(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第6257506号
(P6257506)
(45)発行日　平成30年1月10日(2018.1.10)　(24)登録日　平成29年12月15日(2017.12.15)

(51)Int.Cl.　F I
G06F 11/00 (2006.01)　G06F 9/06 630B

請求項の数 11 (全 18 頁)

(21)出願番号　特願2014-249263(P2014-249263)
(22)出願日　平成26年12月9日(2014.12.9)
(65)公開番号　特開2015-115073(P2015-115073A)
(43)公開日　平成27年6月22日(2015.6.22)
審査請求日　平成28年4月20日(2016.4.20)
(31)優先権主張番号　13196209.4
(32)優先日　平成25年12月9日(2013.12.9)
(33)優先権主張国　欧州特許庁(EP)

(73)特許権者　506012213
ディスペース　デジタル　シグナル　プロセッシング　アンド　コントロール　エンジニアリング　ゲゼルシャフト　ミット　ベシュレンクテル　ハフツング
dspace digital signal processing and control engineering GmbH
ドイツ連邦共和国　パデルボルン　ラーテナウシュトラーセ　２６
Rathenaustr. 26, D-33102 Paderborn, Germany

最終頁に続く

(54)【発明の名称】電子制御装置のメモリ内のソフトウェアを変更する方法

(57)【特許請求の範囲】
【請求項１】
電子制御装置（ＥＣＵ）のメモリ内のソフトウェアを変更する方法であって、
前記メモリは、少なくとも１つの固体メモリ（ＳＰ１）と、揮発性のデータを記憶する少なくとも１つの作業メモリ（ＲＡＭ）とを含み、
前記固体メモリは、前記ソフトウェアの少なくとも一部を形成する複数のオリジナルプログラムルーチンを内部に記憶しており、該複数のオリジナルプログラムルーチンは前記電子制御装置の少なくとも１つのプロセッサによって処理されるものであり、
前記電子制御装置には、少なくとも１つの前記オリジナルプログラムルーチンに加えて又はこれに代えて処理されるバイパスルーチンが格納されており、
当該バイパスルーチンの処理は、前記プロセッサによるプログラムステップの処理フローにおいて前記オリジナルプログラムルーチンが少なくとも終了する前にサービス機能（Ｚ）の呼び出しを行い、前記サービス機能に付加的にバイパスルーチンのメモリアドレスを引き渡し、メモリアドレスの引き渡しが成功した場合に、前記サービス機能が対応するバイパスルーチンを呼び出し、メモリアドレスの引き渡しが成功しなかった場合には、バイパスルーチンの呼び出し無しで前記サービス機能を終了するように行われ、
前記バイパスルーチンは、前記固体メモリ（ＳＰ１）の外部に存在するメモリ領域としての揮発性の前記作業メモリ（ＲＡＭ）へ伝送され、
前記バイパスルーチンを開始するアドレスが、前記サービス機能に引き渡される、
方法において、

第１のステップで、前記サービス機能による前記バイパスルーチンの呼び出しを不活性化し、
第２のステップで、前記プロセッサが前記バイパスルーチンを処理しない時間において、前記バイパスルーチンを新たなバイパスルーチンによって置換し、
第３のステップで、前記サービス機能による前記新たなバイパスルーチンの呼び出しを作動し、
前記第１のステップ及び前記第２のステップ及び前記第３のステップを、前記プロセッサが前記ソフトウェアを処理するランタイムで実行する、
ことを特徴とする方法。
【請求項２】
前記サービス機能への前記バイパスルーチンのメモリアドレスの引き渡しは、記憶されているテーブル（ＡＤ）内の所定の位置を指示するポインタを前記サービス機能（Ｚ）に引き渡し、前記テーブル（ＡＤ）内の前記所定の位置にエントリが存在する場合、当該エントリによって形成されるアドレスのバイパスルーチンを呼び出し、前記テーブル（ＡＤ）内の前記所定の位置にエントリが存在しない場合もしくは当該エントリが妥当でないことが標示されている場合、バイパスルーチンの呼び出し無しで前記サービス機能（Ｚ）を終了するように行われる、
請求項１記載の方法。
【請求項３】
前記バイパスルーチンの記憶は、前記プロセッサが前記ソフトウェアを処理するランタイムで実行される、
請求項１記載の方法。
【請求項４】
既存のバイパスルーチンを新たなバイパスルーチンによって置換する前に、前記ポインタによって定められた前記テーブル（ＡＤ）内の位置の既存のエントリが消去されるか又は妥当でないものとして標示される、
請求項２記載の方法。
【請求項５】
全てのバイパスルーチンにそれぞれステータスレジスタが割り当てられ、各ステータスレジスタは、割り当てられたバイパスルーチンが実行される場合に第１の値へセットされ、それ以外の場合に第２の値へセットされるか又は消去され、
前記置換は、前記テーブル（ＡＤ）内の既存のエントリが消去されるか又は妥当でないものとして標示され、かつ、置換されるべきバイパスルーチンのステータスレジスタが第２の値にセットされたか又は消去された場合にのみ行われる、
請求項４記載の方法。
【請求項６】
前記テーブルのエントリが消去され、前記ステータスレジスタへの問合せが行われ、ステータスレジスタ状態に応じて前記テーブルへの新たなエントリが行われる、
請求項５記載の方法。
【請求項７】
カウンタにより、所定の時点で置換されるべきバイパスルーチンを処理するプロセスの数が監視され、
サービス機能（Ｚ）によってバイパスルーチンが呼び出されるたびに前記カウンタが増分され、バイパスルーチンが終了される際に前記カウンタが減分され、さらに、テーブル内の既存のエントリが消去されるか又は妥当でないものとして標示され、かつ、前記カウンタがゼロとなっている場合にのみ、置換が行われる、
請求項４記載の方法。
【請求項８】
前記テーブルのエントリが消去され、前記カウンタへの問合せが行われ、カウンタ状態に応じて前記テーブルへの新たなエントリが行われる、

請求項７記載の方法。
【請求項９】
電子制御装置（ＥＣＵ）のメモリ内のソフトウェアを変更する方法であって、
前記メモリは、少なくとも１つの固体メモリ（ＳＰ１）と、揮発性のデータを記憶する少なくとも１つの作業メモリ（ＲＡＭ）とを含み、
前記固体メモリは、前記ソフトウェアの少なくとも一部を形成する複数のオリジナルプログラムルーチンを内部に記憶しており、該複数のオリジナルプログラムルーチンは前記電子制御装置（ＥＣＵ）の少なくとも１つのプロセッサによって処理されるものであり、
前記電子制御装置には、少なくとも１つの前記オリジナルプログラムルーチンに加えて又はこれに代えて処理される第１のバイパスルーチンが格納されており、
当該第１のバイパスルーチンの処理は、前記プロセッサによるプログラムステップの処理フローにおいて前記オリジナルプログラムルーチンが少なくとも終了する前にサービス機能の呼び出しを行い、前記サービス機能に付加的に前記第１のバイパスルーチンのメモリアドレスを引き渡し、メモリアドレスの引き渡しが成功した場合、前記サービス機能が対応する前記第１のバイパスルーチンを呼び出し、メモリアドレスの引き渡しが成功しなかった場合には、前記第１のバイパスルーチンの呼び出し無しで前記サービス機能を終了するように行われ、
前記第１のバイパスルーチンは、前記固体メモリ（ＳＰ１）の外部に存在するメモリ領域としての揮発性の前記作業メモリ（ＲＡＭ）へ伝送され、
前記第１のバイパスルーチンを開始するアドレスが前記サービス機能に引き渡される、方法において、
前記プロセッサが前記ソフトウェアを処理するランタイムで、前記第１のバイパスルーチンに加えて、第２のバイパスルーチンが、前記固体メモリ（ＳＰ１）の外部に存在するメモリ領域としての揮発性の前記作業メモリ（ＲＡＭ）へ伝送され、
前記第２のバイパスルーチンを開始するアドレスが前記サービス機能に引き渡される、ことを特徴とする方法。
【請求項１０】
前記作業メモリ（ＲＡＭ）への前記バイパスルーチンの伝送は、前記電子制御装置のインタフェースを介して、前記インタフェースに接続されたコンピュータにより行われる、請求項１から９までのいずれか１項記載の方法。
【請求項１１】
前記消去及び前記問合せ及び前記新たなエントリは、前記電子制御装置のインタフェースに接続されたコンピュータにより行われる、請求項６又は８記載の方法。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、電子制御装置のメモリ内のソフトウェアを変更する方法、すなわち、メモリが、少なくとも１つの固体メモリと、揮発性のデータを記憶する少なくとも１つの作業メモリとを含み、固体メモリは、ソフトウェアの少なくとも一部を形成する複数のオリジナルプログラムルーチンを内部に記憶しており、当該複数のオリジナルプログラムルーチンは制御装置の少なくとも１つのプロセッサによって処理されるものであり、制御装置には、少なくとも１つのオリジナルプログラムルーチンに加えて又はこれに代えて処理されるバイパスルーチンが格納されており、当該バイパスルーチンの処理は、プロセッサによる複数のプログラムステップの処理のフローにおいて、オリジナルプログラムルーチンが少なくとも終了する前にサービス機能の呼び出しを行い、記憶されているテーブル内の所定の位置を指示するポインタを独立変数としてサービス機能に引き渡し、テーブルの当該位置にエントリ（入力値）が存在する場合、当該エントリによって形成されたアドレスのバイパスルーチンを呼び出し、エントリが存在しなかった場合には、バイパスルーチンの呼び出し無しでサービス機能を終了するように行われる方法に関する。
【背景技術】

【０００２】
自動車分野では、ソフトウェアを使用することにより、純粋に機械的な手段では実現できないレベルの効率を有する調整機構乃至制御機構が可能となる。したがって、今日の自動車は、複数の電子制御装置ＥＣＵ（エレクトロニックコントロールユニット）から成るネットワークを有している。各電子制御装置は、種々のプログラムルーチンを処理するそれぞれ少なくとも１つのプロセッサと、作業メモリと、例えばフラッシュメモリによって実現される固体メモリとを含む。
【０００３】
また、全ての制御装置には、センサデータを受信し、アクチュエータを駆動し、相互に通信するためのインタフェースが設けられている。ソフトウェア産業において通常見られるように、ＥＣＵプログラミングにおいても既存の手段へのアクセスが行われる。
【０００４】
新たな制御装置の開発の基礎として、既存の制御装置、例えば先行シリーズの機関に対する機関制御装置が用いられる。この場合、個別の機能の選択的修正又は新たな開発が行われるが、旧いコードの大部分が維持される。これに代えて、メーカが特有の要求に応じて自分自身で制御装置を修正できるよう、納品時に完全にプログラミングされている状態の開発制御装置を注文するケースも多い。どちらのシナリオでも、新たに開発される機能を制御装置の現行のバイナリコードに関連づけて、それぞれのオリジナル機能に置き換えなければならない点は共通している。当該関連づけ過程はバイパス化と称される。
【０００５】
機能のバイパス化を実現するために、例えば、サービスベースのバイパス化へのアクセスが行われる。サービス機能とは、制御装置のメモリに存在する特別な機能である。プログラムコードの所定の位置でプログラムコードが呼び出され、バイパスが技術的に実行される。これは、オリジナル機能に加えてもしくはこれに代えてバイパスルーチンを呼び出し、バイパスルーチンへエントリ量を供給し、処理後にバイパスルーチンによって書き込まれた値を制御装置のメモリの適切な位置へ格納することにより、行われる。
【０００６】
基本的には、バイパスルーチンは、外部システムに存在するか又は制御装置そのものに存在する。後者の手段、いわゆる内部バイパス化は、僅かな付加ハードウェアのみで、専用のインタフェースを設けることなく可能である。
【０００７】
開発制御装置は、一般に、機能バイパスの実行のために準備されていなければならない。サービスベースのバイパス化プロセスが適用される場合、これは、置換される機能の前方及び／又は後方又は内部で、サービス機能呼び出しをバイナリコードに組み入れる必要があることを意味する。
【０００８】
理想的には、こうしたサービス機能呼び出しはソースコード平面で構成されているべきである。ただし、実際には、サービス機能呼び出しを直接にオリジナルＥＣＵアプリケーションのバイナリコードに組み入れるステップは別のプロセスで行われるため、常に利用可能というわけではない。
【０００９】
新たなバイパスコードの実行は、最近の従来技術によれば、制御装置のフラッシュメモリをそれぞれ全く新たにプログラミングすることを前提としている。しかし、この手法には時間がかかり、実行中のテストシーケンスを中断するだけでなく、物理的に制限された数のメモリ過程しか行えないフラッシュメモリを疲弊させてしまう。
【００１０】
さらなる問題点は、多くの制御装置が妥当性検査を実行しているということである。プログラムコードが破壊されていないことを保証するために、制御装置はそのメモリ内容のチェックサムを形成し、その値が正しい場合にのみ動作を無制限に続行する。こうした機器では、チェックサムが更新される場合にしか内部バイパスコードを実行できない。これ

はユーザにとってはどのシナリオにおいても不可能である。
【００１１】
従来技術に対応する典型的なサービスベースの内部バイパス化のシナリオでは、固有のプログラムルーチンに加え、専用のバイパスサービス機能をフラッシュメモリに設けている。バイパスのために用意されている任意のプログラム機能ｆは、サービス機能の２種の呼び出しを拡張している。
【００１２】
ユーザは例えば専用のソフトウェアによりバイパスルーチンをフラッシュメモリにインストールし、当該ソフトウェアがテーブル内のバイパスルーチンのメモリアドレスを入力する。サービス機能の第１の呼び出しは、例えば置換されるべきプログラムルーチンの直前で行われる。サービス機能は独立変数として第１のテーブルエントリに対するインデクスを受け取る。そこにメモリアドレスが存在していれば、サービス機能は当該アドレスのバイパスルーチンを呼び出す。それ以外の場合、サービス機能は受動状態にとどまる。
【００１３】
テーブルは、バイパスルーチンのメモリアドレスを直接に含むことができるか、又は、例えば、システムにインストールされているバイパスルーチンに一義的に割り当て可能な識別番号を含むことによって間接的に伝達することができる。後者の手段は、１つもしくは複数のバイパスルーチンが制御装置のプロセッサからはアドレシングできない外部のメモリ媒体に記憶されている場合に特に有利である。
【００１４】
バイパスルーチンは、実行されている間、実行によって形成される全ての値をバッファメモリへ書き込む。バイパスルーチンの処理後、プロセッサは再び本来のプログラムルーチンへジャンプして戻る。この場合、プログラムルーチンは、バイパス機能が実行されたか否かにかかわらず、終了まで適切に実行される。
【００１５】
プログラムルーチンの実行後、サービス機能の第２の呼び出しが行われる。サービス機能は第２のテーブルエントリに対するインデクスを独立変数として受け取り、その中の上書きルーチンのアドレスを探索する。上書きルーチンはバイパスルーチンに適合するように調整されており、バイパスルーチンとともにインストールされている。上書きルーチンは、バイパスルーチンの結果をバッファメモリから読み出し、これによって、置換されるべきプログラムルーチンが書き込んだ変数を上書きする。
【００１６】
プログラムルーチンによって処理された変数に関する情報は、例えば、制御装置のａ２ｌファイルに由来する。これは、各開発制御装置に添付されている技術的な仕様のファイルである。当該情報は、特に、制御装置上で動作するルーチンからそのつど読み出され書き込まれる変数に対応する第１のブロックと、変数のメモリアドレスを指定する第２のブロックとを含む。バイパス機能の設計については、プログラムルーチンのソースコードが既知である必要はない。
【先行技術文献】
【非特許文献】
【００１７】
【非特許文献１】L.Weiss, "Microcode Patch Facility", IBM Technical Disclosure Bulletin USA, Bd.25, Nr.11A, Apr.1983, pp.5624-5625
【非特許文献２】J.Lyu et al., "A Procedure-Based Dynamic Software Update", Dependable Systems and Networks, 2001
【発明の概要】
【発明が解決しようとする課題】
【００１８】
本発明の課題は、フラッシュメモリの完全に新たなプログラミングを要することなく、既存のプログラムルーチン又は既存のバイパスルーチンを置換できるようにし、さらに、

当該置換を特段の負荷なくいつでも、特にランタイムで行えるようにすることである。有利には、チェックサムの変更が不要となり、制御装置の停止乃至シャットダウンを要さない迅速なローディング過程が可能になるとよい。なお、置換とは少なくとも１回の機能的置換であればよく、必ずしもオリジナルプログラムルーチンの既存のコードを物理的に置換しなくてよい。

【課題を解決するための手段】

【００１９】

この課題は、電子制御装置のメモリ内のソフトウェアを変更する方法であって、前記メモリは、少なくとも１つの固体メモリと、揮発性のデータを記憶する少なくとも１つの作業メモリとを含み、前記固体メモリは、前記ソフトウェアの少なくとも一部を形成する複数のオリジナルプログラムルーチンを内部に記憶しており、該複数のオリジナルプログラムルーチンは前記制御装置の少なくとも１つのプロセッサによって処理されるものであり、前記制御装置には、少なくとも１つの前記オリジナルプログラムルーチンに加えて又はこれに代えて処理されるバイパスルーチンが格納されており、当該バイパスルーチンの処理は、前記プロセッサによるプログラムステップの処理フローにおいて前記オリジナルプログラムルーチンが少なくとも終了する前にサービス機能の呼び出しを行い、前記サービス機能に付加的にバイパスルーチンのメモリアドレスを引き渡し、メモリアドレスの引き渡しが成功した場合に、前記サービス機能が対応するバイパスルーチンを呼び出し、メモリアドレスの引き渡しが成功しなかった場合には、バイパスルーチンの呼び出し無しで前記サービス機能を終了するように行われ、前記バイパスルーチンは、特に前記制御装置のインタフェースを介して、前記インタフェースに接続されたコンピュータにより、前記固体メモリの外部に存在するメモリ領域へ、特に揮発性の前記作業メモリへ伝送され、前記バイパスルーチンを開始するアドレスが、前記サービス機能に引き渡される、方法において、第１のステップで、前記サービス機能による前記バイパスルーチンの呼び出しを不活性化し、第２のステップで、前記プロセッサが前記バイパスルーチンを処理しない時間において、前記バイパスルーチンを新たなバイパスルーチンによって置換し、第３のステップで、前記サービス機能による前記新たなバイパスルーチンの呼び出しを作動し、前記第１のステップ及び前記第２のステップ及び前記第３のステップを、前記プロセッサが前記ソフトウェアを処理するランタイムで実行することにより、解決される。

【図面の簡単な説明】

【００２０】

【図１】従来技術のバイパス化のための開発制御装置を示す図である。

【図２】従来技術のサービス機能の動作を示す図である。

【図３】仮想のａ２１ファイルのセクションを示す図である。

【図４】本発明のバイパス化のための開発制御装置を示す図である。

【図５】本発明のサービス機能の動作を示す図である。

【図６】複数のプロセスを表す環境における本発明のサービス機能の動作を示す図である。

【図７】サービス呼び出しを開発制御装置の機械コードへ後から組み込む手段を示す図である。

【図８】オーバレイメモリを利用してサービス呼び出しをプログラムコードへ後から組み込む手段を示す図である。

【発明を実施するための形態】

【００２１】

本発明の方法では、バイパスルーチンが、特に制御装置のインタフェースを介して、インタフェースに接続されたコンピュータにより、固体メモリの外部に存在するメモリ領域へ、特に揮発性の作業メモリへ伝送され、バイパスルーチンを開始するアドレスがポインタによって定められたテーブル内の位置へ入力される。

【００２２】

バイパスルーチンを固体メモリ（フラッシュメモリ）以外の別のメモリ領域へ書き込む

ことにより、フラッシュメモリが負荷されず、よって、書き込みサイクルが寿命を低下させるような影響を受けないことが保証される。

【００２３】

また、この場合、サービス機能の各呼び出しは提供者側で既に実現されており、ユーザが後からプログラムコードに組み入れる必要はないという利点が得られる。このため、フラッシュメモリのみに関連する周期的な妥当性検査に付随する問題が生じない。

【００２４】

刊行物L.Weiss, "Microcode Patch Facility", IBM Technical Disclosure Bulletin USA, Bd.25, Nr.11A, Apr.1983, pp.5624-5625には、バイパスルーチンをコンピュータシステムの作業メモリへ伝送すること、及び、バイパスルーチンのメモリアドレスをサービス機能へ引き渡すことによってバイパスルーチンを作動することが開示されている。

【００２５】

さらに、従来技術から、プロセッサによる処理の期間中、置換されるべきコードにプロセッサがアクセスを試みた場合にプロセスを停止するウォッチポイントを置換されるべきプログラムルーチンの前方にセットすることによってプログラムコードを変更することが公知である（J.Lyu et al., "A Procedure-Based Dynamic Software Update", Dependable Systems and Networks, 2001）。

【００２６】

本発明の重要な利点は、オリジナルプログラムルーチンの置換又は既存のバイパスルーチンの置換にバイパスルーチンを利用でき、その際に制御装置を停止させなくて済むということである。したがって、有利には、バイパスルーチンが作業メモリに記憶される場合に、当該バイパスルーチンの記憶を、プロセッサによるソフトウェアの処理のランタイムで行うことができる。

【００２７】

先行のバイパスルーチンが記憶されていない場合、本発明では、ユーザ又は使用されているソフトウェアが、バイパスルーチンを、最初にポインタが指示しているフラッシュメモリ以外（例えば作業メモリ）のそれまで空だった位置へ記憶した後、バイパスルーチンのアドレスをテーブル内へ記憶する。上述した構成のごとくバイパスルーチンに加えて上書きルーチンが設けられている場合、ポインタが指示している、それまで空だったテーブル位置に対しても、この記憶が同様に行われる。なお、ポインタは、上書きルーチンを呼び出すための独立変数としてサービス機能に引き渡される。

【００２８】

これに対して、既にバイパスルーチン及び／又は上書きルーチンが構成されている場合、つまり、各テーブル位置が各アドレスのエントリによって占有されている場合、本発明の実施形態によれば、既存のバイパスルーチンを新たなバイパスルーチンによって置換する前に、ポインタによって定められたテーブル内の位置に存在するエントリが消去されるか又は不活性化される。なお、当該置換は、プロセッサが既存のバイパスルーチンを処理しない時間中に行われる。同様に、既存の上書きルーチンが置換されるケースでも、既存の上書きルーチンが処理されない時間において置換を行うと有利である。

【００２９】

後者のケースでは、交換前に、バイパスルーチン及び上書きルーチンのアドレスを含む２つのテーブルエントリが消去及び／又は不活性化される。その後、サービス機能呼び出しにより、後続の全てのプログラムサイクルにおいて、そのつど空のエントリが探索され、これらは受動状態に留められる。つまり、制御装置コードがオリジナルプログラムルーチンによって実行される。

【００３０】

その後、バイパスルーチンは、時間的に非クリティカルに、すなわち、時間的な危険なしで、消去可能であり、新たなバイパスルーチンによって置換される。また、このことは上書きルーチンにも当てはまる。バイパスルーチンコードが完全に作業メモリへ書き込まれる場合、バイパスルーチン開始の作業メモリのアドレスは、テーブル内のポインタが定

めている位置で入力され、当該バイパスルーチンコードが独立変数としてサービス機能へ引き渡される。上書きルーチンが設けられる場合、同じことが上書きルーチンでも同様に行われる。

【００３１】

これに代えて、旧いバイパスルーチンの消去を行わないこともできる。旧いバイパスルーチンに少なくとも機能的に置き換わる新たなバイパスルーチンを、旧いバイパスルーチンに加えて、作業メモリの空き領域にインストールしてもよい。このことも同様に上書きルーチンにも当てはまる。テーブルエントリは、インストール完了後に、そのつど新たな妥当なアドレスにしたがって変更される。

【００３２】

本発明の有利な実施形態によれば、バイパスルーチンが実行されない状態となってから既存のバイパスルーチンへのアクセスが行われることが保証されるように行われる。まさに実行されているルーチンを消去もしくは操作してしまうと、制御装置が機能エラーを起こし、テスト用自動車もしくはテスト台が損壊するほどの重大な影響が生じうるからである。

【００３３】

上記の点を保証するために、本発明の方法の実施形態では、バイパスルーチン、特に全てのバイパスルーチンにそれぞれステータスレジスタが割り当てられ、各ステータスレジスタは、割り当てられたバイパスルーチンが実行される場合にセットされ、それ以外の場合に消去される。ここで、置換は、テーブル内の既存のエントリが消去され、かつ、置換されるべきバイパスルーチンのステータスレジスタが消去された場合にのみ行われる。

【００３４】

別の実施形態によれば、特にサービス機能によってバイパスルーチンが呼び出されるたびにカウンタが増分され、バイパスルーチンが終了される際に当該カウンタが減分される。さらに、テーブル内の既存のエントリが消去され、かつ、カウンタがゼロとなった場合にのみ、置換が行われる。この実施形態は、制御装置上に並行処理される多数のプロセスが存在しており、相互の中断がありうるために種々の優先度を付してプロセスを実行するケースで有利であり、これにより、同じバイパスルーチンもしくは代替的なバイパスルーチンの中断及び新たな実行、又は、同一もしくは複数のバイパスルーチンの複数回の同時処理が可能となる。カウンタ値がゼロであることは、所定の時点で動作しているプロセスのいずれもが、所定のバイパスルーチンもしくは別の構造段に存在しているバイパスルーチンのいずれかをその時点で実行していないことを意味する。なお、ステータスレジスタ及びカウンタは、バイパスルーチンだけでなく、バイパスルーチンに接続されている上書きルーチンに設けることもできる。この場合、プロセッサが１つもしくは２つのルーチンの置換時点でバイパスルーチン及び上書きルーチンの双方を実行しないことが保証される。

【００３５】

また、ステータスレジスタ及びカウンタを全てのバイパスルーチン及び上書きルーチンに対して設けてもよい。この場合、プロセッサによって既存のバイパスルーチンもしくは上書きルーチンのいずれも実行されないことが保証される。

【００３６】

本発明の有利な実施形態では、インタフェースを介して接続されているコンピュータ上のソフトウェアによりテーブルエントリが消去され、カウンタ及び／又はレジスタへの問合せが行われ、カウンタ状態及び／又はレジスタ状態に応じてテーブルへの新たな入力が行われる。

【００３７】

バイパスルーチンの呼び出しに作用するサービス機能呼び出しは、有利には、ソースコード平面で制御装置コード内に準備されることにより実現される。

【００３８】

これに対して、本来は行われないが、オリジナルプログラムルーチンをバイパスルーチ

ンによって置換したいという場合には、本発明の別の実施形態により、サービス機能呼び出しを後から制御装置のプログラムコード内に構成するとよい。

【００３９】

本発明によれば、プログラムルーチンのための固体メモリに本来存在していないサービス機能呼び出しを構成するために、有利な第１の実施形態として、バイパスルーチンを記憶するプログラムルーチンの少なくとも一部を、固体メモリのオリジナルのメモリ領域から別のメモリ領域（特には同じ固体メモリの別のメモリ領域）へ移し替えることができる。オリジナルのメモリ領域に残っているプログラムルーチン部分と移し替えられたプログラムルーチン部分との間にはプログラム分岐が設けられる。この場合、少なくとも１つのサービス機能呼び出しが、残っているプログラムルーチン部分に続いて、又は、移し替えられたプログラムルーチン部分に続いて、オリジナルのメモリ領域のうちローディングによって空いた一部の領域に記憶される。ここで「移し替えられたプログラムルーチン部分に続いて」とは、開始部に続いても終了部に続いてもよい。

【００４０】

上述したプログラム分岐は、例えば、所定のアドレスでの処理を続行するようにプロセッサをトリガするジャンプ命令によって行われる。当該アドレスは、ジャンプ命令において指示されており、移し替えられたプログラムルーチン部分又は先行して置換されたサービス機能呼び出しがどこに記憶されているかを表している。

【００４１】

移し替えに関連して、さらに、移し替えのために設けられているプログラムルーチン部分のアドレスデータを検査して当該アドレスデータを移し替えの前もしくは後に適合化するか、又は、アドレスデータを含まないが適合化は必要なプログラムルーチン部分のみを移し替えるように構成できる。アドレスデータの適合化は、例えば、移し替えのために設けられているプログラムルーチン部分の内部のアドレスに関連する絶対アドレスデータが移し替え後にも移し替えられたプログラムルーチン内で同じ値もしくは同じ命令を指示するように行われる。反対に、当該適合化を、移し替えのために設けられているプログラムルーチン部分以外のアドレスに関連する相対アドレスデータが移し替え後にも移し替えられたプログラムルーチン外で同じ値もしくは同じ命令を指示するように行ってもよい。

【００４２】

フラッシュメモリの所定領域への移し替えが行われる場合、本発明によれば、この移し替えは、プロセッサがソフトウェアを処理する制御装置のランタイム以外で行われる。また、作業メモリの部分領域への移し替えが行われる場合も、これは、移し替えられるべき部分を含むプログラムルーチンがプロセッサによって現に処理されていないことが保証されないかぎり、ランタイムでは行うことができない。このために、例えば、現に処理されているメモリ領域を指示しているプロセッサポインタについて、当該メモリ領域が部分的に置換すべきルーチンのメモリ領域であるか否かが検査される。

【００４３】

サービス機能呼び出しを後から実現する別の手段として、固体メモリでプログラムルーチンに対しては本来設けられていないサービス機能呼び出しを構成するために、制御装置のオーバレイメモリが利用される。オーバレイメモリのメモリアドレスは、割り当て情報により、固体メモリ内の置換されるべきプログラムルーチンのプログラムルーチン命令のメモリアドレスに割り当てられる。これにより、プロセッサは、固体メモリ内のアドレスのプログラムルーチン命令に代えて、オーバレイメモリ内の対応するアドレスの命令を処理し、オーバレイメモリ内の対応するアドレスに、作業メモリの所定のアドレス以降のプログラム処理を続行するようにプロセッサをトリガするジャンプ命令を記憶させる。なお、ここでの所定のアドレスはジャンプ命令によって識別される。さらに、当該アドレス以降作業メモリのメモリ領域には、サービス機能呼び出し、及び、少なくとも１つのプログラムルーチン命令の再構成、及び、オリジナルプログラムルーチンに戻る旨のジャンプ命令が記憶される。

【００４４】

ここで、本発明は、既知の構造の制御装置にオーバレイメモリと称されるメモリ領域が存在しており、制御装置のオーバレイ機能がスイッチオンされた場合に、固体メモリのアドレス位置の命令に代えて、固体メモリのアドレスの割り当て情報によって対応づけられているオーバレイメモリのアドレスの命令が実行されるという事実を利用している。

【００４５】

割り当て情報は、プロセッサレジスタもしくは記憶されているテーブルの内容によって定めることができる。

【００４６】

このように行われる割り当てにより、固体メモリのアドレスに存在するプログラムルーチン命令が無視され、これに代えて、オーバレイメモリの対応アドレスのジャンプ命令が実行される。当該ジャンプ命令は、プロセッサに対し、これによって識別される作業メモリ内のアドレスへの分岐をトリガする。当該アドレス以降の領域には、上述したバイパスルーチンの処理をトリガするサービス機能呼び出しと、「重畳」プログラムルーチン命令の再構成とが存在しており、これにより、置換されるべきプログラムルーチンの全てのオリジナル命令が実行されることが保証される。ここで、サービス機能呼び出しは、再構成されるプログラムルーチン命令の前または後で書き込まれる。その後には、オリジナルプログラムルーチンへ戻る旨のジャンプ命令、すなわち、重畳されたアドレスへジャンプする旨のジャンプ命令が生じる。

【００４７】

このように、サービス機能呼び出しの挿入は、１回だけでなく複数回行うことができる。よって、例えば、場合により設けられる上書きルーチンに対してサービス機能呼び出しを実現することができる。

【００４８】

固体メモリ内の少なくとも１つのメモリアドレスをオーバレイメモリ内の少なくとも１つの対応アドレスへ重畳させてサービス機能呼び出しを統合するこうした手法は、固体メモリの操作が完全に禁止されるという利点を有する。固体メモリの操作禁止は、予め定められたプログラムルーチン部分の移し替えが行われる場合に要求される。

【００４９】

有利には、本発明の方法の別の実施形態において、まず制御装置でスイッチオンされているオーバレイ機能を遮断し、続いて任意の順序で、オーバレイメモリにジャンプ命令を記憶し、ジャンプ命令が指示する作業メモリに、サービス機能呼び出しと重畳命令の再構成とを統合することができる。これらが行われてから、制御装置のオーバレイ機能がスイッチオンされる。

【００５０】

本発明の別の実施形態によれば、バイパスルーチンが制御装置外のメモリ領域へ書き込まれるように構成できる。例えば、当該メモリ領域は、別の制御装置のメモリ、又は、他の制御装置をシミュレートするコンピュータのメモリであって、特にはラピッドコントロールプロトタイプシステムである。また、唯一のサービス機能呼び出ししか必要なく、これは、上述したように、メーカ側で置換されるべきプログラムルーチン内に既に構成されていてもよいし、又は、事後的に本発明の方法によって実現されてもよい。

【００５１】

後者のケースでは、サービス機能の独立変数が示しているテーブル内の位置の内容は、制御装置内のメモリでなく外部のメモリを指示しているので、制御装置又はプロセッサは、バイパスルーチンの実行に必要なデータを外部メモリの制御装置又は外部メモリを有するシミュレーションコンピュータに送信し、そこに記憶されているバイパスルーチンを実行するようにトリガされる。当該送信は、１つもしくは複数の制御装置とシミュレーションコンピュータとの間のインタフェース及び通信経路を介して行われる。その後、バイパスルーチンの結果が、特に当該結果によってオリジナルプログラムルーチンの結果を上書きするために、通信によってサービス機能呼び出しを実行する制御装置へ返送される。このために、返送されてくる結果は制御装置のテーブルへ入力され、特には上書きルーチン

をトリガするサービス機能によって、当該上書きルーチンを実行するために再び読み出される。

【実施例】

【００５２】

以下に、本発明の実施例を図に則して詳細に説明する。

【００５３】

図１には、サービスベースの内部機能バイパス化の基本方式が示されている。開発制御装置ＥＣＵには、固体メモリＳＰ１（一般的にはフラッシュメモリ）と、作業メモリＲＡＭとが設けられている。固体メモリＳＰ１には機械言語で符号化されたプログラムが存在しており、このプログラムは複数のプログラムルーチン、例えばルーチンｆ１，engine_idle_rev，ｆ３を含んでいる。ルーチンの個数はここでは図示の簡単化のために大幅に低減されている。実際の制御装置プログラムは典型的には数百の個別ルーチンから成る。

【００５４】

実際の適用例では、制御装置ＥＣＵは機関制御装置であり、ルーチンengine_idle_revはアイドリング回転数を制御するためのルーチンである。燃料消費量を低減するには、機関のアイドリング回転数をできるだけ低く保つことが有意である。言い換えれば、必要充分な回転数に機関動作を密に関連させて維持すべきである。

【００５５】

ただし、ここでの値は、機関モデルごとに設定された定数ではなく、種々の可変要素によって定められる可変値である。なお、ここでの可変要素には、機関モデルに加え、例えばバッテリの充電レベル、又は、電気負荷による瞬時バッテリ負荷、又は、機関温度などが含まれる。

【００５６】

多くの影響要因が定量化可能であってそれぞれ大きな重みを有するため、最適化できる可能性も大きい。プログラマであれば、アイドリング回転数を制御する機能の確実な手段が設けられることを所望するはずである。

【００５７】

１つのシナリオとして、提供者に機関制御装置を注文した自動車メーカが、当該提供者に対して、アイドリング回転数を制御する機能を試験したい旨を伝達するケースが考えられる。別のシナリオでは、制御装置の機能拡張が所望されるかもしれない。例えば、自動車が停止した場合の自動遮断機能を自動車モデルに設けたいというケースなどである。制御装置上に存在するアイドリング回転数の制御機能は、こうした機能を有していないので、ユーザは自分自身で置換を行いたいという要望を提供者に伝達する。

【００５８】

提供者側は、これに応じて、バイパスサービス機能ＺをフラッシュメモリＳＰ１の空きメモリ領域に格納し、場合により置換されるべき機能engine_idle_revの前後でサービスの呼び出しを実行する。

【００５９】

引き渡される独立変数ｉｄ１，ｉｄ２は、機能アドレステーブルＡＤにおける制御装置機能のメモリアドレスの位置を指示するポインタである。サービス機能Ｚは、引き渡されたポインタの指示しているテーブル位置がメモリアドレスのエントリを有しているか否かを検査する。エントリが存在している場合には、サービス機能は当該アドレスでｉｄ１のバイパスルーチン又はｉｄ２の上書きルーチンを待機し、これらを呼び出す。そうでない場合には、サービス機能Ｚは受動状態のままに留められる。

【００６０】

フラッシュメモリＳＰ１に記憶されている機能engine_idle_revを新たな機能eir_bypによって置換しようとするユーザは、当該新たな機能eir_bypを従来技術から周知の手法でフラッシュメモリＳＰ１の空き領域に格納し、そのアドレスを機能アドレステーブルＡＤの位置ｉｄ１へ入力する。また、ユーザは、上書き機能eir_byp_owrをフラッシュメモリＳＰ１の別の空き領域に格納し、そのアドレスを機能アドレステーブルＡＤの別の位置ｉ

ｄ２へ入力する。

【００６１】

バイパス機能eir_bypは固有のプログラムコードを含む。ただし、バイパス機能eir_bypは、自身が書き込んだ全ての値をまずバッファメモリＶＡＲへ格納する。ここでの値は、eir_byp以外の他の機能に関連しない定義されたグローバル変数のセットから成る。オリジナル機能engine_idle_revが適切に実行された後、eir_byp_owrは、engine_idle_revが書き込んだ変数を、eir_bypが作成したバッファメモリＶＡＲの値で上書きする。

【００６２】

上書き機能eir_byp_owrはバイパス機能eir_byp専用に作成されているので、当該上書き機能にとっては、バッファメモリＶＡＲのどの変数がプログラムコードのどの変数を上書きしなければならないかは既知である。フラッシュメモリＳＰ１上に１つだけでなく複数のバイパス機能がインストールされる場合、各バイパス機能はそれぞれ固有の上書き機能を受け取る。オリジナル機能engine_idle_revがどの変数を処理するかという情報、及び、そのメモリアドレスの情報は、制御装置に添付されたａ２ｌファイルから取り出される。

【００６３】

上書き機能を呼び出す第２のサービス呼び出しＺ（ｉｄ２）は、engine_idle_revにおいて全ての変数書き込みアクセスが行われた後、当該変数が別のフローで使用されるまでに行われなければならない。そうでないとプログラムフローの一貫性が保持されないからである。ここで、engine_idle_revのリターンによって何らの値も引き渡されない場合には、呼び出しを当該リターン指示の直後に行うことができる。それ以外の場合、呼び出しはリターン指示より前の機能の内部で行われなければならない。

【００６４】

上述したシナリオでは、オリジナル機能もバイパス機能も完全に実行される。ただし、制御装置ＥＣＵが例えば時間的にクリティカルな機能を実行する別のシナリオでは、機能engine_idle_revが完全に不活性化されることもありうる。このシナリオでは、バイパス機能がその値をバッファメモリＶＡＲへ書き込むのではなく、プログラムコードへ直接に書き込む。第２のサービス呼び出しは省略され、サービス機能Ｚは、アドレスｆ３又はこのアドレスｆ３が記憶されているテーブル位置を指示するポインタを、第２の独立変数として受け取る。なお、サービス機能Ｚは、バイパス機能の呼び出し後にｆ３へのジャンプを行うこともできる。

【００６５】

図２には、従来技術によるバイパスサービス機能の基本機能のフローチャートが示されている。呼び出しの際には、サービスに、機能アドレステーブルＡＤ内の位置を表す独立変数ｉｄｎが引き渡される。サービスは、当該位置において、所定のメモリアドレス又は空きエントリ乃至妥当でないアドレスを表す値を探索し、探索結果に応じて空きエントリをマークする。

【００６６】

エントリが空である場合、サービスを終了するリターン指示が直接に行われる。メモリアドレスが発見された場合には、当該位置で所定の機能が待機され、当該機能の呼び出しが実行される。適切な機能が当該アドレスにまさに存在することは、使用されるバイパスインプリメンテーションソフトウェアによって保証される。

【００６７】

図３には、仮想ａ２ｌファイルのセクションが示されている。このファイルは、制御装置ソフトウェアの技術的仕様を含んだ、開発制御装置の提供者データの一部である。図３には、機能の仕様を含むセクションの一部が示されている。この場合、機能engine_idle_revに対するバイパス機能を書き込もうとするプログラマは、当該機能が、変数var4，var13，var2を読み込み、かつ、変数var8，var2，var14，var11，var22を書き込み、かつ、設定値rcp_n_3568，scri_896，elec_m_cswによってパラメータ化されていることを読み出す。なお、ａ２ｌファイルの別のセクションには、上記各変数のメモリアドレスが挙げら

れている。
【００６８】
図４には、本発明にしたがって構成された制御装置で図１のバイパスシナリオが実行される様子が示されている。バイパス機能全体、すなわち、バイパスルーチンeir_byp及び上書きルーチンeir_byp_owrの双方が、フラッシュメモリＳＰ１ではなく作業メモリＲＡＭにロードされる。
【００６９】
作業メモリＲＡＭは揮発性メモリであるため、その内容は制御装置の遮断後には保持されない。このため、迅速かつ任意の頻度での書き込みが可能である。本発明によれば、制御装置を停止させて新たなプログラミングを行う必要なく、所望に応じてバイパスルーチンを交換もしくは拡張できる。
【００７０】
このために、有利な構成では、まず、機能アドレステーブルＡＤ内のエントリｉｄ１，ｉｄ２が消去される。この消去に応じて、サービス機能Ｚは、次の呼び出しによって開始されるまで、受動状態にとどまる。制御装置コードは、オリジナル機能engine_idle_revを考慮してプラン通りに処理される。バイパスルーチンは消去され、新たなルーチンによって置換される。新たなバイパスルーチンがロードされると、相応のアドレスが機能アドレステーブルＡＤ内の独立変数ｉｄ１，ｉｄ２が示す位置へ書き込まれる。
【００７１】
また、旧いバイパスルーチンを消去することなく、新たなバイパスルーチンを補充的にロードすることもできる。この場合、テーブルエントリの予めの消去は抑圧される。補充コードは有利には先行するバイパスルーチンの実行されない時間にインストールされるので、エントリは単純に上書きされる。これにより、全般的に書き込みが行われたことを確認できる。
【００７２】
作業メモリＲＡＭにコードがインストールされることにより、妥当性検査に付随する問題が発生しない。妥当性検査は固体メモリＳＰ１においてしか行われないからである。別の態様として、本発明の全ての実施例において、インタフェースを介して外部から書き込まれる制御変数により、バイパスコードを実行すべきか否かをサービス機能に伝達することもできる。このために、サービス機能は、制御変数の値を検査し、検査された条件乃至値が存在する場合にのみ、独立変数が指示するアドレスへのジャンプを行う。
【００７３】
本発明の方法によれば、既存のバイパスルーチンが実行期間中に消去乃至上書きされないことが必然的に保証される。
【００７４】
このために、図５では、サービス関数Ｚに制御機能が拡張されている。制御変数BypCodeActiveは、本明細書の従来技術の欄に言及した意味での仮想スイッチ又はステータスレジスタとして機能する。当該制御変数は、バイパスルーチンの実行前は１にセットされ、終了後に０にセットされる。使用されるバイパスインプリメンテーションソフトウェアは制御装置ＥＣＵのインタフェースを介して値BypCodeActiveを読み出し、この値が１である場合に、バイパスルーチンへのアクセスすなわち上書きを拒絶するように構成されている。
【００７５】
サービス機能は、上述した主機能のほか、他の機能を担当することもできる。例えば、制御装置ＥＣＵの遮断前に作業メモリＲＡＭに存在するコードを抹消することにより、制御装置の次回のスタート後に破損したコードが実行されないことを保証できる。
【００７６】
別の実施例では、サービス機能は、バイパスルーチンがダウンロード又は交換された後、遅くとも第１のバイパスルーチンの実行が要求されるまでに、新たなルーチンがダウンロードされたけれども未だ初期化されていないことを識別し、エントリｉｄ０によって構

成される機能アドレスを呼び出す。機能アドレスｉｄ０には、有利には、バイパスインプリメンテーションソフトウェアによってダイナミックにダウンロードされた機能が存在しており、この機能は、バイパスルーチンが参照している要素、例えば複数のグローバル変数の初期化を行う。したがって、サービス機能は、バイパスルーチンでの新たなセットの第１の呼び出しの前にまず初期化機能が実行されることを保証する。

【００７７】

上述した制御機能はどのようなシナリオにおいても充分でない。例えば、動作しているバイパスルーチンがより高い優先度を有する第２のプロセスによって中断されることがある。この場合、第２のプロセスも同様にバイパスルーチンを実行するので、実行後にはBypCodeActive=0がセットされるが、保持されているバイパスコードは未だ終了していない。このように、複数のプロセスが同時に複数のバイパスルーチンを実行するシナリオでは、いずれのプロセスにおいてもバイパスルーチンがアクティブとならないことが保証されなければならない。

【００７８】

この問題は、図６に示されている態様によって解決される。ここではBypCodeActiveはバイナリ値でない。つまり、この値はステータスレジスタではなく、バイパスルーチンが実行される前にアトミックに増分され、終了後にアトミックに減分される整数である。

【００７９】

理論的には各整数値はゼロ以上の値を取ることができ、この値によってどれだけの個数のバイパスルーチンが実際に処理されるかを表している。置換のためのバイパスルーチンへのアクセスは、BypCodeActiveがゼロである場合にのみ行われる。

【００８０】

図７には、サービス機能呼び出しがメーカ側又は供給者側で準備されていないケースにおいて、こうしたサービス機能呼び出しを後から機械コードに組み入れる実施例が示されている。

【００８１】

図７の左方には制御装置の固体メモリが示されている。ここには３つのルーチンfcn1, fcn2, fcn3から成るプログラムが記憶されている。また、固体メモリにはバイパスサービス機能Ｚが存在している。これは、提供者又はユーザ自身が固体メモリにインストールしたものである。図７の右方には、fcn2にサービス呼び出しが配置された後の同じ固体メモリが示されている。

【００８２】

ここでは、fcn2のコードの一部が所望のサービス呼び出しの位置から固体メモリの充分に大きな空きメモリ領域へ移されており、サービス呼び出しがその前方に書き込まれている。移動によって空いたfcn2のメモリ領域には、サービス呼び出しを指示するジャンプ命令が書き込まれている。さらに、別のジャンプ命令が移動されたfcn2部分の後方に書き込まれ、前方のプログラムフローへ戻っている。

【００８３】

バイパスサービス機能Ｚは、制御装置の提供者側で予めインストール可能である。ユーザは、上述した方法により、ソフトウェア内の所望の位置にバイパスサービス機能の呼び出しが構成されるように制御できる。ユーザがソフトウェアのソースコードにアクセスして提供者との協働作業を行う必要はない。

【００８４】

ふつう、サービス機能Ｚを呼び出すために後から構成されたコードは、制御装置に適合させなければならない。プログラム機能に独立変数を引き渡す具体的な様式は、プロセッサアーキテクチャや使用されるプログラミング言語又は使用されるコンパイラなどに依存して変化する。よって、本発明の方法の実行に用いられるバイパスインプリメンテーションソフトウェアは、例えば専用の仕様ファイルを介して各制御装置に関する情報を読み込み、新たに構成された機能呼び出しを相応に適合化する制御を行わなければならない。

【００８５】

本発明の方法の別の実施例では、実行されるソフトウェアが、移動されるコードのアドレスデータを移動前に検査し、必要に応じて移動の前又は後にこのアドレスデータを当該移動に合うように適合化する。当該適合化は、特に、移動されるコード以外のアドレスを指示する相対アドレスデータと、移動されるコード内のアドレスを指示する絶対アドレスデータとを必要とする。例えば、或るメモリ命令に対して「現在位置－２０」という位置指示が受け取られ、当該メモリ命令が移動されなかった場合、コードそのものを修正することなくコードの位置を変更することは許されない。これを行うとプログラムを正しく動作させることができなくなってしまうからである。ソフトウェアはこの場合には移動の実行を拒絶するか、又は、別の実施例として、所定の領域において移動可能な代替コードシーケンスを探索する。さらに別の実施例として、移動されるコード部分における相対アドレスデータを相応に適合化したり、又は、同じ機能を有する新たなコード乃至新たなコードシーケンスによって置換したりしてもよい。

【００８６】

選択的に、開発制御装置で利用可能な各機能を利用して、フラッシュメモリ上に存在する機械コードを操作することなく、必要なサービス機能呼び出しをプログラムコードに一時的に結合することもできる。このことは図８に示されている。

【００８７】

例えば、オーバレイメモリＯＶ、すなわち、多くの開制御装置に存在している、メモリ内容に他の内容を一時的に重畳させる装置を利用することができる。また、物理メモリアドレスを論理メモリアドレスに新たに割り当てるメモリマネジメントユニットＭＭＵも利用可能である。メモリマネジメントユニットＭＭＵは本発明ではオーバレイメモリと同様に利用できる。オーバレイメモリと同様、メモリマネジメントユニットＭＭＵは多数の開発制御装置でプロセッサ機能の一部となっている。

【００８８】

ここで、本発明の方法を実行する際には、機械コードの充分に長いコードシーケンスに、作業メモリＲＡＭの所定のアドレスを指示するジャンプ命令が重畳される。当該アドレスは、重畳されたコードの再構成とサービス機能呼び出しとを含むコードシーケンスの開始部を表しており、サービス機能呼び出しは再構成されたコードの前又は後に行われる。コードシーケンスはジャンプ命令で終了するが、このジャンプ命令は、重畳されたコードシーケンスに直接に後続する機械命令に続いている。サービス機能はバイパスルーチンと同様に作業メモリＲＡＭへ書き込まれる。

【００８９】

ただし、制御装置コードが周期的な妥当性検査を含む場合、この手法ではコードのチェックサムが変更される。

【００９０】

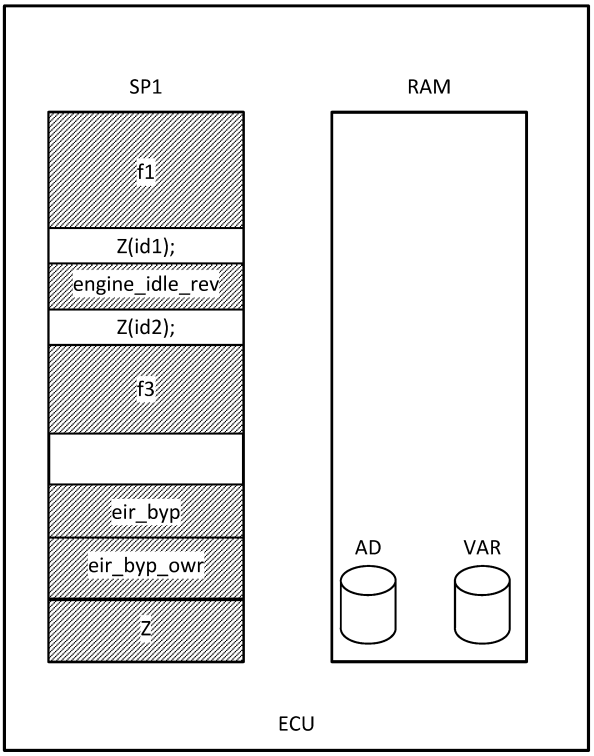
さらに、ウォッチポイントの利用も可能である。これは、多くの開発制御装置で利用可能な機能である（ただし１回に利用できる制御装置の数は限られる）。ブレイクポイントとは異なり、ウォッチポイントは、制御装置を単純に所定の開発制御装置モデル内の所望の位置にあるものと捉えず、代わりに設けられたメモリに書き込まれるコードを実行する。これにより、例えばユーザは、所定の条件（例えばif(x>4)break;）に停止を関連づけることができる。こうした監視コードはサービス呼び出しからも形成可能である。

【符号の説明】

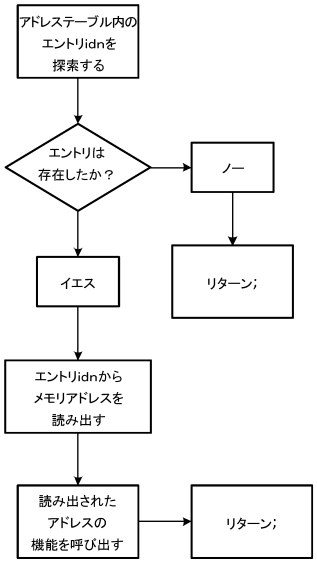
【００９１】

ＥＣＵ　開発制御装置、　ＳＰ１　固体メモリ、　ｆ１，engine_idle_rev，ｆ３　ルーチン、　Ｚ　バイパスサービス機能、　ｉｄ１，ｉｄ２　位置、　eir_byp　アドレス、　eir_byp_owr　上書きルーチン、　ＲＡＭ　作業メモリ、　ＡＤ　機能アドレステーブル、　ＶＡＲ　バッファメモリ

【図１】



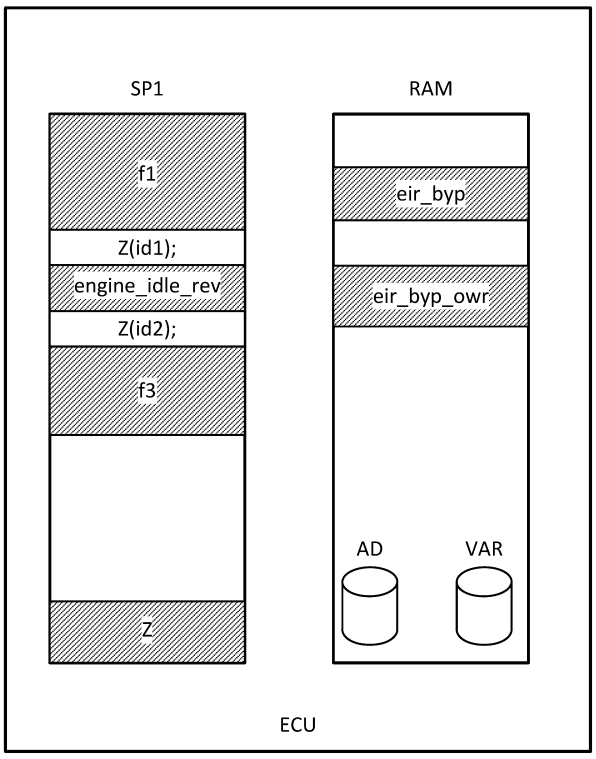
【図２】



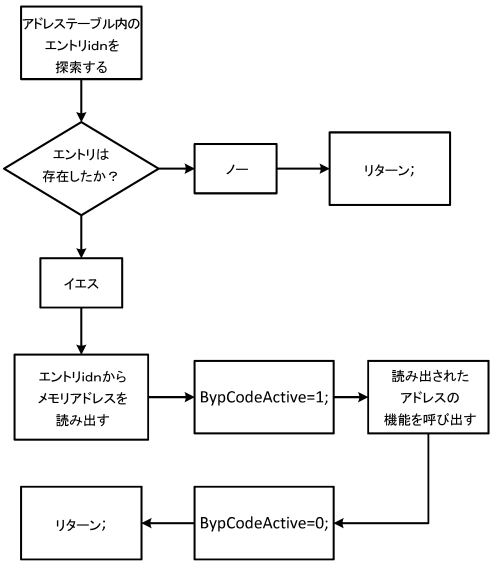
【図３】

```
\begin FUNCTION engine_idle_rev

        \begin IN_MEASUREMENT
                var4
                var13
                var2
        \end IN_MEASUREMENT

        \begin OUT_MEASUREMENT
                var8
                var2
                var14
                var11
                var22
        \end OUT_MEASUREMENT

        \begin DEF_CHARACTERISTIC
                rcp_n_3568
                scri_896
                elec_m_csw
        \end DEF_CHARACTERISTIC

\end FUNCTION

\begin FUNCTION engine_temp

        \begin IN_MEASUREMENT
                ...
```

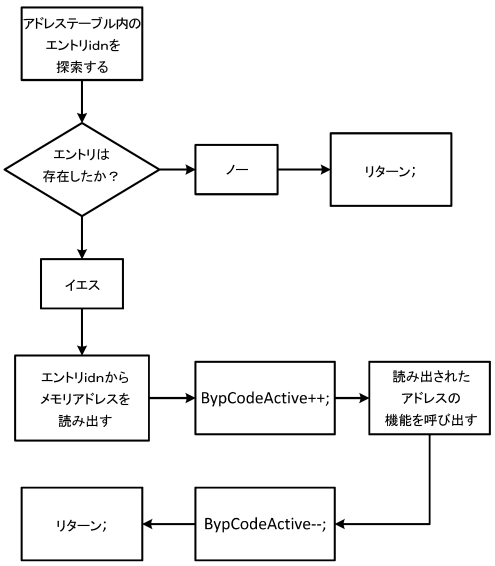
【図４】



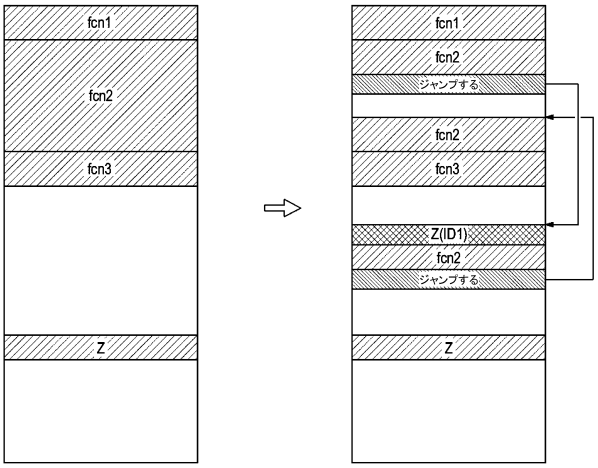
【図５】


アドレステーブル内の
エントリidnを
探索する
エントリは
存在したか？
ノー
リターン;
イエス
エントリidnから
メモリアドレスを
読み出す
BypCodeActive=1;
読み出された
アドレスの
機能を呼び出す
リターン;
BypCodeActive=0;

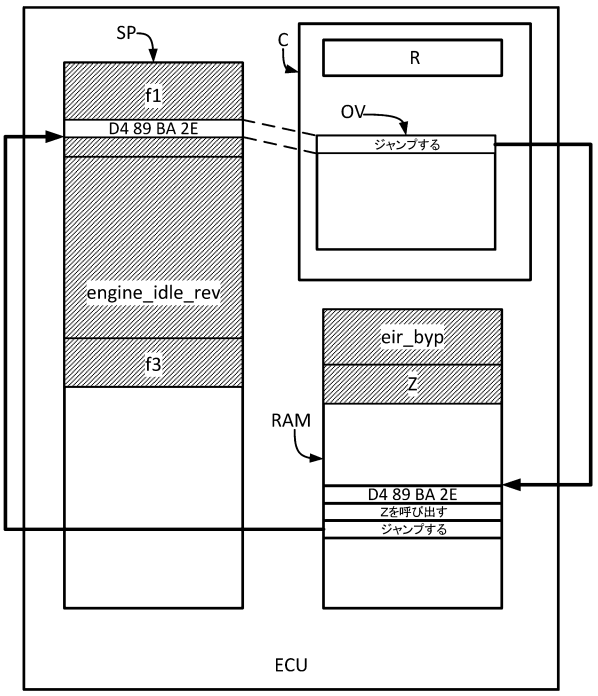
【図６】


アドレステーブル内の
エントリidnを
探索する
エントリは
存在したか？
ノー
リターン;
イエス
エントリidnから
メモリアドレスを
読み出す
BypCodeActive++;
読み出された
アドレスの
機能を呼び出す
リターン;
BypCodeActive--;

【図７】


fcn1
fcn2
fcn3
Z
fcn1
fcn2
ジャンプする
fcn2
fcn3
Z(ID1)
fcn2
ジャンプする
Z

【図８】


SP
C
R
f1
D4 89 BA 2E
OV
ジャンプする
engine_idle_rev
f3
eir_byp
Z
RAM
D4 89 BA 2E
Zを呼び出す
ジャンプする
ECU

フロントページの続き

(74)代理人 100114890
弁理士 アインゼル・フェリックス＝ラインハルト
(74)代理人 100099483
弁理士 久野 琢也
(72)発明者 トルステン フーフナーゲル
ドイツ連邦共和国 ザルツコッテン マリエンシュトラーセ ４４
(72)発明者 マーク ドレスラー
ドイツ連邦共和国 ホアン－バート・マインベルク バーンホーフシュトラーセ ２７
(72)発明者 バスティアン ケラース
ドイツ連邦共和国 パーダーボルン ゴルトレーゲンヴェーク ２１

審査官 坂庭 剛史

(56)参考文献 特開昭６１－０１６３０２（ＪＰ，Ａ）
特開平０７－１５２５５１（ＪＰ，Ａ）
特開平０８－０９５７７１（ＪＰ，Ａ）
特開平０８－１３７５１４（ＪＰ，Ａ）
米国特許第０６２６０１５７（ＵＳ，Ｂ１）
特開２００８－１０２７６１（ＪＰ，Ａ）
米国特許出願公開第２００８／０１４８２５０（ＵＳ，Ａ１）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ １１／００
Ｇ０６Ｆ ９／４４５
Ｂ６０Ｒ １６／０２