(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第4427899号
(P4427899)
(45)発行日　平成22年3月10日(2010.3.10)　(24)登録日　平成21年12月25日(2009.12.25)

(51)Int.Cl.　F I
G06F 9/48 (2006.01)　G06F 9/46 452Z

請求項の数 5 (全 12 頁)

(21)出願番号　特願2000-402421(P2000-402421)
(22)出願日　平成12年12月28日(2000.12.28)
(65)公開番号　特開2002-202891(P2002-202891A)
(43)公開日　平成14年7月19日(2002.7.19)
審査請求日　平成19年2月27日(2007.2.27)

前置審査

(73)特許権者　000004260
株式会社デンソー
愛知県刈谷市昭和町１丁目１番地
(74)代理人　100082500
弁理士　足立　勉
(72)発明者　谷　芳伸
愛知県刈谷市昭和町１丁目１番地　株式会社デンソー内
(72)発明者　松田　啓資
愛知県刈谷市昭和町１丁目１番地　株式会社デンソー内
(72)発明者　春名　克俊
愛知県刈谷市昭和町１丁目１番地　株式会社デンソー内

最終頁に続く

(54)【発明の名称】電子制御装置及び記録媒体

(57)【特許請求の範囲】
【請求項１】
複数の入力装置から入力される入力データに基づき出力装置を制御する電子制御装置であって、
前記複数の入力装置から前記入力データを取得するための入力処理プログラムであって前記複数の入力装置のそれぞれに応じた入力処理プログラムを順次実行してなる入力処理と、
前記入力処理の完了後に実行され、前記出力装置への出力データを前記入力処理で得られた入力データに基づき生成するためのアプリケーションプログラムを実行してなるアプリケーション処理と、
前記アプリケーション処理の完了後に実行され、そのアプリケーション処理で生成された出力データを前記出力装置に出力するための複数の出力処理プログラムを順次実行してなる出力処理と、からなる制御処理を繰り返し実行し、
前記入力処理において、前記入力処理プログラムの実行によって得られる入力データについて、その入力データをメモリ（以下、第１のメモリと言う）に記憶する第１の記憶処理と、その入力データに応じて実行されるべきアプリケーションプログラムを表す情報である教示情報を生成する入力調停と、を行い、
前記アプリケーション処理において、前記入力調停にて生成された前記教示情報が表すアプリケーションプログラムを実行するとともに該実行では前記第１のメモリに記憶された入力データを利用するようになっており、その実行で生成できた出力データをメモリ（

以下、第２のメモリと言う）に記憶する第２の記憶処理を実行し、
前記出力処理において、前記第２のメモリに記憶された出力データについて前記出力装置への出力の可否を判断する出力調停を行い、その出力調停で出力可と判断した出力データを、前記出力処理プログラムを実行することによって前記出力装置に出力し、
前記制御処理における前記第１の記憶処理が完了すると、その第１の記憶処理にて前記第１のメモリに記憶した前記入力データを、少なくとも次回の制御処理における前記第１の記憶処理まで更新せず、
前記制御処理における前記入力調停が完了すると、その入力調停にて生成した前記教示情報を、少なくとも次回の制御処理における前記入力調停まで更新せず、
前記制御処理における前記第２の記憶処理が完了すると、前記第２のメモリに記憶した前記出力データを、少なくとも次回の制御処理における前記第２の記憶処理まで更新せず、
前記制御処理における前記出力調停が完了すると、その出力調停での判断結果を、少なくとも次回の制御処理における前記出力調停まで更新しないようになっていること
を特徴とする電子制御装置。
【請求項２】
請求項１に記載の電子制御装置において、
前記制御処理を所定期間毎に実行し、該制御処理をその所定期間内に完了すること
を特徴とする電子制御装置。
【請求項３】
請求項１または２に記載の電子制御装置において、
前記アプリケーションプログラムの実行によって得られる出力データをそのアプリケーションプログラム（以下、前者アプリケーションプログラムと言う）とは異なる他のアプリケーションプログラム（以下、後者アプリケーションプログラムと言う）で利用する場合、その前者アプリケーションプログラムの実行によって得られる出力データであって前記第２のメモリに記憶される出力データを、次回の制御処理における前記入力処理にてその第２のメモリから読み出して前記第１のメモリにコピーすること
を特徴とする電子制御装置。
【請求項４】
請求項１～３のいずれかに記載の電子制御装置において、
プログラムの未実行の期間が、当該電子制御装置による制御異常を回避するために予め定められる期間に達したアプリケーションプログラム（以下、未実行アプリケーションプログラムと言う）がある場合には、前記アプリケーション処理において、その未実行アプリケーションプログラムも実行すること
を特徴とする電子制御装置。
【請求項５】
コンピュータを請求項１～４のいずれかに記載の電子制御装置として動作させるための制御プログラムを記録したコンピュータ読み取り可能な記録媒体。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】
電子制御装置等に関する。
【０００２】
【従来の技術及び発明が解決しようとする課題】
従来より、メモリに記憶されたアプリケーションプログラムに従って、例えば所定時間毎に各種センサ等の状態を入力し、その入力データに応じて制御対象（出力装置）を制御するか否かを判定し、制御すると判定された場合にはその制御対象を制御するための出力を行う電子制御装置が知られている。
【０００３】
このような電子制御装置のアプリケーションプログラムは、制御対象や制御目的などに応

じて所定の処理単位毎に作成し、その作成したそれぞれのアプリケーションプログラムを連続的に実行するように構成すること、すなわち複数のアプリケーションプログラムを順次実行するスケジューリング方式（シンプルスケジューリング）で実行することで、複数の制御対象の制御または１の制御対象の複数の制御を１の電子制御装置で実現している。

【０００４】

しかしながら、このような電子制御装置では、それぞれのアプリケーションプログラムで入力→判断→出力の処理を行っているため次のような問題があった。すなわち、同一の入力装置から複数のアプリケーションプログラムによって入力を行ったり、同一の出力装置に対して複数のアプリケーションプログラムによって出力を行う場合、最初に実行されたアプリケーションプログラムでの処理と後に実行されたアプリケーションプログラムでの処理において、入力装置から入力した情報が変化してしまい処理結果に矛盾が生じたり、最初に実行したアプリケーションプログラムによる処理と後に実行したアプリケーションプログラムによる処理で異なる情報を出力してしまい、出力装置が誤動作する可能性があった。そのため、各処理の結果に矛盾が起きないように各アプリケーションプログラムによる処理において調停を行う必要があり、その調停のために各アプリケーション間で情報をやりとりする必要があるため、各アプリケーション間に依存関係が生じてしまい、システムの複雑さが増すにつれアプリケーションプログラムの設計保守等が困難となるといった問題があった。またこのような電子制御装置では各アプリケーションプログラム毎の処理において入出力を行っていたため、同一の入出力装置に対して複数のアプリケーションプログラムによる処理で入出力を行っており、プログラムに無駄があるといった問題もあった。

【０００５】

そこで本発明は、上述した問題点を解決し、入出力コードの重複を無くすことを可能にしてプログラムを格納するメモリ容量を削減することができ、アプリケーションプログラムのモジュール性（独立性）を高めアプリケーションの自由度を向上させることができ、信頼性を確保することができる電子制御装置等を提供することを目的とする。

【０００６】

上述した問題点を解決するためになされた請求項１に記載の電子制御装置は、制御処理を、図１（ａ）に例示するように入力処理とアプリケーション処理と出力処理に分け順次実行して行う。このとき、入力処理では、複数の入力処理プログラムを順次実行する。一方、アプリケーション処理では、複数のアプリケーションプログラムのうち、入力処理で得られた入力データに応じて実行されるべきアプリケーションプログラムが実行される。そして、出力処理では出力処理プログラムを順次実行して行う。すなわち入出力処理は順次呼び出しを行うスケジューリング方式（シンプルスケジューリング）で構成し、アプリケーション処理はそのアプリケーション処理に必要な入力データが存在した時（外部イベント発生時）にその入力データ（イベント）に応じたアプリケーションプログラムを実行するという方式（イベントドリブン方式）で構成する。したがって、イベントに応じたアプリケーションプログラムのみ実行されるため、処理の不要なアプリケーションプログラムが実行されることがなくなる。

【０００７】

また、アプリケーション処理では、入力装置からの入力、及び出力装置への出力は行われないようになっている。したがって、入出力の前後の依存関係を考慮することなくアプリケーションプログラムを作成することが可能となり、信頼性が向上する。また、アプリケーションプログラムを独立したプログラムとして扱うことができるようになり、例えば機能単位で、アプリケーションプログラムの追加や削除を容易に行うことができる。またこのように制御処理を、入力処理、アプリケーション処理、出力処理に分けることにより、従来のようにそれぞれのアプリケーション処理内で入出力を行う場合に比べ、プログラムのサイズを小さくすることが可能となり、ＲＯＭ等のメモリ容量を削減することができる。

【０００９】

そしてさらに、請求項１の電子制御装置は入力処理にて入力調停を行う。すなわち、入力データに応じて実行されるべきアプリケーションプログラムを表す情報である教示情報を生成する。例えば、入力データＡと入力データＢが共に１の場合に、入力調停結果としてイベントＡを発生するといった処理を行う。そしてアプリケーション処理において、このイベントＡ発生時に実行すべきアプリケーションプログラムを実行する。このようにすることで、アプリケーションプログラムにおける処理に必要なレベルで各アプリケーションプログラムを実行させることができ、アプリケーションプログラム内で入力データに矛盾がないか等をいちいちチェックする必要もなくなる。したがって、アプリケーションプログラムの作成が容易になり、モジュール性も高くなって信頼性も向上する。

【００１０】

同様に、出力処理において、データの出力開始前に出力調停を行う。このようにすれば、例えば、アプリケーション処理では出力ポートの詳細な構成を意識することなく、より高いレベルの出力データを出力するだけでよくなる。また、各アプリケーションプログラム同士が出力データの内容に矛盾がないかを判定する必要がなくなり、アプリケーションプログラムの作成が容易になり、モジュール性も高くなって信頼性も向上する。

【００１１】

ここで、入力調停と出力調停とを行う５階層で構成されるプログラムの間の関係を図１（ｂ）に例示する。このように入力調停と出力調停との双方を行うようにすれば、アプリケーションプログラムの独立性は極めて高くなる。

【００１２】

そして、さらに、入力データの同期をとるようになっている。具体的には、ある周期の入力処理で得たデータは、次の周期の入力処理を実行するまでは変化させないようになっている。このようにすれば、データ入力完了後（入力処理完了後）の各処理において使用する入力データはすべて同じ状態となる。したがって、例えばアプリケーション処理中に入力データが変化してしまい、変化前の入力データと変化した入力データとの双方に基づいたアプリケーション処理を行ってしまうような矛盾は発生しなくなる。したがって信頼性（安全性）を容易に確保することができる。

【００１３】

同様に、出力データの同期をとるようになっている。具体的には、ある周期のアプリケーション処理で確定した出力データは、次の周期のアプリケーション処理を実行するまでは変化させないようになっている。これによれば、アプリケーション処理完了後の各処理において使用する出力データはすべて同じ状態となる。したがって、例えば出力処理中に出力データが変化してしまい、変化前の出力データと変化した出力データとの双方に基づいた出力処理を行ってしまうような矛盾が発生しなくなる。したがって信頼性（安全性）を容易に確保することができる。

【００１４】

また、入力調停結果の同期をとるようになっている。具体的には、ある周期の入力調停で確定した情報は、次の周期の入力調停を実行するまでは変化させないようになっている。これによれば、入力調停完了後の各処理において使用する入力調停結果は、すべて同じ状態となる。したがって、例えばアプリケーション処理中に入力調停結果が変化してしまい、変化前の入力調停結果と変化した入力調停結果との双方に基づいたアプリケーション処理を行ってしまうような矛盾が発生しなくなる。したがって信頼性（安全性）を容易に確保することができる。

【００１５】

同様に、出力調停結果の同期をとるようになっている。具体的には、ある周期の出力調停で確定した情報は、次の周期の出力調停を実行するまでは変化させないようになっている。これによれば、出力調停完了後の各処理において使用する出力調停結果は、すべて同じ状態となる。したがって、例えばデータの出力中に出力調停結果が変化してしまい、変化前の出力調停結果と変化した出力調停結果に基づいたデータの出力を行ってしまうような矛盾が発生しなくなる。したがって信頼性（安全性）を容易に確保することができる。

【００１６】
ここで、制御処理は繰り返し実行することとなるが、この場合請求項２に示すように構成することができる。請求項２に示すように所定期間毎に周期的に制御処理を実行し、その所定期間内に制御処理が完了することで、例えばリアルタイムＯＳのような複雑な処理を極力省いた低コストで信頼性の高い装置とすることができる。
またアプリケーションプログラム間で処理結果を利用する場合は、請求項３のようにして、直接アプリケーション間で処理結果（データ）の受け渡しをしないようにするとよい。例えば、アプリケーション間のデータの通信は、出力処理や入力処理を１度通して別のアプリケーションに通知することで、安全設計が可能な仕組みとなる。そして、アプリケーション・アプリケーション間の結合が切れて、機能の付け外しが可能な機構となる。よって、アプリケーションプログラム同士の独立性が高くなり、アプリケーションプログラムの開発が容易になる。また、入出力信号の信頼性（安全性）を容易に確保することができる。
【００１７】
ところで、アプリケーションプログラムは上述したイベントドリブンで実行されるため、例えば、あるアプリケーションプログラムに対応するイベントが発生せずにそのアプリケーションプログラムが起動しない状態が長時間継続する場合が考えられる。このような場合に、外部のノイズ等によってそのアプリケーションプログラムによる処理の処理結果や出力データを格納したＲＡＭのデータが変化してしまうと、その変化したデータに基づいた出力処理により、制御対象の誤動作が発生する。そこで請求項４のようにするとよい。請求項４のように、プログラムの未実行の期間が、制御対象の誤動作（換言すると、電子制御装置による制御異常）を回避するために予め定められる期間が経過したアプリケーションプログラムを実行し、その処理結果を格納したＲＡＭの内容をリフレッシュする。すなわち、例えばイベントによって起動されるアプリケーションプログラムは、定期的なイベントの発生によっても起動される仕組みを持たせ、通常の処理よりも長い間隔で、かつ、内部のデータのＲＡＭ化け等による不都合が起こってもシステムに支障をきたさない間隔でのデータリフレッシュを保証するようにする。こうすることで、誤動作の影響を最小限に抑えることができる。
【００１８】
以上のような構成とすることで、入出力信号の信頼性の確保、アプリケーションの自由度向上、プログラムを格納するＲＯＭ容量を節約することができる。
【００１９】
なお、請求項５に示すような、コンピュータを請求項１～４の何れかに記載の電子制御装置として動作させるための制御プログラムを記録したコンピュータ読み取り可能な記録媒体によれば、コンピュータを請求項１～４の何れかに記載の電子制御装置として動作させることができる。そのような制御プログラムは、例えば、フロッピーディスク（登録商標）、光磁気ディスク、ＣＤ－ＲＯＭ、ハードディスク、ＲＯＭ等のコンピュータ読み取り可能な記録媒体に記録し、必要に応じてコンピュータシステムにロードして起動することにより用いることができる。
【００２０】
【発明の実施の形態】
以下、本発明が適用された実施例について図面を用いて説明する。なお、本発明の実施の形態は、下記の実施例に何ら限定されることなく、本発明の技術的範囲に属する限り種々の形態を採りうることは言うまでもない。
【００２１】
図２は、実施例としての電子制御装置であるボデーＥＣＵ１０を備えたボデー系システム１００を示す図である。ボデー系システム１００は、特許請求の範囲におけるネットワークに相当する車内ＬＡＮ５０に接続されたボデーＥＣＵ１０と、ボデーＥＣＵ１０と車内ＬＡＮ５０を介して通信可能な通信対象ＥＣＵ２０を備える。ボデーＥＣＵ１０及び通信対象ＥＣＵ２０は、ＣＰＵ、ＲＯＭ、ＲＡＭ、Ｉ／Ｏ等を備えたコンピュータシステムで

あり、Ｉ／Ｏには入力装置３０や出力装置４０や車内ＬＡＮ５０が接続されている。

【００２２】

通信対象ＥＣＵ２０は、入力装置３０の状態を読み取ってパケットを生成し車内ＬＡＮ５０を介してボデーＥＣＵ１０へ送信する。また、車内ＬＡＮ５０を介して入力されるボデーＥＣＵ１０からのパケットの制御情報に基づいて出力装置４０を制御する。通信対象ＥＣＵ２０は、例えば、Ｄ席ドアＥＣＵ２０ａやＰ席ドアＥＣＵ２０ｂやインパネＥＣＵ２０ｃ等である。Ｄ席ドアＥＣＵ２０ａはドライバ側のドアに設置されたＥＣＵであり、入力装置３０としてＤ席ドアコントロールスイッチ３０ａなどが接続されており、出力装置４０としてＤ席ドアロックモータ４０ａやＤ席パワーウインドウモータなどが接続されている。Ｐ席ドアＥＣＵ２０ｂは補助席側のドアに設置されたＥＣＵであり、入力装置３０としてＰ席ドアコントロールスイッチ３０ｂ等が接続され、出力装置４０としてＰ席ドアロックモータ４０ｂやＰ席パワーウインドウモータなどが接続されている。また、インパネＥＣＵは、インパネに設置されたＥＣＵであり、例えば、出力装置４０としてブザー４０ｃやランプ４０ｄ等が接続されている。

【００２３】

ボデーＥＣＵ１０は、通信対象ＥＣＵ２０から車内ＬＡＮ５０に送信されるパケットを所定時間毎に受信して、Ｉ／Ｏを介して取り込み、取り込んだパケットの内容に応じたアプリケーションを起動して、出力装置４０の制御が必要な場合には、制御対象の通信対象ＥＣＵ２０への制御指示を含むパケットを生成して送信する制御処理を行う。

【００２４】

このようなボデーＥＣＵ１０の処理について図３を参照して説明する。図３に示すように、ボデーＥＣＵ１０は、電源が投入されると、Ｓ１で、各Ｉ／ＯポートやＲＡＭ等のシステムの初期化を行う初期化処理を行う。続くＳ２で、アイドルタスクを管理するアイドルタスクマネージャを実行する。そして、Ｓ３でアイドル時に実行するユーザ定義処理を実行し、ＣＰＵの状態を監視するＣＰＵ監視マネージャを実行する。そして再びＳ２に戻りアイドルタスクマネージャを実行する。このようにＳ２～Ｓ４をループして実行する。

【００２５】

そしてＣＰＵには、図示しないタイマから５ｍｓ毎にタイマ割込みがかかる。このタイマ割込みによって、Ｓ５のシステムマネージャへ処理が移行し、Ｓ６の入力処理、Ｓ７の入力調停処理、Ｓ８のアプリケーション処理、Ｓ９の出力調停処理、Ｓ１０の出力処理、Ｓ１１のスリープマネージャを順次実行して、タイマ割込みから復帰し、元のＳ２～Ｓ４で構成されるアイドルループに戻る。なお、このタイマ割込みによるメイン処理は、タイマ割込みの間隔である５ｍｓよりも短時間で終了するように構成されている。

【００２６】

このうち、Ｓ６の入力処理からＳ１０の出力処理に至る各ステップの処理をさらに詳細に説明する。

Ｓ６では、Ｉ／Ｏから入力処理を行う入力処理プログラムを順次実行する。すなわち、入力処理プログラム１、入力処理プログラム２、入力処理プログラム３、入力処理プログラム４、…、入力処理プログラムｉ、のように順次実行する。すべての入力処理プログラムの実行が終了すると、Ｓ７の入力調停処理へ移行する。

【００２７】

Ｓ７の入力調停処理は、Ｓ６の入力処理によってＩ／Ｏから入力したデータを総合して調停を行う処理である。すなわち、Ｓ８のアプリケーションプログラムを起動させるためのイベントフラグを立てるか否かを、入力処理（Ｓ６）によって入力されたデータに基づいて判定する処理である。入力調停処理（Ｓ７）は、それぞれのイベントに応じた複数の入力調停プログラムで構成されており、入力調停プログラム１、入力調停プログラム２、…入力調停プログラムｊ、のように順次実行する。そして、すべての入力調停処理プログラムの実行が終了すると、Ｓ８のアプリケーション処理へ移行する。

【００２８】

アプリケーション処理（Ｓ８）は、複数のアプリケーションプログラム１、アプリケー

ションプログラム２、…、アプリケーションプログラムｋで構成されており、各アプリケーションプログラムは直接Ｉ／Ｏへの入出力処理を行わないように構成してある。すなわち入力は、メモリを媒介して、特許請求の範囲における入力処理に相当する入力層（Ｓ６の入力処理と、Ｓ７の入力調停処理に相当する）によって行い、出力は、メモリを媒介して、特許請求の範囲における出力処理に相当する出力層（Ｓ９の出力調停処理と、Ｓ１０の出力処理に相当する）によって行うのである。

【００２９】

また各アプリケーションプログラム間の情報のやりとりは、直接的には行わない。すなわち、図４に示すように、例えばアプリケーションプログラム１とアプリケーションプログラム４が実行される場合に、アプリケーションプログラム１を実行した結果（処理結果）のデータは、アプリケーションプログラム４では使用しないように構成する。つまり、アプリケーションプログラム１は、処理結果をＲＡＭ上のアプリ処理結果格納領域に記憶するが、そのデータはアプリケーションプログラム４の処理からは直接的には参照しない。アプリケーションプログラム４による処理にてアプリケーションプログラム１の処理結果を必要とする場合には、予めＳ７の入力調停処理で、アプリ処理結果格納領域に格納された処理結果を入力データ格納領域にコピーするようにする。そしてイベントフラグに応じてアプリケーションプログラム４が起動された際にその入力データ格納領域から処理結果のデータを参照する。したがって、例えアプリケーションプログラム１の直後にアプリケーションプログラム４を実行する場合であっても、アプリケーションプログラム４は出力結果格納領域ではなく、入力結果格納領域を参照して処理に利用するため、直前のアプリケーションプログラム１の処理結果ではなく、１周期前のタイマ割込み時のアプリケーションプログラム１の処理結果を利用して処理を行うのである。

【００３０】

このように、入出力処理と入出力以外の処理を分け、データの受け渡しをアプリケーションプログラム間で直接的に行わないようにすることで、各アプリケーションの独立性が高まり、機能単位であるアプリケーションプログラムの追加削除が容易にできるようになる。

【００３１】

また、いずれのアプリケーションプログラムを実行するかは、管理用のアプリケーションプログラムである簡易ＯＳ処理によって決定する。この決定処理は、予め各アプリケーションプログラムがどのイベントが発生した際（イベントフラグが立ったとき）に実行されるかを登録するイベントテーブルに基づいて行う。このイベントテーブルには、イベントフラグに対応したアプリケーションプログラムのアドレスが登録（記憶）されており、入力調停処理によってイベントフラグが立てられた場合（イベントが発生した場合）には、対応するアプリケーションプログラムのアドレスをコールして、処理をそのアプリケーションプログラムに移行する。なお、１のイベントに対して複数のアプリケーションがイベントテーブルで対応付けられている場合には、そのイベントに対応したアプリケーション同士は、順次実行するようにする。すなわち、イベントフラグ１に対応するアプリケーションプログラムが、アプリケーションプログラム１、アプリケーションプログラム２、アプリケーションプログラムｘと複数ある場合には、アプリケーションプログラム１が終了したらアプリケーションプログラム２へ処理を移行し、アプリケーションプログラム２が終了したらアプリケーションプログラム３に処理を移行するのである。そして、すべてのイベントフラグに対応するアプリケーションプログラムの処理が終了した場合に、簡易ＯＳ処理を終了して、Ｓ９へ移行する。

【００３２】

Ｓ９の出力調停処理は、Ｓ８のアプリケーション処理によってアプリ処理結果格納領域に記憶されたデータに基づきＳ１０の出力処理で出力を行うか否かを調停する処理である。すなわち、例えば、複数のアプリケーションプログラムが同一の出力対象に対する異なる出力指示を処理結果格納領域に格納した場合に、いずれの処理結果を反映させて出力するかを調停する処理を行うのである。例えば、出力調停処理（Ｓ９）は、複数の出力調停プ

ログラムで構成されており、出力調停プログラム１、出力調停プログラム２、…出力調停プログラムｌ、のように順次実行する。そして、すべての出力結果調停プログラムの実行を完了すると、処理はＳ１０の出力処理へ移行する。

【００３３】

Ｓ１０の出力処理では、Ｉ／Ｏへの出力処理を行う出力処理プログラムを順次実行する。すなわち、出力処理プログラム１、出力処理プログラム２、…、出力処理プログラムｍ、のように順次実行する。すべての出力処理プログラムの実行が終了すると、Ｓ１１のスリープマネージャへ移行する。

【００３４】

このようにして、従来ひとまとめで行っていた処理を、Ｓ６の入力処理、Ｓ７の入力調停処理、Ｓ８のアプリケーション処理、Ｓ９の出力調停処理、Ｓ１０の出力処理の５つに分割し、各処理において、データの同期を取る。すなわち、図５に示すように、ある時間で確定したデータを次の処理部へ受け渡し、それ以後は変化させない。例えば、Ｓ６の入力処理でスイッチがＯＮと確定した後は、次の周期の入力処理を実行するまでは、たとえ入力の状態がＯＦＦになっていたとしてもスイッチの状態はＯＮとして処理を行う。同様にＳ７の入力調停処理で確定した情報は次の周期の入力調停処理まで変化させない。またＳ８のアプリケーション処理で確定した情報は次のアプリケーション処理まで変化させない。そして、出力調停処理によって確定した情報は、次の出力調停処理まで変化させない。したがって、各処理部の処理完了時点でデータが確定（同期）する。そして、各処理部の処理中に前の処理部で生成したデータは変化しないので、例えば変化前のデータに基づく処理と変化後のデータに基づいて処理を行ってしまうような処理矛盾を避けることができ、安全設計も容易にできる。

【００３５】

これまで説明したように入力層、出力層のプログラムは順次実行を行う。すなわちシンプルスケジューリングで実行する。一方、アプリケーション層では、どのアプリケーションプログラムが実行されるかは、イベントの有無を判断して、簡易ＯＳ処理で決定する。

【００３６】

このような構成とすることで、イベントの発生しないアプリケーションプログラムは、イベントが発生しない間は実行されないことになる。したがって、実行されないアプリケーションプログラム用のアプリ処理結果格納領域は、その間更新されない。そのため外部のノイズ等によってアプリ処理結果格納領域のデータが書き換えられた場合（いわゆるＲＡＭ化け）には、Ｓ８の出力調停処理、Ｓ９の出力処理によって誤った出力がなされてしまう場合がある。このような状況は、アプリ処理結果記憶領域が長時間更新されなければされないほど発生する可能性が高くなる。特に、例えば制御の誤りによって人に対して悪影響を与える可能性のあるシステムを制御する場合には、このようなことが起こらないように、メモリの値を所定時間毎に更新することが望ましい。

【００３７】

そこで、図６に示すように簡易ＯＳ処理において、一定時間以上起動されていないアプリケーションプログラムを順次起動して、そのアプリケーションプログラム用のアプリ処理結果記憶領域を更新する。この一定時間は、タイマ割込み間隔よりも長い間隔で、かつアプリ処理結果記憶領域のデータが外部のノイズ等によって変化してしまってもシステムの制御に支障をきたさない時間とする。すなわちボデー系システム１００（図１参照）においては、ＲＡＭ化けによって、例えば図示しないパワーウィンドウの閉制御のように本来閉じてはいけない状態で閉じる指示が出力される状況が想定される。このような異常な制御が続く時間が５０ｍｓ程度であれば人に対して悪影響を及ぼさない。したがって、例えば４０ｍｓの間、起動されていないアプリケーションプログラムから順次実行するようにする。例えば、図６に示すように、アプリ２とアプリ３とアプリ４が、４０ｍｓの間実行されていなければ、アプリ２を実行し、アプリ３を実行し、アプリ４を実行する。そして、イベントに応じたアプリを実行する前述の処理を行う。このような構成を採ることで、ＲＡＭ化けが起きても、信頼性や安全性を確保することができる。

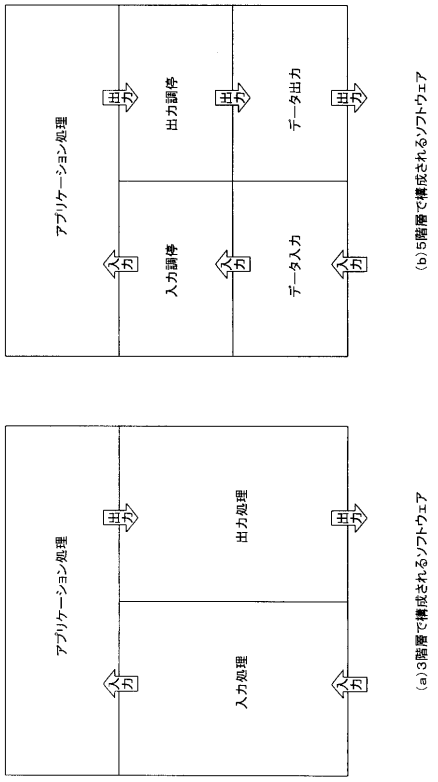
【００３８】
こうしたボデー系システム１００によれば、例えば、Ｄ席ドアコントロールスイッチ３０ａと図示しない入力装置３０としてのワイヤレス（電波等を使用し、ドアを施錠・解錠するシステム）の双方から異なる指示がほぼ同時に連続して入力された場合、すなわち例えば、Ｄ席ドアコントロールスイッチ３０ａの状態に基づきＤ席ドアＥＣＵ２０ａが全席ドア施錠の指示を出し、ワイヤレスは全席ドア解錠の指示を車内ＬＡＮ５０を介してボデーＥＣＵ１０に出力した場合、ボデーＥＣＵ１０は、Ｓ６の入力処理によってＩ／Ｏからこれらの情報を取り込む。そして、Ｓ７の入力調停処理によって、いずれか１の指示を優先して利用する。例えば、Ｄ席ドアコントロールスイッチ３０ａの状態を優先するように調停する。そして、全席ドア施錠の指示が入力された旨のイベントフラグを立て処理をＳ８へ移行する。よって、簡易ＯＳ処理によって、このイベントフラグに対応するアプリケーションに実行が移される。例えば、アプリ１の処理によって、アプリ処理結果格納領域に全席ドア施錠の指示が書き込まれる。一方このとき、他の入力装置３０の状態に基づくイベントによって例えばアプリ２が実行され、その処理結果として、Ｄ席ドア解錠の指示がアプリ処理結果格納領域に書き込まれていたとする。この場合、Ｓ９の出力調停処理によって、出力の調停を行い全席ドア施錠を優先して、Ｓ１０の出力処理に渡し、Ｓ１０の出力処理によって、出力装置４０の接続された各ドアＥＣＵに対してドア施錠の指示を送信する。こうして、極めて短い時間にドアが解錠→施錠または施錠→解錠することを防止することができる。すなわち複数の矛盾する入力データや出力データを調停することができ、安全で信頼性の高い制御を実現することができる。

【００３９】
このように、人間の操作する装置系に対する制御を行う場合には、人間と制御対象の機械の処理時間（認識時間）の差を上述した調停や同期によって、調整することができる。また車内ＬＡＮ５０による遅延等によるデータの到着のずれ等の影響もこれらの機構により抑えることができる。

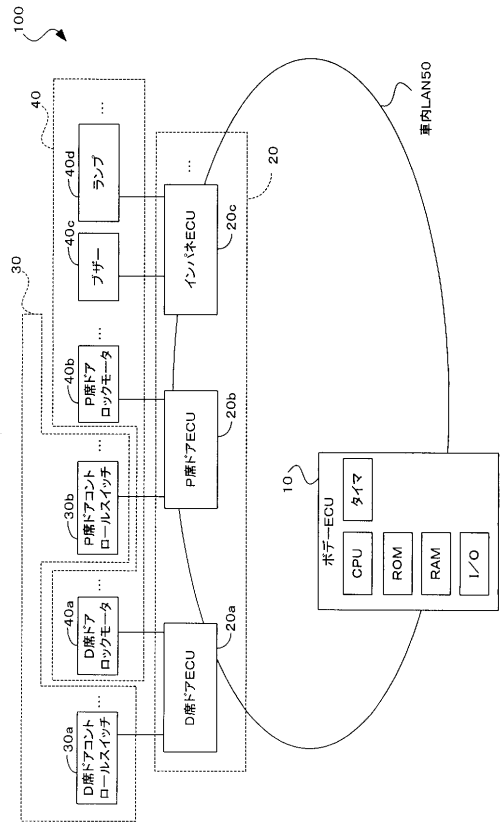
【図面の簡単な説明】
【図１】　制御処理の階層化を説明する説明図である。
【図２】　ボデー系システムの構成を示すブロック図である。
【図３】　各処理の分割の状態と実行順を説明する説明図である。
【図４】　アプリケーションプログラム間のデータのやりとりの方法を示す説明図である。
【図５】　各処理におけるデータの同期を説明する説明図である。
【図６】　メモリ状態の再設定に関する説明図である。

【符号の説明】
１０…ボデーＥＣＵ
２０…通信対象ＥＣＵ
２０ａ…Ｄ席ドアＥＣＵ２０ａ
２０ｂ…Ｐ席ドアＥＣＵ２０ｂ
３０…入力装置
３０ａ…Ｄ席ドアコントロールスイッチ
３０ｂ…Ｐ席ドアコントロールスイッチ
４０…出力装置
４０ａ…Ｄ席ドアロックモータ
４０ｂ…Ｐ席ドアロックモータ
４０ｃ…ブザー
４０ｄ…ランプ
５０…車内ＬＡＮ
１００…ボデー系システム

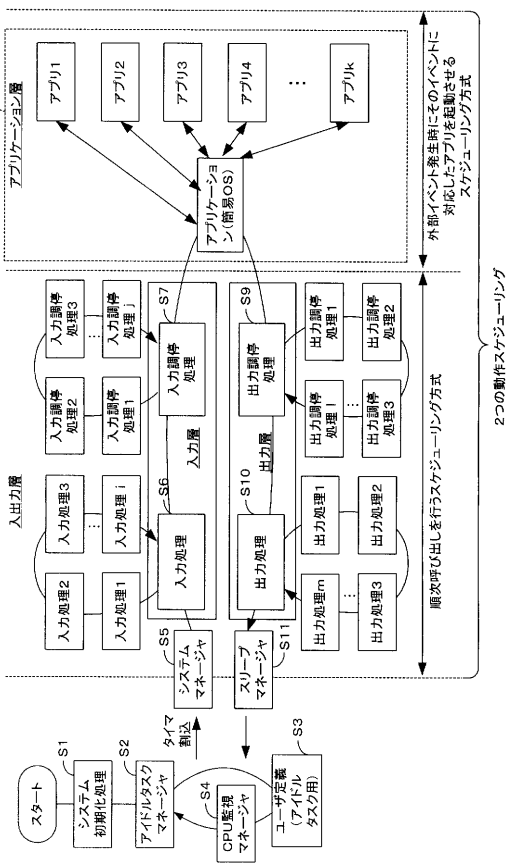
【図１】


アプリケーション処理
出力
出力調停
データ出力
入力
入力調停
データ入力
(b)5階層で構成されるソフトウェア
アプリケーション処理
出力
出力処理
入力
入力処理
(a)3階層で構成されるソフトウェア

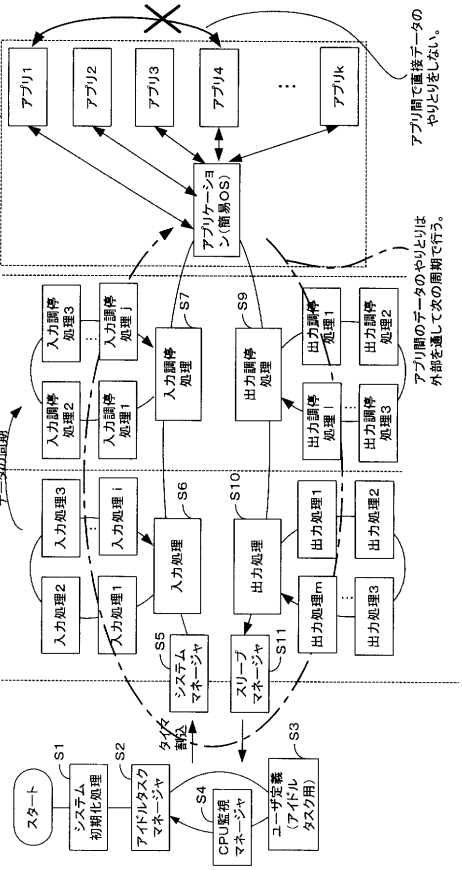
【図２】


100
40
30
車内LAN50
20
10
ランプ
ブザー
インパネECU
P席ドアECU
D席ドアECU
ボデーECU
CPU
ROM
RAM
I/O
タイマ
D席ドアコントロールスイッチ
P席ドアコントロールスイッチ
D席ドアロックモータ
P席ドアロックモータ

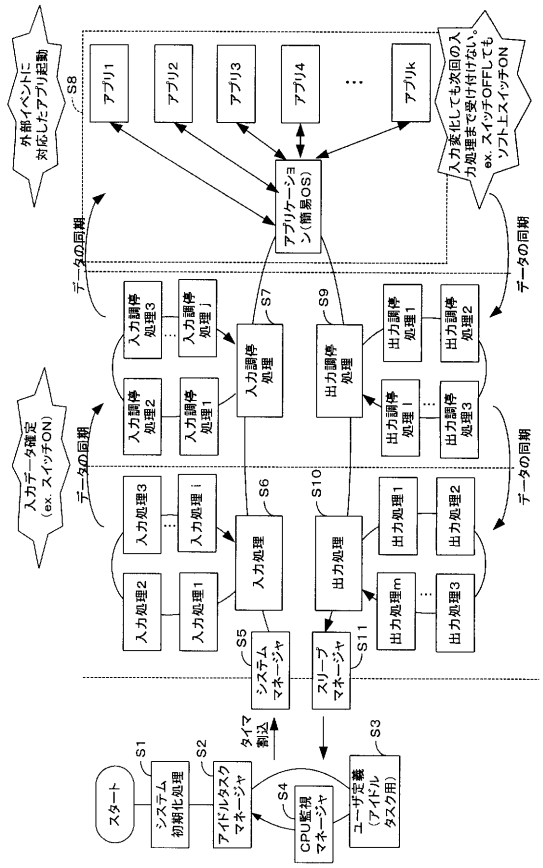
【図３】


スタート
システム初期化処理
アイドルタスクマネージャ
CPU監視マネージャ
ユーザ定義（アイドルタスク用）
タイマ割込
システムマネージャ
スリープマネージャ
入出力層
入力層
出力層
入力処理
出力処理
入力調停処理
出力調停処理
アプリケーション層
アプリケーション（簡易OS）
アプリ1
アプリ2
アプリ3
アプリ4
アプリk
外部イベント発生時にそのイベントに対応したアプリを起動させるスケジューリング方式
順次呼び出しを行うスケジューリング方式
2つの動作スケジューリング

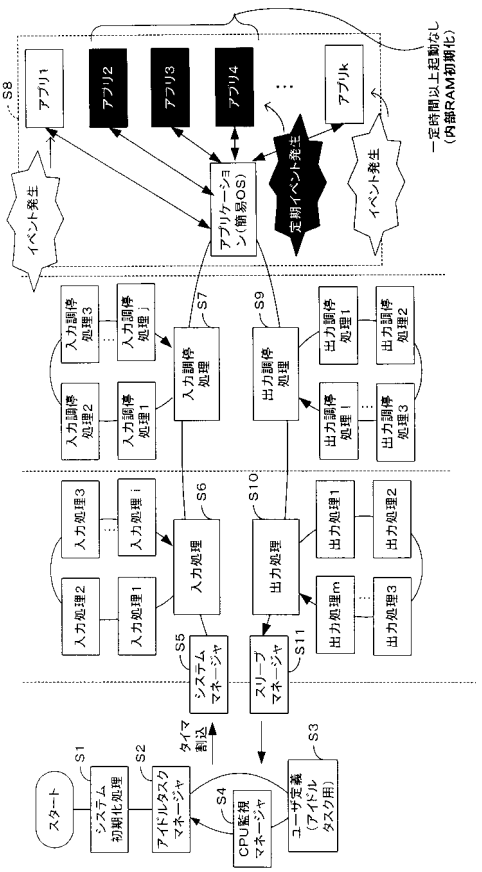
【図４】


スタート
システム初期化処理
アイドルタスクマネージャ
CPU監視マネージャ
ユーザ定義（アイドルタスク用）
タイマ割込
システムマネージャ
スリープマネージャ
アプリケーション（簡易OS）
アプリ1
アプリ2
アプリ3
アプリ4
アプリk
データの周期
アプリ間で直接データのやりとりをしない。
アプリ間のデータのやりとりは外部を通して次の周期で行う。

【図５】


外部イベントに対応したアプリ起動
S8
アプリ1
アプリ2
アプリ3
アプリ4
アプリk
入力変化しても次回の入力処理まで受け付けない。ex. スイッチOFFしてもソフト上スイッチON
アプリケーション(簡易OS)
データの同期
入力データ確定(ex. スイッチON)
入力調停処理3
入力調停処理j
入力調停処理2
入力調停処理1
入力調停処理
S7
出力調停処理
S9
出力調停処理1
出力調停処理2
出力調停処理3
入力処理3
入力処理i
入力処理2
入力処理1
入力処理
S6
出力処理
S10
出力処理1
出力処理2
出力処理m
出力処理3
S5
システムマネージャ
スリープマネージャ
S11
タイマ割込
S1
スタート
システム初期化処理
S2
アイドルタスクマネージャ
S4
CPU監視マネージャ
S3
ユーザ定義(アイドルタスク用)

【図６】


S8
アプリ1
アプリ2
アプリ3
アプリ4
アプリk
一定時間以上起動なし(内部RAM初期化)
イベント発生
アプリケーション(簡易OS)
定期イベント発生
イベント発生
入力調停処理3
入力調停処理j
入力調停処理2
入力調停処理1
入力調停処理
S7
出力調停処理
S9
出力調停処理1
出力調停処理2
出力調停処理3
入力処理3
入力処理i
入力処理2
入力処理1
入力処理
S6
出力処理
S10
出力処理1
出力処理2
出力処理m
出力処理3
S5
システムマネージャ
スリープマネージャ
S11
タイマ割込
S1
スタート
システム初期化処理
S2
アイドルタスクマネージャ
S4
CPU監視マネージャ
S3
ユーザ定義(アイドルタスク用)

フロントページの続き

(72)発明者　江川　邦隆
愛知県刈谷市昭和町１丁目１番地　株式会社デンソー内
(72)発明者　加藤　滋郎
愛知県刈谷市昭和町１丁目１番地　株式会社デンソー内
(72)発明者　新田　修一
愛知県刈谷市昭和町１丁目１番地　株式会社デンソー内

審査官　井上　宏一

(56)参考文献　特開平１０－０１５８３６（ＪＰ，Ａ）
特開平１０－０２１０９３（ＪＰ，Ａ）
特開２０００－０９７８１０（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
G06F　9/46 -9/54