非常に多くの数の電源型の表現をいかにサポートするかを示す。その電源型の論理インターフェイスは、曖昧なビットブロックを使って、例えば、単一の、統一され、一貫したメソッドでパラメータ表現される。本発明によって、固定のソフトウェア／ファームウェア・イメージで、スキーマ・インスタンスに関連づけられたあらゆるインターフェイス・ファセットを使うことが可能になる。様々なスキーマによって、（周辺デバイス・スキーマ内の）非常に広範な周辺デバイスを表すことができる。これらのメカニズムとそれに対応するメソッドは、記述されたファセットに一つの実装をバインドするための一貫性のある一定のメソッドを備える。

【００４１】

（宣言的な記述ファセットの署名）

これまで、ビットブロックを曖昧なものとして説明した。ここで、電源の例を使って、本発明の他のメカニズムとメソッドがどのように実世界の電源の特異性をサポートするかについて示す。

【００４２】

ビットブロックの引数／パラメータは、（電源の）所定の論理／論理スキームの規定された／暗示された要求／意図に一致する特定のファセット群に関連づけられる。まず、電源スキームで必要とされる論理ファセットに焦点を当てる。この論理記述は、電源が何らかのメソッドでサポートする必要のある電源ファセット／インターフェイスの概要を示すものである。これは、ビットブロックをどのように解釈するかを決定することに関連する電源スキームである。電源記述インスタンスを扱っていることがわかれば、ビットブロックを解釈することができる。ビットブロックは、（所定のビットブロック・インスタンスの）電源のコンテクストを知る論理にとっては曖昧ではない。

【００４３】

一般的／特定的な型の階層にわたり、ビットブロックの中で様々にネストしたセクションを解釈することができる。この例を単純化するためによくあることであるが、（電源のような）特殊な（リーフ（ｌｅａｆ））型の各々は、それ自体のビットブロックの全体を扱えるものとする。例えば、周辺デバイス（型の種類）のより一般的な概念では、ビットブロックの解釈に対してさらに別の規約を課すことはない。

【００４４】

例えば、電源型は、それ自体の特殊なビットブロック（（上の）表１で詳述された）を宣言してこれを適用する。このビットブロックで使われる実値は、様々な特定の一般的／汎用電源型の間に存在する多くの違いを記述したものである。仮想的な電源のサンプル値を表１に示した。

【００４５】

電源デバイスは、パワー制御パルス幅の概念（その他の種類の周辺デバイスではありそうもない概念）を備えることができる。パワーオン／オフ制御は全く異質の概念であり、その他の（周辺）デバイス型にはほとんど適応できない。例えば、（一般的に）センサー或いは（特に）温度センサーには、一般的に「パワーオン／オフ制御」の概念がない。一般的には、パワーオン／オフ・ビヘイビアも、関連する電気信号パルス幅も（周辺）デバイスには与えられない。さらに、特定の電源副次型では、異なるパワーオン／オフの電気パルス幅が必要である。

【００４６】

また、電源インスタンスは、固有電源のインターフェイスに対して選択可能な異なる固有配線を要求することができる。これは、当該電源インスタンスを所定のシステム／製品内にどのように配置するかに依存する。システム／製品のレイアウトのやり方は、あらゆる電源オフ／オフ信号のためにどのホストの配線／ピンを（どのような幅又は持続時間であろうと）使う必要があるかに影響する。どの配線を信号（パルス）のために使わなければならないのかは、別の物理レベルの問題であり、物理スキームで記述される。

【００４７】

（論理記述は新しい型のシステムを実装する）

多くのオブジェクト指向システムでは、特定型電源の各々を表すクラスが存在する。各クラスインスタンスはパワーオン／オフのパルス幅のメンバー（特性／属性）を承継するが、各クラスでは、所定の電源のために使える様々な配線／インターフェイスを扱う異なるメソッド論理が必要となる。本発明ではこれが不要である。本発明がなければ、急速に増える別個の電源型／クラスが必要になる（ことが多い）。

【００４８】

その他のソフトウェアシステム（例えば、オブジェクト指向プログラミング言語）のほとんどは、クラス／型の定義に基づいて一般化されるものである。この型に基づく一般化は、クラス・メンバーのタイプとクラス・メソッドの署名の両方から影響を受けやすい。クラスのメソッドの署名には、メソッドの戻り値の型、メソッドの引数の個数、各メソッドの引数の型が含まれる。本発明を使って、宣言的に作成された（ｔａｉｌｏｒｅｄ）プロトタイプは、記述されたファセットを実装するものである。ビットブロックのような技術は、全てのファセットの署名を交換可能にするので、副次型化する必要がなくなる。

【００４９】

それは副次型固有のビットブロックを解釈するだけである（即ち、各副次型がそれ自体のビットブロック・フォーマットを普及させる（ｐｒｏｍｕｌｇａｔｅｓ）ものである）。これによって、異なる引数の個数と異なる引数の型（にもかかわらず）で機能する一貫したインターフェイスが可能になる。従って、本発明の実施形態のプロトタイプ・スキームは、（周辺）デバイス・スキーマを構成することができる。これは、さらに柔軟な処理／プログラミングの汎化メカニズムを提供するものである。

【００５０】

このさらに柔軟な処理／プログラミングの汎化によって、あらゆる数の特定の周辺デバイスを同時にサポートする処理を条件付で組み込み可能な固定のソフトウェア／ファームウェア・イメージの生成が可能になる。新しいデバイス固有のコードをロードする（或いはリンクする）必要はない。本発明を利用するシステムは、いかなる種類の再ブート、リスタート、或いは、再初期化をも必要としない。実行時に（動的ハードウェア環境に対して）固定の部品を適応させることは実際上役に立つことである。宣言された（或いは、追加された）データが単純に変更されても（一般的に、周辺デバイスの）スキーマで十分に対応可能である。別の観点では、記述スキームのインスタンスの集まり（ｃｏｌｌｅｃｔｉｏｎ）（即ち、本発明の実施形態のスキーマ）によって、周辺デバイスのサポートを完全にデータ主導とすることができる。

【００５１】

動作時の一例では、１００ミリ秒のパワーオン／オフパルス幅は共通デフォルト値である。特定の電源型では、より長いか或いは短いパルス幅が必要とされることがある。本発明は、そのような全ての差異に対応することができる。

【００５２】

本発明の実施形態によれば、単一の固定のソフトウェア／ファームウェア・イメージは、（ブランド、製造元、モデルなどの）非常に態様名固有の型の実装を含むことができる。固定の論理の中には、（周辺）デバイスなどきわめて一般的なデバイスのインスタンスを操作できるものがある。電源バデイスのような一般的でないバデイスのインスタンスを操作できるものもある。他方、我々の一定のイメージ内にあるその他の論理では、例えば、非常に特殊なモデル／デバイス（例えば、「ＦＯＯモデル２５００サーバコンピュータ

のＡＣＭＥ電源モデル１２Ａ」）のような特定のタイプを扱うことができる。
【００５３】
超汎化されたスキーマタ（例えば、電源）の性質のため、本発明はユニークなレベルの処理／プログラムの拡張性を提供する。新しいタイプをより簡単でよりパワフルに追加することができる。さらに、スキーマ規約（即ち、インターフェイスの取り決め事）はより非制限的である。個々のプロトタイプスキームはより一般的なものであるため、新しいタイプでは既存の論理に適合して協調動作することができる。既存の論理は（一般的なスキーマの態様（ｓｈａｐｅ）によって広がった）非常にゆるいこれらのプロトタイプスキームのインターフェイスに対してだけには敏感である。本発明は、先例のないレベルの拡張性（即ち、カスタマイズの可能性）を提供する。
【００５４】
（論理記述アーチファクト：「論理記述表」）
論理記述表のようなものは、（例えば、電源）あらゆるスキームタイプの各インスタンス用の固定サイズのエントリを備えることが多い。論理記述インスタンス／エントリは、スキーム・インスタンス（例えば、周辺デバイス）を表す。その固定のソフトウェア／ファームウェア・イメージは、リストされた各スキームのインスタンス（例えば、各電源スキーム・インスタンス）のファセット・インターフェイスを利用する。（電源のような）所定のスキームタイプのインスタンス間には具体的な（そして、ほとんど任意である）差異があってもよい。従って、記述されたファセットの機械的な部分（ｄｅｓｃｒｉｂｅｄ　ｆａｃｅｔ　ｍａｃｈｉｎｅｒｙ）は非常に間接的で曖昧である。異なる特定の／別個の電源インスタンスをインターフェイス・ファセットのセットに関連づけることができる。通常のオブジェクト指向プログラミングシステムでは、その異なるインスタンスは異なる電源副次型を必要とする。本発明の実施形態のプロトタイプに基づく記述規約では、固有のファセットスキームを完全に異なるコードにバインドすることができる。実際には、副次型を必要とせずに、それらを副次型の特定の実装論理にバインドすることができる。本発明の実施形態のプロトタイプスキームは、唯一の共通スレッドである。
【００５５】
この共通スレッドは、ビットブロックのようなメカニズムを使って柔軟に維持される。スキーマ内の全てのファセット署名では、任意のユニークなビットブロックが共有される。ビット数はスキーマ全体に対して（とスキーマ内の各スキームに対して）固定である。一方、これらの予約ビットの利用／解釈はスキーム固有のものである。
【００５６】
これらのビットの解釈は、当該プロトタイプスキームの種類に依存する。実装する者は、新しいプロトタイプ・ファセットスキームをいつ宣言／定義するか（即ち、新しいスキーム・インスタンスをいつ追加／宣言するのか）を選択する。従って、実装する者は、区別されたファセットスキームを介するか（即ち、別個のスキームをスキーマに導入することを選択するか）、もしくは別個のパラメータ／特性値を介するか（即ち、既存のスキーム・インスタンスの論理記述値を変えることによって）のいずれか一方でデバイスの差異をモデル化することができる。
【００５７】
（物理的なスキーマ）
ここまで周辺ハードウェア環境の宣言的論理記述の利用について説明してきた。それに対応する（周辺ハードウェア環境の）物理記述も必要とされる。物理記述はそれに対応する論理記述とペアをなして完全な記述となる。
【００５８】
動作中の電源の例に戻って、特定の電源タイプが使われているものとする。必要とされることが多い様々な物理記述を例示するために、この電源は、（所定のホストチップ／ハードウェアで利用可能な）所定の汎用入力／出力ピンを介して制御されるものとする。さらに、常にオン状態の組み込み制御可能副次システム専用の第２の電源があるものとする。代替的に、第３の電源インスタンスは、割り込み不可の電源（例えば、バッテリ駆動）

でもよい。
【００５９】
この第２の電源インスタンスは、第１のもののようにホストの（オペレーティング）プラットフォームに直接配線されてもよいが、それとは全く異なるものでもよい。論理レベルでは、第２の電源はシステム（ボード）割り込みを全く扱わなくてもよい。フロントパネルのボタン押下事象の知識がなくてもよく、異なる配線／ピンを使ってホストの／動作上のプラットフォームに接続されていてもよく、また、異なる電気信号パルス幅（及び／又は、トリガ規約）を使ってもよい。
【００６０】
本発明の実施形態の固定のソフトウェア／ファームウェア・イメージは、それが所定のスキーム内でその他の差異を扱うことと同様にこれらの物理的な構成の差異を容易に扱うことができる。（上で議論した）スキームの差異と同じように、本発明では、これらの物理的な／配線レベルの差異のためのソフトウェア／ファームウェア・イメージを変える必要がない。本発明は、ソフトウェア／ファームウェア・イメージを変更することなく、変化するハードウェア環境（例えば、ホットスワップ）を処理することができる。
【００６１】
上述した電源の物理態様のうちの幾つかを扱うために、本発明では、論理周辺デバイス（例えば、各電源）の各インスタンスを、動作上の／ホストのハードウェア（例えば、マイクロコントローラコア・ベースの特定用途向け集積回路ＡＳＩＣ）上の（或いは、それを通してアクセスされる）様々な特徴に関連づける。電源スキームのような周辺スキームは、ポート或いはピン（或いは、ハードウェア内の物理的に別個のビットなど）との関連づけを必要するファセットを備えていてもよい。ファンのスキームのようなその他の周辺スキームでは、（ファン速度を読むための）ファンのタコメータ回路と（ファン速度制御に使われるパルス幅変調回路のような）ハードウェアのその他のビットと関連づける必要がある。電源と対照的に、ファンは、動作上の／ホストのポートとは関連がなく、動作上の／ホストのピンとも関連しない。
【００６２】
本発明は、コンテクスト依存の詳細を（システム）スキーマ全体に渡って一貫したものにするために、コンテクストに依存する関連内容（スキーム間の依存性／委譲関係などを（上述した）ビットブロックを介してサポートする。上述したように、このビットブロックは、当該特定のスキーム（例えば、電源）によって変更可能な解釈（即ち、フォーマット）を備える。この特定の（プロトタイプ）スキームは、そのインターフェイス・ファセットに影響を与える。従って、それは、このスキームコンテクストでビットブロックの解釈とはなんであるかを指示するものである。
【００６３】
論理が現在のスキームを知らない場合は、ビットブロックは全く曖昧なものである。組み込みファームウェア／ソフトウェア・イメージ内で見つかったコードのほとんどは、その他のビットの集合と同じようにあらゆるビットブロックを操作することができる。ファセット・実装論理のようなスキーム固有の論理だけが、スキーム固有の知識を組み込むことができる。スキーム固有の論理だけがビットブロックを正しく解釈することができる。このスキーム固有の情報／知識を隠蔽すること（ｈｉｄｉｎｇ）は重要なことである。それは、（プロトタイプ）スキーム内の各（プロトタイプ）スキームに関する強いモジュール性／カプセル化の境界を与えるものである。
【００６４】
また、ビットブロック値はスキームによって変わることがある。この種のスキーム固有の値は反映値と呼ばれる。これは、所定のスキームの全インスタンスに共通の情報を反映、及び／又は、表すものである。各ビットブロックの一部は、包括的かつさらに一般的なスキームの副次スキームの全てに対して一貫している必要がある。この種の再帰的にネストされた副次スキーム（プロトタイプ階層）は本発明の一部であると考えられる。ビットブロック値は、特定のスキーム・インスタンス毎に異なることが多い。これらは、単純に

当該スキーム・インスタンスを最もよく記述する特定の特性／属性値を反映する。

【００６５】

また動作中の電源の例に戻って、電源の多くは、例えば、それらが使う動作上の／ホストのポート或いはピンを識別する所定のビットを使うことができる。この例を簡単にするために、本発明の範囲に係る制約がない状態で、動作上の／ホストのＡＳＩＣピンを介した通信を行う電源スキーム・インスタンスに焦点を当てる。そのピンは、所謂汎用入力／出力（ＧＰＩＯ）信号を搬送するものである。（論理）ＧＰＩＯ信号のために使われる副次スキームでは、従来の１ビットのＧＰＩＯピンと（バスをそのままの形態として）複数のＧＰＩＯピンを使って、論理ＧＰＩＯ信号を送るケースを記述することができる。そのＧＰＩＯピンの集合は、所謂ビットバングド・バス（ｂｉｔ－ｂａｎｇｅｄ　ｂｕｓ）を表すことができる。この例を簡単にするために、ＧＰＩＯバス・フラグを偽（ｆａｌｓｅ）に設定する。言い換えれば、その例の電源（インスタンス）は、単純な（論理）ＧＰＩＯ信号を利用することができる。これは、ＧＰＩＯバスに関するビットブロックパラメータの全てを無意味なものにするものである（少なくともこの特定の例の電源について）。しかしながら、本発明を適用することによって得られた一つの特定のスキームを例示するために、ＧＰＩＯバスに関するパラメータを（以下の表２に）示す。

【表２】

表2：汎用ＩＱ（ＧＰＩＱ）信号＃４の論理記述（ビットブロックのみ）

セクション／ビットオフセット	ビット数	用途	一例の値
1	16	パック化ビットフィールド	0x0060
15:8	8	予約(現在未使用)	0x00
7:0	8	(論理)GPIOピン番号	0x60
2	16	専用ビットフィールド	0x0060
15:0	16	GPIO信号マスク	0x0060
3	32	パック化ビットフィールド	0x053205FF
31:30	2	GPIO信号タイプ	2
29:29	1	GPIOバスフラグ(1＝真、0＝偽)	0
28:25	4	GPIOバスビット(ビットバングド・バスのビット数)	0x1
24:17	8	GPIOバスポート番号	0x00
16:9	8	GPIOバスビット位置(レジスタ＋ピン)	0x00
8:4	5	GPIOバスビット開始オフセット	00000b
3:3	1	方向(0＝入力／リード、1＝出力／ドライブ)	1b
2:2	1	負論理(真／偽が逆)	1b
1:1	1	初期値(真／偽、論理に与えられる)	1b
0:0	1	オープンドレイン(真／偽、論理に与えられる)	1b

【００６６】

表１は、所定の電源を記述したビットブロックパラメータを示す。この電源では多数の論理入力／出力信号を使用した。これらは、汎用入力／出力（ＧＰＩＯ）信号と呼ばれる。表２は、これらの論理ＧＰＩＯ信号とＧＰＩＯ信号＃４のうちの一つがどのように記述されるかを示すものである。表２には、（純粋に論理的なＧＰＩＯ信号とは相容れない論

理ＧＰＩＯピンへのリファレンスが含まれている。
【００６７】
電源の論理記述（スキーム）は、ＧＰＩＯ信号のための一つ以上の論理記述（スキーム・インスタンス）に依存する。本例では多数のスキーム間の依存性／委譲関係について記述している。ＧＰＩＯピンの物理的な態様は表２で示されるビットブロック内に記述されていることを理解されたい。これらの物理的なＧＰＩＯピンの態様は、（表２で示されるビットブロック内の）論理ＧＰＩＯ信号記述とインターリーブされる。例えば、負論理の初期値とオープンドレイン属性はまさにＧＰＩＯピンの物理属性である（そして、論理ＧＰＩＯ信号の論理態様ではない）。
【００６８】
（様々な理由から）この種の論理的／物理的なインターリーブがなされるが、これは本発明の論理記述対物理記述のペアの性質を変えるものではない。また、その２つの概念的な区別が変わることもない。例えば、表２で（０ｘ６０として）示される論理ＧＰＩＯピン番号は、（本実施形態ではたまたま７８である）物理的なＡＳＩＣピン番号とマッピングされ／関連づけられる。論理ＧＰＩＯ－信号記述と物理ＧＰＩＯ－ピン記述間には１対１の同一性マッピングがある。様々なＧＰＩＯピン／信号の属性がどこに現れるかは、１対１の（同一性）マッピング全体にわたって無関係である。この記述の態様のインターリーブは厳格な論理－物理の区別と同形である。
【００６９】
その結果、その他の因子は、（様々な記述の態様）のインターリーブか、もしくは、（様々な記述の態様を）区別することのどちらの実装を選択するかに対して強く影響する。例えば、異なる電源の型／モデルは、異なるＧＰＩＯ信号パルス幅を必要とする可能性がある。電気信号パルス幅は確かに物理的な態様であるが、便宜上これを、別個の論理的な電源記述と関連づけることができる。そこからそれを獲得して、実際に使われる物理記述とする必要がある。そのピンに対して単一の静的な一電気信号パルス幅を設定したくない場合もある。まれであるが、異なる論理ＧＰＩＯ信号はパルス幅だけで区別されることもある。
【００７０】
本発明はそのような工夫の全てを含むが、基本的な結果は維持されている。即ち、（電源のような）所定の周辺デバイスの完全な宣言的な記述を形成する物理記述と論理記述のペアである。
【００７１】
当業者であればすぐにわかることであるが、ファンなどの他の種類の周辺デバイスのスキームは全て類似の方式で機能する。例えば、ファンは、どの論理的なＧＰＩＯ信号が、（所定のファン・インスタンスに対して）論理パルス幅変調（ＰＷＭ）回路が使うタコメータ信号を提供するのかを識別するために、ビットブロック内の所定のビットを使うことができる。電源の例と同様に、論理ＧＰＩＯ信号は、物理ＧＰＩＯピンに（或いは、ビットバングド・バスと）マッピングされる必要がある。
【００７２】
結局、論理スキームの各インスタンスに適切な物理記述は論理記述値によって明快に参照される。（論理的に記述された所定の周辺インスタンスのために）利用される物理記述がビットブロックから決定されることが多い。
【００７３】
（ホストハードウェア環境）
ハードウェア環境を構成する周辺デバイスを宣言的に記述する本発明のアプローチを使う本発明のいくつかの態様は、ホストのハードウェア環境を宣言的に記述するために提供される。例えば、多くの組み込みファームウェア／ソフトウェアホストは、基本的に多くの特定用途回路が組み込まれるＡＳＩＣである。これらの集積回路は、例えば、リアルタイムクロック、乱数生成器、ＵＡＲＴ（ユニバーサル非同期レシーバ・トランスミッタ）、或いは、その他の多くの関数を実装することが可能である。これらの集積副次回路の各

々を周辺デバイスの特定の副次型として扱うことができる。このケースでは、これらの回路があたかも別の周辺スキームであるかのように、本発明の実施形態の論理記述と物理記述が適用される。

【００７４】

周辺デバイスと同様に、（組み込みソフトウェア／ファームウェアのための）特定のホストのハードウェア環境にＡＳＩＣ回路が欠如していることは、それに対応する周辺デバイスが欠如していることと同じことである。同じ種類の（宣言的な）記述技術は、（ホストのＡＳＩＣ回路か或いは等価の周辺デバイスのいずれか一方に対する）その欠如を示すことができる。

【００７５】

実際は、論理スキームは不揮発性記憶部などに格納される。上述したように、これらの論理記述はそれに対応する物理記述とマッピングされる。例えば、一つの物理記述で、ホストＡＳＩＣに不揮発性記憶部を配置することができ、その他の物理記述で、チップ外のフラッシュメモリ部に不揮発性記憶部を配置することができる。いずれも論理的には単に不揮発性記憶部である。そのような論理的な不揮発性記憶コンポーネントの全てが所定のファセットを共有する。本発明の実施形態のソフトウェア／ファームウェアが組み込まれた一定のイメージによって、同じ論理的な不揮発性記憶部のファセット（インターフェイス）を介して不揮発性記憶部の各インスタンスを操作することができる。ソフトウェア／ファームウェアが組み込まれた一定のイメージの一部は、物理記述の詳細によってパラメータ表現される。その部分は、オンチップの不揮発性記憶部とチップ外の不揮発性記憶部間の差異を隠蔽することができる。また、電源の一事例としてのパタンが例示されたように、チップ外の不揮発性記憶ハードウェア（即ち、所謂「フラッシュ」部）に存在する無数の差異を宣言的に記述することができる。

【００７６】

ほとんどの組み込みファームウェア／ソフトウェア部品の論理はプロトタイプスキーム・ファセットだけに依存するが、論理記述の詳細／値や物理記述には依存しない。同様に、組み込みファームウェア／ソフトウェア部品論理のほとんどは、物理記述の詳細／値のどれにも依存しない。プロトタイプスキーム・ファセットだけは見ることができる。組み込みソフトウェア／ファームウェアのほとんど全てが、ビットブロックを曖昧なものとして扱うことができる。低レベルの実装論理の中で焦点が当てられる部分だけが結局（当該論理記述及び／又は物理記述の詳細に基づいて）適切な論理に切り替わる。

【００７７】

（宣言によるホストのハードウェア環境でのポータビリティ）

組み込みファームウェア／ソフトウェアのホストのハードウェア環境の宣言的な記述は、新たなレベルのポータビリティを提供するものである。従って、本発明は、（マイクロコントローラ・コアがＡＳＩＣファミリ／ライン全体で類似のインストラクションセットをもつ場合に）ファームウェア／ソフトウェア・イメージを変更することなく、ユニークなレベルのポータビリティを達成するものである。図４は、本発明の実施形態の一定のファームウェア／ソフトウェア・イメージのポータビリティを利用するケースを示す。図４に示されているように、固定サイズの単一の組み込みファームウェア／ソフトウェア部品イメージを様々な製品用にカスタマイズすることができる。まず、一定のファームウェア／ソフトウェア・イメージが生成される（「製品設計」工程、「インストール」工程、「コンサルト」工程）。コンサルト工程は、ファームウェア／ソフトウェアプログラミングガイド（ＡＰＩドキュメントなど）を見直し、助言を求めるために利用される。これらの工程が完了すると、本システムが実装され、次に、「コンパイル、リンク、テスト」工程に入る（必要に応じて繰り返される）。通常、これは、必要に応じて実行可能な所望のイメージの生成を繰り返すサイクルである。

【００７８】

初期化工程は一度だけ実行されることが望ましい。各対象製品に対して「追加」（或いは、編集）記述工程が一度実行される。本発明の実施形態では、本工程の自動化が可能に

なり、その結果、真に既製のファームウェア／ソフトウェア部品を再利用することができる。最後に説明した工程は、当該部品を一つ以上の対象製品に組み込むことを含む。基本的には、この工程は、再利用可能なファームウェア／ソフトウェア部品を含む最終的な完成品のアセンブリ工程である。

【００７９】

（例）

以下の例では、所定のマイクロコントローラ・コアのための本発明の態様の動作を示す。上述したように、物理レベルで、所定のＡＳＩＣから見いだされる抽象概念（例えば、汎用入力／出力（ＧＰＩＯ）ピン）がある（これらはＡＳＩＣチップの下側にある物理的なピンである）。特定のホストチップ上の１つ（或いは、それ以上の）で、ピンを論理ＧＰＩＯ信号として直接利用可能な場合がある。（組み込みファームウェア／ソフトウェア・イメージ）に対する様々なニーズの全てに対応するにはピン数が十分でないことがある。それに対する１つの解決方法は、論理ＧＰＩＯ信号記述の数を限定する（そして、その結果、物理ＧＰＩＯピン記述のペア数を減らす）ことである。このアプローチでは、ホストＡＳＩＣの制限に順応するために、固定のイメージの特徴と機能が犠牲となる。適切な記述を単純に変更することだけで十分である。

【００８０】

また、製品設計者（即ち、ボード設計者）はＡＳＩＣピン・イクスパンダメカニズムである別個のチップ／部品（第２の補完的チップ）を使うことができる。主ホストチップといっしょに、このイクスパンダチップは、動作するホストのハードウェア環境と一体的な部品となる。この複雑な構成をこの例に導入することによって、本発明が様々なホストのハードウェア環境の詳細の違いをどのように単純に無視できるのかが示される。さらに、本発明は、所定のファームウェア／ソフトウェア・イメージを変更することなく、それを実行することができる。

【００８１】

動作中の電源の例に戻って、（表１と表２で）示したように、電源は様々な論理ＧＰＩＯ信号を実装するための所定のホストＡＳＩＣピンを利用する。これらの信号を使って一論理ファセット、即ち、「パワーオン」ビヘイビアを実装することができる。その他のＧＰＩＯ信号（とそれに対応するＧＰＩＯピン）を使って、その他の論理ビヘイビア・ファセットを実装することができる。

【００８２】

どのように電源スキームが（特定のホストＡＳＩＣ上で）論理ＧＰＩＯ信号を物理的なＧＰＩＯピンにマッピングされたかについて上述したが、ここで、論理ＧＰＩＯ信号を論理バスにマッピングさせる方法について説明する。ＧＰＩＯ信号の集合（とそれに対応するＧＰＩＯピン）が「ビット・バングド（ｂｉｇ－ｂａｎｇｅｄ）」バスに割り当てられる。この方法で、ピン・イクスパンダチップは機能する。例えば、４本のＧＰＩＯ信号は、１６（２４）通りの異なる信号の組み合わせまで符号化可能である。この場合、表２のＧＰＩＯバス・フラグは正論理であるので、表２のＧＰＩＯバス関連属性は現実的に意味がないわけではない。

【００８３】

従って、ビットブロックを使用した際の記述の力の一部を見ることができる。これは単純な一連のＧＰＩＯインターフェイスを記述するために使うことができる。これは、当該電源で（様々なＧＰＩＯ信号に対して）どの物理的なＡＳＩＣピンを使うかについて記述するために利用される。これは、どのように複数のＧＰＩＯ信号を使って（電源か、電源に配線されるチップ外のピン・イクスパンダのいずれか一方に対する）ありのままのビットバングド・バスインターフェイスを構成するかについて記述するために利用される。

【００８４】

電源プロトタイプスキームは、一般的なデフォルトの実装に付属するものである。この実装論理は、ビットブロックの内容を取り除くものである。デフォルト・ビヘイビアが特定の電源に不適切である場合は、各ビヘイビア・ファセットを（電源スキームで必要に応

じて利用可能である）代替のビヘイビアに置き換えることができる。プロトタイプスキームに対する単純な編集を行うことによって、デフォルトのプロトタイプ・ビヘイビアを置き換えることができる。

【００８５】

組み込みソフトウェア／ファームウェア論理で焦点となる少数の孤立したビットだけが、この特定のビットブロックの解釈に関する情報をもつ必要がある。電源スキームに関連する論理の全てから自然なモジュールが構成される。このモジュールには、（電源スキームのインスタンスのための）ビットブロックをどのように解釈するかについての知識をもつべきである論理だけが含まれる。本発明に係る幾つかのアプリケーションは、モジュールを動的にロードすることを望むが、所定の範囲の所望の適用可能性を所定の固定のモジュール群に対応づけることが可能である。そのようなモジュールを固定サイズの実行可能なイメージに前もってロードすることが可能である。一旦これが実行されると、必要なことは記述ブロックに宣言的な変更を行うことだけである（図３参照）。この種の固定サイズの実行可能なイメージは、再利用可能なソフトウェア／ファームウェア部品となる。

【００８６】

図３は本発明の実施形態の配置アーキテクチャを示す。本図は、本発明を適用した組み込みファームウェア部品を示す。この特定のファームウェア部品は、３つの（可能性として一定サイズの）部分：即ち、ブートブロック、記述ブロック、動作ブロックに分割される固定サイズのファームウェア・イメージを特徴とする。

【００８７】

図３に示されているように、本発明の実施形態の配置アーキテクチャは、コンフィグレーション・ユーティリティ、ホスト、オプションとしての記述レジストリを含む。コンフィグレーション・ユーティリティは、（例えば、製品ライン全体の）ホストＡＳＩＣと周辺デバイスを記述し、製品のアセンブリ後に部品を構成するために使われる。（部品は構築されるが、カスタマイズされないことに注意されたい）（ＡＳＩＣなどの）ホストは、記述ブロックとブート（イメージ）ブロックを含む。記述ブロックは、論理記述と物理記述を含み、ホストＡＳＩＣと周辺デバイスの両方を記述したものである。（所謂固定のイメージのファームウェア／ソフトウェア「部品」を構築するために）固定容量の記憶部を前もって割り当てるか、或いは、（チップ外の／二次的記憶を含むか、或いは、含まない）プールから記憶部を動的に割り当てることができる。ホストは、既知のスキーマタタイプを宣言する（自己記述（ｓｅｌｆ－ｄｅｓｃｒｉｂｉｎｇ））。ブートブロックは、必要に応じて（特に、存在する各周辺デバイスに対して）記述された／パラメータ表現されたイニシャライザを呼び出して、ファームウェアとソフトウェアの更新の仲介を行ってインストールするための固定のイメージである。動作ブロックは、記述によってパラメータ表現された固定のイメージである。

【００８８】

動的発見モニター（ＤＤＭ）を使って、中央集中化された記述レポジトリ（ｒｅｐｏｓｉｔｏｒｙ）との協調を可能にすることができる。本発明の実施形態のＤＤＭは周辺を連続的に監視し、記述をアクティベート或いはアップロードしたり、及び／又は、存在する周辺デバイスを監視する。

【００８９】

オプションとしての記述レポジトリは、（既知の／サポートされている）ホストＡＳＩＣ領域と（既知の／サポートされている）周辺デバイス領域を記述し、オンデマンドで（例えば、ＤＤＭによって新しいデバイスが発見されると）必要な（物理／論理）記述を供給するものである。

【００９０】

（ホスト（ＡＳＩＣ）記述は周辺記述を補完する）

（ソフトウェア／ファームウェア部品のための）ホスト／ＡＳＩＣ処理サポートプラットフォームを記述する際に、論理記述の集合／コンテナ／表が使われる。各記述（ブロックエントリ）は、それに対応するホストのＡＳＩＣ回路／副次システムのインスタンスに

対応する。各周辺デバイスに対しても、通常、同様の論理記述の集合／コンテナ／表がある。これらの論理（とそれに対応する記述）から記述ブロックが構成される（図３参照）。

【００９１】

一般的に、ＡＳＩＣ回路／コンポーネント・システムであるものと、周辺デバイス／コンポーネント・システムであるものとの差異が常に明確であるとは限らない。このため、これらの記述の特徴を統合することは当然のことである。本発明は、ホストと周辺デバイスの両方をサポートするために等しく適用される。

【００９２】

さらに、論理ホスト記述と物理ホスト記述は概念上全く異なるものである。論理ホスト・態様（ａｓｐｅｃｔ）／コンポーネント記述は、結果的に名づけられた／識別された物理記述にマッピングされる値を供給する。周辺記述と同様に、１対１の論理・物理の同一性マッピングを行うことによって、２つの記述をインターリーブすることが可能になる。上述したように、これは、通常、実装の便宜を鑑みて実行されるものである。乱数生成器、ＵＡＲＴポート、不揮発性記憶装置、タコメータ回路、パルス幅変調回路などのＡＳＩＣ回路／コンポーネント－システムのために、プロトタイプスキーマタを生成することができる。そのプロトタイプスキームの集合は、周辺スキーマと組み合わせて（主要な）システムスキーマを構築するためのホストスキーマを含む。

【００９３】

ホスト（ＡＳＩＣ）と周辺（デバイス）の記述を混在させてマッピングされることに注意されたい。例えば、個々の周辺スキーマタのパレットは、（特定ベンダーの）製品ライン用の周辺スキームを定義することができる。新しいモデルがこの製品ラインに入ると、それらは、同じ周辺スキームをいつも再利用することができる。それらは、同じホストスキーマを再利用するか、或いは、再利用しない。周辺スキーマとは無関係に、ホストスキーマをスワップイン／アウトすることができる。

【００９４】

当業者であればすぐにわかることであるが、その他の多くの種類の役に立つスキームグルーピングを割り当てることができる。サイズの制約以外に、本発明の実施形態の実行可能な固定サイズのイメージは、（様々なホストと様々な周辺デバイスの）個々のプロトタイプスキームの全て（或いは、その大多数）をサポートすることができる。実際的（及び経済的）には、未使用のスキームを（ある限度を越えて）携帯することを正当化することは難しい。さらに、実行可能なイメージ自体は、ホストのインストラクションセットに依存する（即ち、それは、例えば、Ｈ８やＡＲＭなどのマクロコントローラ処理のコアに依存する）。

【００９５】

本発明は、（特に、選別製品ラインでは）組み込みファームウェア／ソフトウェア部品を変更する要求がほとんどないシステムのためのものである。ホストのハードウェアを設計し、所望のソフトウェア／ファームウェア部品をレイアウトすることができる。再コンパイルも再初期化も不要であり、実行時での組み込みファームウェア／ソフトウェア・イメージの変更も不要である（オーバレイの変更も動的リンクも動的ロードなども不要）。単一で固定のソフトウェア／ファームウェア・イメージをホストの一連のハードウェアと一連の周辺デバイスで使うことができる。製品ラインの設計者は、組み込みファームウェア／ソフトウェア部品の適用可能性の範囲と製品ラインの計画をマッピングする。このことによって、部品の組み込みが劇的に高速化し、新しい製品モデル開発のための製品化に要する時間を劇的に削減することができ、他方、品質保証コストを劇的に下げることができる。さらに、これによって、（小さな記述的な宣言によって）本分野での多くのハードウェアの変更のほとんどをサポートすることができる。実際には、ハードウェアの変更に対応することは、（動的発見モニターを介して（図３参照））完全に自動化可能である。

【００９６】

（動的ハードウェア環境）

実際は、ホストのハードウェア環境が（実行時に）動的に変わることはまれである。一旦製品が出荷されると、普通、ホストハードウェア環境は固定される。ソフトウェア／ファームウェアホストがＡＳＩＣ（チップ）である場合は、それがプリント基板（ＰＣＢ）に固定的に接続されることがある。まれであるが、ＡＳＩＣを変更できることがある。例えば、補完的なサポートチップは空きソケットにあってもよい。ホストＡＳＩＣチップ自体は一ソケットにあってもよい。これは、それもフィールドアップグレードされる可能性があることを意味する。或いは、ホストはカード（即ち、ドーターカード、メザニンカード、バスカードなど）上にあってもよい。これらも動的に追加可能であり、取り外し可能であり、スワップ可能である。

【００９７】

他方、周辺のハードウェア環境は頻繁に変わる。例えば、ワークロード・ベアリング（ライン／ＣＰＵ）カード（例えば、ブレード）は追加されたり、取りはずされることが多く、メザニンカード・スロット／ヘッダにオプションとしてのハードウェア製品が設置されたり、ディスクドライブは出されたり入れられたりする。

【００９８】

従って、本発明の幾つかの実施形態では、必要に応じて記述ブロックに対する変更を自動化することによって変化する周辺ハードウェア環境をサポートする（図３参照）。具体的には、本発明の幾つかの実施形態では、変化とそれに関連する更新を発見することをサポートする。ここでの発見とは、ハードウェア環境での変化を検出することを意味する。当業者であれば理解できることであるが、発見するための多くの方法が存在する。ほとんどの発見技術は特定のハードウェア環境に特有のものである。例えば、あるハードウェアは、ある形態の非同期事象を（割り込みライン、Ｉ２Ｃバスメッセージ、或いは、その他の形態の入力手段を介して）発生させるように設計されている。（Ｉ２Ｃ（Ｉｎｔｅｒ－ＩＣ）バスは、集積回路（ＩＣ）間での通信リンクを提供する双方向２線シリアルバスである）この非同期事象によって知らされるハードウェアの変化を名づけ／識別するための規約が存在する。

【００９９】

どのように発見がなされたかに無関係に、本発明は異なる周辺デバイス記述を単純にアクティベートする／デアクティベートすることによって検出された変化を処理する能力を備える。代替的に、本発明の幾つかの実施形態では、（例えば、ネットワーク、シリアルポートなどを介して）遠隔の送信元から必要な周辺デバイスの記述を得ることができる。本発明の実施形態の動的発見モニター（ＤＭＭ）を図３に示す。

【０１００】

記述の変更自体は、通常、事象－条件－アクション（ＥＣＡ）トリガメカニズム／メソッドを利用する関連論理にトリガを与えることができる。本発明の実施形態の幾つかの態様には、論理的な／物理記述の所謂「アクティブ」レポジトリを生成するオプションが含まれている。これは結局、アクティブな記述ブロックである（図３参照）。繰り返すが、これは、一定のソフトウェア／ファームウェア・イメージの性質／サイズ／レイアウトに対して一切の変更を加えることなく実行可能である。新しいソフトウェア／ファームウェアを動的にリンクする必要も、動的にロードする必要もない。ソフトウェア／ファームウェアの再初期化は不要である。

【０１０１】

ビットブロックとは、所定のスキーマの中でこのビットブロック（６４－２５６、或いは、それ以上）が表すことのできる様々なデータ構造の全てに対する曖昧なユニオンのようなものであると考えられる。ビットブロックのレイアウト／内容は、利用される論理／物理記述スキームに依存する。ファセット論理（ゲッター／セッターからパワーオン／オフ／リサイクルのような動作のビヘイビアまでの）は、常に、所定のスキーム（スキーマをもつ）に適用される。ファセット論理は、そのスキームに関してどのような仮定を設定できるのかがわかっているので、実行時では、それが受け取るビットブロックを適切にキャストすることができる。

【０１０２】
ほとんどの論理は、ファセット論理よりも上位レベルにある。これは、ほとんどの論理がビットブロックを曖昧なものとして扱うことを意味している。ほとんどの論理は、所定のインスタンスの対応する記述の詳細／値をビットブロックとしてのファセットに入れるだけである。ほとんどの論理は、このビットブロックをどのように解釈利用するかについては全く判っていない。この意味では、ビットブロックは常に曖昧である。しかしながら、特定のファセット論理は、ビットブロックをどのようにレイアウトするか（また、それをどのように解釈／利用するか）についてはわかっている。
【０１０３】
本発明の態様は、（所定の支配的スキーマ内で）統一され／一貫した全プロトタイプスロット／ファセット・インターフェイスを構築するメソッドとともに、（型ではなく）プロトタイプのネストした／階層構造つきのエンティティについて宣言的に記述するメソッドであると考えられる。所定のスキーマ内の所定数のビット（例えば、６４、１２８、或いは、２５６）は、常に、ユニバーサルな／一つの引数として全てのプロトタイプのビヘイビア／メソッド／関数に引き渡される。
【０１０４】
実行時の初期化を行って、スキーム固有のスキームーファセットー実装モジュールのエントリポイントに至るまで、一つのタイプを通常曖昧なブロックにバインドする処理が遅らされる。このモジュールに入るときはいつも、このタイプのビットブロックにバインドすることが繰り返される。この様にバインドを遅らせることによって、本発明は、宣言的な（プロトタイプの）スキーマの記述範囲を大きく拡張するメソッドを提供することができる。これによって、本発明の実施形態のデータドリブンのファームウェア／ソフトウェア論理をより広く応用できるものにすることができる。同様に、固定サイズの一つのファームウェア／ソフトウェア・イメージをはるかに柔軟で、応用可能で価値のあるものにすることができる。
【０１０５】
本発明を特定の実施形態と例に基づいて説明したが、請求項で与えられるより広い本発明の精神と範囲から逸脱することなくその実施形態に対して様々な修正や変更を行えることは明らかである。従って、明細書と図面は限定するためのものではなく例示するためのものである。
【図面の簡単な説明】
【０１０６】
【図１】本発明の一実施形態のアーキテクチャの概念の一例を示す図である。
【図２】本発明の実施形態の一態様に係る実装の一例を示す図である。
【図３】本発明の一実施形態の配置アーキテクチャの一例を示す図である。
【図４】本発明の一実施形態に係る一例の利用ケースのワークフローを示す図である。
【誤訳訂正３】
【訂正対象書類名】図面
【訂正対象項目名】全図
【訂正方法】変更
【訂正の内容】

【図１】

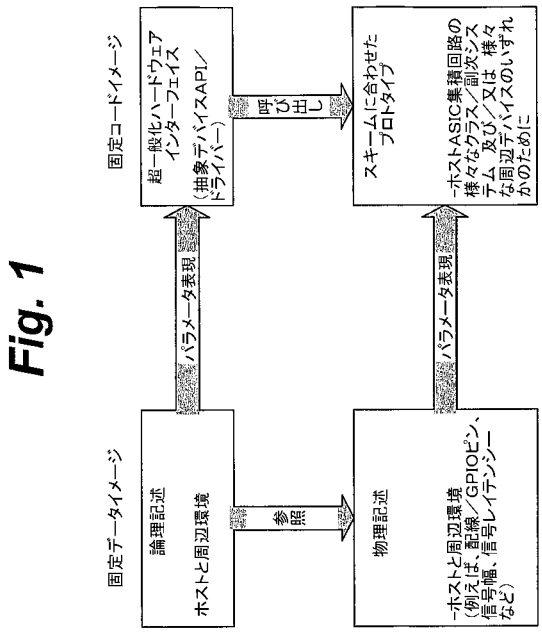

Fig. 1

【図２】

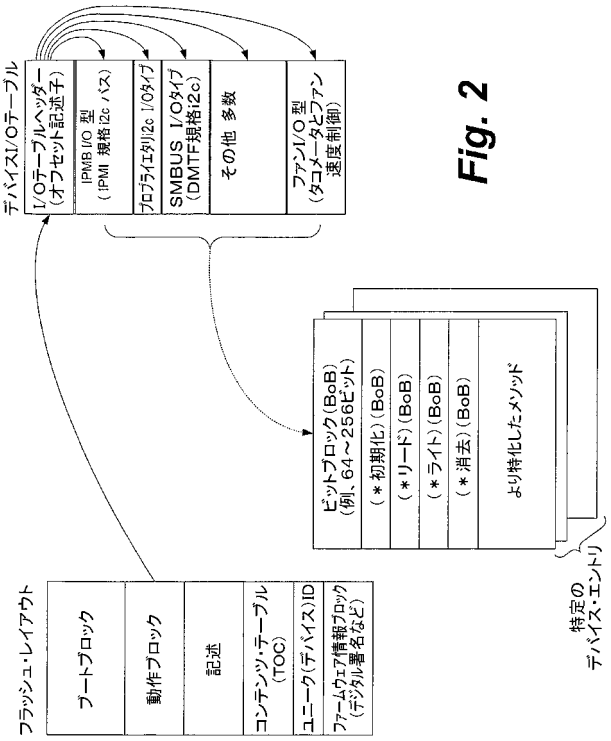

Fig. 2

【図３】

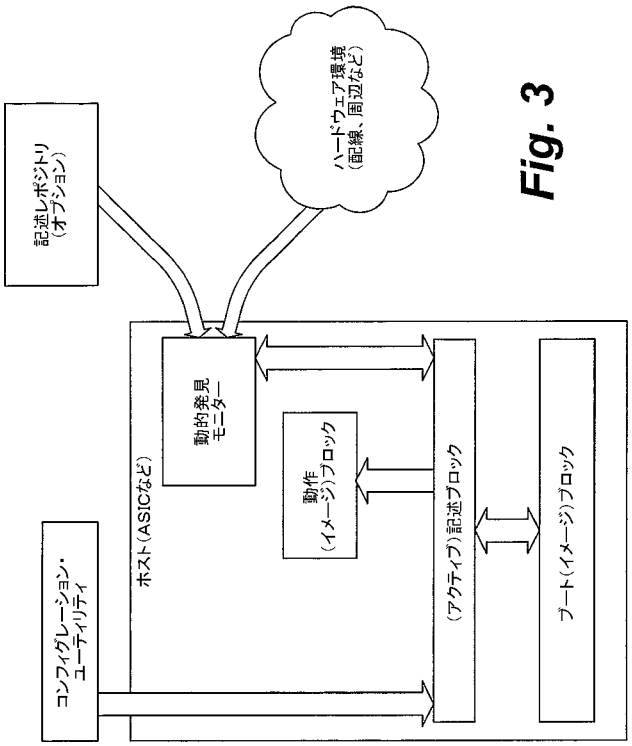

Fig. 3

【図４】

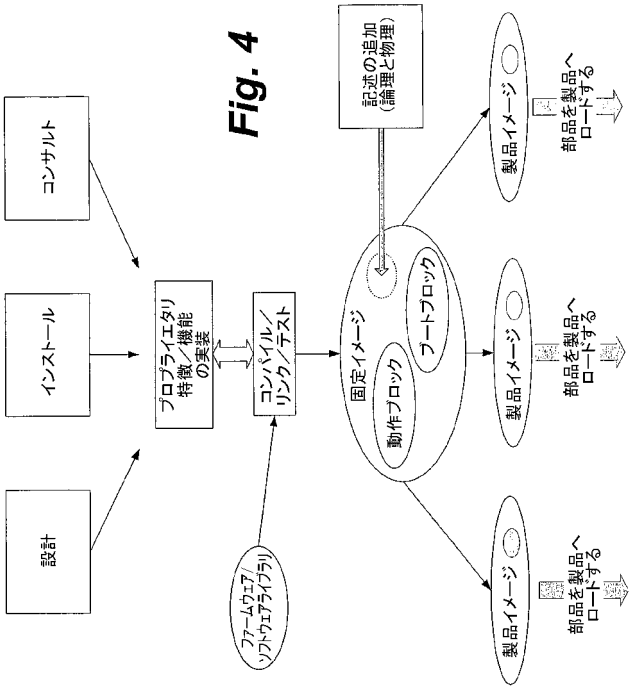

Fig. 4