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

特許第6558317号
(P6558317)

(45)発行日　令和1年8月14日(2019.8.14)　(24)登録日　令和1年7月26日(2019.7.26)

(51)Int.Cl.		F I		
G06F 15/78	(2006.01)	G06F	15/78	516
G06F 9/48	(2006.01)	G06F	9/48	370
G06F 11/14	(2006.01)	G06F	11/14	612
G06F 15/177	(2006.01)	G06F	15/177	A
F02D 45/00	(2006.01)	F02D	45/00	372Z

請求項の数 4　(全 12 頁)

(21)出願番号　特願2016-137794(P2016-137794)
(22)出願日　平成28年7月12日(2016.7.12)
(65)公開番号　特開2018-10425(P2018-10425A)
(43)公開日　平成30年1月18日(2018.1.18)
審査請求日　平成30年10月1日(2018.10.1)

(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　100106149
弁理士　矢作　和行
(74)代理人　100121991
弁理士　野々部　泰平
(74)代理人　100145595
弁理士　久保　貴則
(72)発明者　徐　金旭
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　加藤　優一

最終頁に続く

(54)【発明の名称】電子装置

(57)【特許請求の範囲】
【請求項１】
複数の処理部を備えた電子装置であって、
各処理部は、
変数を演算する演算部（Ｓ１０、Ｓ２２）と、
自処理部における前記演算部での演算にエラーが発生したか否かを判定する判定部（Ｓ１１、Ｓ２３）と、
前記判定部にてエラーが発生したと判定された場合、エラーが発生したと判定された演算で得られた前記自処理部の変数を初期化する第１初期化部（Ｓ１４、Ｓ２６）と、を備え、
他処理部における前記演算部の演算で得られた変数値を保持しつつ、他処理部の演算した変数に関連する変数を、保持した変数値を用いて前記演算部が演算するものであり、
さらに、
前記演算部の演算が複数の前記処理部間で並行実行され、前記判定部にてエラーが発生したと判定された場合、エラーが発生したことを複数の前記処理部間で共有するために、エラーが発生したことを記憶するエラー記憶部（Ｓ１３、Ｓ２５）と、
前記エラー記憶部を確認することで、他処理部の前記演算部での演算にエラーが発生しているか否かを監視する監視部（Ｓ１５、Ｓ２７）と、
前記監視部での監視でエラーが発生していた場合、該エラーが発生している演算で得られた変数に関連する前記自処理部の変数を初期化する第２初期化部（Ｓ１６、Ｓ２８）と

、備えている電子装置。
【請求項２】
複数の前記処理部は、一つのマイコンに設けられたコアである請求項１に記載の電子装置。
【請求項３】
変数値が記憶された変数記憶部を備えており、
前記演算部は、前記変数記憶部に記憶された変数を演算するとともに、演算で得られた値に変数を更新し、
前記監視部は、前記自処理部における前記演算部の演算によって得られた値に変数を更新する前に前記エラー記憶部を確認し、
前記第２初期化部は、前記監視部での監視でエラーが発生していた場合、前記自処理部における前記演算部の演算によって得られた値に変数を更新する前に、前記変数記憶部に記憶された、前記自処理部の変数を初期化する請求項１又は２に記載の電子装置。
【請求項４】
前記第２初期化部は、他処理部の前記演算部での演算にエラーが発生していた場合、該エラーが発生している演算で得られた変数に関連する前記自処理部の変数を含む、前記自処理部における全ての変数を初期化する請求項１又は２に記載の電子装置。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、演算を実行する電子装置に関する。
【背景技術】
【０００２】
従来、演算を実行する電子装置の一例として、特許文献１に開示された車載制御装置がある。この車載制御装置は、非数が発生した変数を初期化するとともに、同変数を参照して算出される変数など、非数が発生した変数によって影響を受ける変数を一括して初期化する。
【先行技術文献】
【特許文献】
【０００３】
【特許文献１】特開２０１１－１６４８１４号公報
【発明の概要】
【発明が解決しようとする課題】
【０００４】
ところで、電子装置には、演算を並行実行する複数の処理部を備えたものがある。このような電子装置では、演算結果が非数になるなど演算にエラーが発生した変数と、この変数の影響を受ける変数とが、異なる処理部に配置されていることもある。また、処理部は、複数の処理部における他の処理部で変数が演算されると、その変数の値を保持し、その後、保持しておいた値を用いて他の変数の演算を行う。
【０００５】
よって、処理部は、他の処理部での演算にエラーが発生した場合、エラーが発生前の変数値を保持することがある。このため、処理部は、エラーが発生した演算で得られた変数が他の処理部によって初期化され、且つ、この変数の影響を受ける変数を初期化したとしても、保持しておいたエラーが発生前の演算で得られた変数値を用いて他の変数の演算を行なってしまう。従って、電子装置は、エラーが発生前の演算で得られた変数値を用いて演算された変数が初期化されないという問題がある。
【０００６】
本開示は、上記問題点に鑑みなされたものであり、変数の演算を並行実行する複数の処理部が、エラーが発生した演算で得られた変数に加えて、この変数を用いて演算された変数を初期化することができる電子装置を提供することを目的とする。

【課題を解決するための手段】
【０００７】
上記目的を達成するために本開示は、
複数の処理部を備えた電子装置であって、
各処理部は、
変数を演算する演算部（Ｓ１０、Ｓ２２）と、
自処理部における演算部での演算にエラーが発生したか否かを判定する判定部（Ｓ１１、Ｓ２３）と、
判定部にてエラーが発生したと判定された場合、エラーが発生したと判定された演算で得られた自処理部の変数を初期化する第１初期化部（Ｓ１４、Ｓ２６）と、を備え、
他処理部における演算部の演算で得られた変数値を保持しつつ、他処理部の演算した変数に関連する変数を、保持した変数値を用いて演算部が演算するものであり、
さらに、
演算部の演算が複数の処理部間で並行実行され、判定部にてエラーが発生したと判定された場合、エラーが発生したことを複数の処理部間で共有するために、エラーが発生したことを記憶するエラー記憶部（Ｓ１３、Ｓ２５）と、
エラー記憶部を確認することで、他処理部の演算部での演算にエラーが発生しているか否かを監視する監視部（Ｓ１５、Ｓ２７）と、
監視部での監視でエラーが発生していた場合、該エラーが発生している演算で得られた変数に関連する自処理部の変数を初期化する第２初期化部（Ｓ１６、Ｓ２８）と、備えていることを特徴とする。
【０００８】
このように、本開示は、自処理部における演算部での演算にエラーが発生したか否かを判定し、エラーが発生したと判定した場合、エラーが発生したと判定された演算で得られた自処理部の変数を初期化する。また、本開示は、エラーが発生したと判定された場合、エラーが発生したことを複数の処理部間で共有するために、エラーが発生したことを記憶部に記憶し、この記憶部を確認して、他処理部の演算部での演算にエラーが発生しているか否かを監視する。
【０００９】
このため、本開示の各処理部は、自処理部における演算部での演算で用いる変数が、エラーが発生している演算で得られた変数であるか否かを把握することができる。そして、本開示は、監視部で監視してエラーが発生していた場合、エラーが発生している演算で得られた変数に関連する自処理部の変数を初期化する。よって、本開示は、変数の演算を実行する複数の処理部が、エラーが発生した演算で得られた変数に加えて、この変数を用いて演算された変数を初期化することができる。
【００１０】
なお、特許請求の範囲、及びこの項に記載した括弧内の符号は、一つの態様として後述する実施形態に記載の具体的手段との対応関係を示すものであって、発明の技術的範囲を限定するものではない。
【図面の簡単な説明】
【００１１】
【図１】実施形態におけるマイコンの概略構成を示すブロック図である。
【図２】実施形態におけるマイコンの処理動作を示すシーケンス図である。
【図３】実施形態における第１コアの処理動作を示すフローチャートである。
【図４】実施形態における第２コアの処理動作を示すフローチャートである。
【発明を実施するための形態】
【００１２】
以下において、図面を参照しながら、発明を実施するための形態を説明する。図１に示すように、ＥＣＵ２００は、マイクロコンピュータ１００が設けられている。なお、ＥＣＵ２００は、マイコン１００の他にも、各種電子素子が設けられていてもよい。

【００１３】
マイクロコンピュータ１００は、電子装置に相当する。以下においては、マイクロコンピュータ１００をマイコン１００と略称で記載する。マイコン１００は、第１コア１０、第１コア用ＲＡＭ１２、第２コア２０、第２コア用ＲＡＭ２２、共有ＲＡＭ３０などを備えて構成されている。このように、マイコン１００は、所謂デュアルコアマイコンである。しかしながら、本発明は、三つ以上のコアを含むマルチコアマイコンでも採用できる。
【００１４】
第１コア１０と第２コア２０のそれぞれは、処理部に相当する。各コア１０，２０は、予め定められた演算順序に従って予め定められたプログラムを実行する。また、各コア１０，２０は、浮動小数点演算を実行可能に構成されている。なお、コアは、演算プロセッサやＣＰＵコアと言い換えることもできる。各コア１０，２０の処理動作に関しては、後程詳しく説明する。
【００１５】
第１コア１０は、第１コアフラグが記憶された記憶領域である第１フラグ領域１１を含んでいる。第１コアフラグは、第１コア１０によって値が設定されるフラグである。また、第１コアフラグは、第１コア１０による演算にエラーが発生したか否か、すなわち第１コア１０の演算状態を示すフラグである。よって、第１コアフラグは、第１コア１０による演算にエラーが発生したか否かによって値が書き換えられる。この第１コアフラグは、第１コア１０のみが値を設定できる。このため、第２コア２０は、第１コアフラグの値を設定できない。
【００１６】
なお、第１コア１０は、自コア１０の演算でエラーが発生したと判定した場合、第１コアフラグをオンし、エラーが発生したと判定していない場合、第１コアフラグをオフすると言える。また、第１コア１０は、第１コアフラグをオンすることでエラー状態を書き込み、第１コアフラグをオフすることで非エラー状態を書き込むと言える、
一方、第２コア２０は、第２コアフラグが記憶された記憶領域である第２フラグ領域２１を含んでいる。第２コアフラグは、第２コア２０によって値が設定されるフラグである。また、第２コアフラグは、第２コア２０による演算にエラーが発生したか否か、すなわち第２コア２０の演算状態を示すフラグである。よって、第２コアフラグは、第２コア２０による演算にエラーが発生したか否かによって値が書き換えられる。この第２コアフラグは、第２コア２０のみが値を設定できる。このため、第１コア１０は、第２コアフラグの値を設定できない。
【００１７】
なお、第２コア２０は、自コア２０の演算でエラーが発生したと判定した場合、第２コアフラグをオンし、エラーが発生したと判定していない場合、第２コアフラグをオフすると言える。また、第２コア２０は、第２コアフラグをオンすることでエラー状態を書き込み、第２コアフラグをオフすることで非エラー状態を書き込むと言える。
【００１８】
このように、各コア１０，２０は、自コアの演算にエラーが発生したことを記憶可能に構成されている。なお、各コアフラグの設定に関しては、後程、各コア１０，２０の処理動作とともに説明する。
【００１９】
第１コア用ＲＡＭ１２（以下、第１ＲＡＭ１２）は、第１コア１０専用の記憶装置である。第１ＲＡＭ１２には、第１コア１０のみが用いる変数などが記憶されている。一方、第２コア用ＲＡＭ２２（以下、第２ＲＡＭ２２）は、第２コア２０専用の記憶装置である。
【００２０】
また、第２ＲＡＭ２２は、記憶領域として変数領域２２ａを含んでおり、第２コア２０のみが用いる変数などが記憶されている。例えば、変数領域２２ａには、第２コア２０のみが用いる変数ｄが記憶されている。変数ｄは、第２コア２０によって演算されて、値が

書き換えられる。変数領域２２ａは、変数記憶部に相当する。

【００２１】

共有ＲＡＭ３０は、第１コア１０と第２コア２０に共通に設けられた記憶装置である。共有ＲＡＭ３０は、記憶領域として、第１共有領域３１、第２共有領域３２、第３共有領域３３を含んでいる。また、第１共有領域３１には、第１コア１０によって値が設定される第１フラグが記憶されている。第２共有領域３２には、第２コア２０によって値が設定される第２フラグが記憶されている。よって、第１コア１０は、第２フラグの値を参照することはできる。同様に、第２コア２０は、第１フラグの値を参照することはできる。

【００２２】

なお、第１フラグは、第１コアフラグと同様に、第１コア１０による演算にエラーが発生したか否かを示すフラグである。よって、第１フラグの値は、第１コアフラグと同じ第１コア１０の演算状態を示している。このため、第２コア２０は、第１フラグの値を参照することで、第１コア１０の演算状態を認識できる。

【００２３】

一方、第２フラグは、第２コアフラグと同様に、第２コア２０による演算にエラーが発生したか否かを示すフラグである。よって、第２フラグの値は、第２コアフラグと同じ第２コア２０の演算状態を示している。このため、第１コア１０は、第２フラグの値を参照することで、第２コア２０の演算状態を認識できる。

【００２４】

なお、第１コア１０は、自コア１０の演算でエラーが発生したと判定した場合、第１フラグをオンし、エラーが発生したと判定していない場合、第１フラグをオフすると言える。同様に、第２コア２０は、自コア２０の演算でエラーが発生したと判定した場合、第２フラグをオンし、エラーが発生したと判定していない場合、第２フラグをオフすると言える。

【００２５】

また、第３共有領域３３には、第１コア１０と第２コア２０の両方が用いる変数が記憶されている。例えば、第３共有領域３３には、第１コア１０と第２コア２０が用いる変数ｃが記憶されている。変数ｃは、第１コア１０によって演算されて、値が書き換えられる。

【００２６】

ところで、各コア１０，２０が実行する浮動小数点演算は、浮動小数点型データを用いた演算である。演算結果の浮動小数点型データは、第１ＲＡＭ１２や、第２ＲＡＭ２２や、共有ＲＡＭ３０に記憶される。この演算結果の浮動小数点型データは、変数に相当する。よって、上記変数ｃと変数ｄは、浮動小数点型データである。

【００２７】

変数ｃは、第１コア１０が浮動小数点演算を行うことで得られる。つまり、第１コア１０は、第３共有領域３３に記憶された変数ｃを演算するとともに、演算で得られた値に変数ｃを更新する。よって、第１コア１０は、第３共有領域３３の変数ｃを演算結果の値で書き換えると言える。

【００２８】

一方、変数ｄは、第２コア２０が浮動小数点演算を行うことで得られる。つまり、第２コア２０は、変数領域２２ａに記憶された変数ｄを演算するとともに、演算で得られた値に変数ｄを更新する。よって、第２コア２０は、変数領域２２ａの変数ｄを演算結果の値で書き換えると言える。さらに、第２コア２０は、変数ｃの値を保持しつつ、保持した値を用いて変数ｄを演算する。このため、変数ｃは、他処理部における演算部の演算で得られた変数に相当する。変数ｄは、保持した変数に関連する変数に相当する。また、変数ｃと変数ｄは、関連する変数と言える。

【００２９】

このように、変数ｃは、第１コア１０と第２コア２０の両方で用いられるため、共有ＲＡＭ３０の第３共有領域３３に記憶されている。一方、変数ｄは、第１コア１０で用いら

れず、共有ＲＡＭ３０に記憶されている必要がないため、第２ＲＡＭ２２に記憶されている。

【００３０】

各コア１０，２０は、例えば、ＩＥＥＥ７５４規格に従ったデータ形式で、浮動小数点型データを表す。ＩＥＥＥ７５４規格に従う単精度の浮動小数点型データは、１ビットの符号部、８ビットの指数部、２３ビットの仮数部で構成される。

【００３１】

この規格の浮動小数点データでは、例えば、数値として表現できない演算結果を非数として表現できるようになっている。０で除する演算や、∞（無限大）を用いた演算等の演算結果が、数値として表現できない演算結果の一例として挙げられる。

【００３２】

例えばこの規格では、指数部の８ビットが全て１で、すなわち指数部が１０進表記の２５５で、仮数部が０以外の場合、この浮動小数点型データを非数と表す。また、この規格では、指数部の８ビットが全て１で仮数部が０の場合、この浮動小数点型データを無限大と表す。なお、０で除する演算や、無限大を用いた演算のような、数値として表現できない演算結果となる演算は、無効な演算と言える。

【００３３】

また、各コア１０，２０で演算したデータが非数か無限大の場合に、各コア１０，２０での演算にエラーが発生したと言うことができる。さらに、各コア１０，２０での演算が無効な演算であった場合に、各コア１０，２０での演算にエラーが発生したと言うことができる。

【００３４】

なお、以下においては、演算にエラーが発生した一例として、各コア１０，２０で演算した浮動小数点型データが非数であった場合を採用する。よって、第１コアフラグ及び第１フラグは、第１コア１０の演算で得られた変数が非数であるか否かを示しているものとする。一方、第２コアフラグ及び第２フラグは、第２コア２０の演算で得られた変数が非数であるか否かを示しているものとする。

【００３５】

ここで、図２、図３、図４を用いて、第１コア１０と第２コア２０の処理動作に関して説明する。第１コア１０と第２コア２０とは、図２に示すように演算を並行実行する。本実施形態では、一例として、第１コア１０で非数が発生し、第２コア２０で非数が発生しない場合を採用する。

【００３６】

まず、図３を用いて第１コア１０の処理動作に関して説明する。第１コア１０は、所定の実行タイミングで図３のフローチャートをスタートする。

【００３７】

ステップＳ１０では、変数ｃを演算する（演算部）。第１コア１０は、処理を開始すると、変数ｃを演算する。

【００３８】

ステップＳ１１では、非数が発生したか否かを判定する（判定部）。第１コア１０は、ステップＳ１０での演算で非数が発生したか否かを判定する。言い換えると、第１コア１０は、自コア１０における演算にエラーが発生したか否かを判定する。第１コア１０は、例えばｃ＝９／０の場合、非数が発生する。

【００３９】

第１コア１０は、ステップＳ１１で非数が発生したと判定した場合、ステップＳ１２へ進む。ステップＳ１２では、第２コア２０に非数状態を通知する。第１コア１０は、ステップＳ１１で非数が発生したと判定した場合、第２コア２０に対して、自コア１０で非数が発生したことを通知する。なお、本発明は、ステップＳ１２を省略しても目的を達成できる。

【００４０】

ステップＳ１３では、第１フラグをオンする（エラー記憶部）。第１コア１０は、ステップＳ１１で非数が発生したと判定した場合、第１共有領域３１の第１フラグをオンする。言い換えると、第１コア１０は、非数状態を第１共有領域３１に書き込む。

【００４１】

このように、第１コア１０は、第２コア２０と演算を並行実行しており、非数が発生したと判定した場合、非数が発生したことを第２コア２０との間で共有するために、非数が発生したことを共有ＲＡＭ３０に記憶すると言える。また、第１コア１０は、第１共有領域３１の第１フラグをオンすることで、演算を並行実行している第２コア２０との間で、自コア１０で非数が発生したことを共有すると言える。なお、第１コア１０は、ステップＳ１１で非数が発生したと判定した場合、第１コアフラグをオンするとともに、第１共有領域３１の第１フラグをオンする。

【００４２】

ステップＳ１４では、変数ｃを初期化する（第１初期化部）。第１コア１０は、非数が発生した変数ｃを初期化する。つまり、第１コア１０は、非数が発生したと判定した場合、非数が発生したと判定された演算で得られた自コア１０の変数ｃを初期化する。

【００４３】

第１コア１０は、ステップＳ１１で非数が発生したと判定していない場合、ステップＳ１５へ進む。ステップＳ１５では、第２フラグがオンであるか否かを判定する（監視部）。第１コア１０は、第２共有領域３２を確認することで、すなわち、第２フラグを確認することで、他処理部である第２コア２０での演算で非数が発生しているか否かを監視する。

【００４４】

第１コア１０は、第２フラグがオンであると判定した場合、ステップＳ１６へ進み、第２フラグがオンであると判定しなかった場合、ステップＳ１７へ進む。なお、後程説明するが、第２コア２０は、第１コア１０と同様に、自コア２０での演算で非数が発生した場合、第２フラグをオンする。

【００４５】

ステップＳ１６では、変数ｃを初期化する（第２初期化部）。第１コア１０は、ステップＳ１５で第２フラグがオンであり、第２コア２０での演算で非数が発生していたと判定した場合、非数が発生している変数ｄに関連する変数ｃを初期化する。第１コア１０は、例えばｃ＝０とする。第１コア１０は、例えばプログラムの内容などによって、変数ｃが変数ｄに関連するか否かを確認できる。なお、第１コア１０は、変数ｃが変数ｄに関連しないと判定した場合、ステップＳ１６を行う必要がない。

【００４６】

ステップＳ１７では、変数ｃに演算結果を書き込む。第１コア１０は、ステップＳ１０での演算結果を第３共有領域３３に書き込む。

【００４７】

次に、図４を用いて第２コア２０の処理動作に関して説明する。第２コア２０は、所定の実行タイミングで図４のフローチャートをスタートする。

【００４８】

ステップＳ２０では、変数ｃをラッチする。第２コア２０は、第３共有領域３３に記憶されている変数ｃを読み込んで、第２ＲＡＭ２２に記憶（保持）しておく。

【００４９】

ステップＳ２１では、通知により変数ｄを初期化する。第２コア２０は、第１コア１０から非数状態の通知がなされた場合、変数ｄを初期化する。第２コア２０は、例えばｄ＝０とする。当然ながら、第２コア２０は、第１コア１０から非数状態の通知がなかった場合、変数ｄの初期化を行わない。

【００５０】

ステップＳ２２では、変数ｄを演算する（演算部）。第２コア２０は、ステップＳ２０でラッチした変数ｃを用いて変数ｄを演算する。このように、第２コア２０は、第１コア

１０における演算で得られた変数ｃを保持しつつ、保持した変数ｃに関連する変数ｄを、保持した変数ｃを用いて演算する。

【００５１】

ステップＳ２３では、非数が発生したか否かを判定する（判定部）。第２コア２０は、自コア２０における演算にエラーが発生したか否かを判定する。第２コア２０は、ステップＳ２３で非数が発生したと判定した場合、ステップＳ２４へ進む。

【００５２】

ステップＳ２４では、第１コア１０に非数情報を通知する。なお、第２コア２０は、ステップＳ２４を省略しても目的を達成できる。

【００５３】

ステップＳ２５では、第２フラグをオンする（エラー記憶部）。第２コア２０は、ステップＳ２３で非数が発生したと判定した場合、第２共有領域３２の第２フラグをオンする。言い換えると、第２コア２０は、非数状態を第２共有領域３２に書き込む。

【００５４】

このように、第２コア２０は、第１コア１０と演算を並行実行しており、非数が発生したと判定した場合、非数が発生したことを第１コア１０との間で共有するために、非数が発生したことを共有ＲＡＭ３０に記憶すると言える。また、第２コア２０は、第２共有領域３２の第２フラグをオンすることで、演算を並行実行している第１コア１０との間で、自コア２０で非数が発生したことを共有すると言える。なお、第２コア２０は、ステップＳ２３で非数が発生したと判定した場合、第２コアフラグをオンするとともに、第２共有領域３２の第２フラグをオンする。

【００５５】

ステップＳ２６では、変数ｄを初期化する（第１初期化部）。第２コア２０は、非数が発生した変数ｄを初期化する。つまり、第２コア２０は、非数が発生したと判定した場合、非数が発生したと判定された演算で得られた自コア２０の変数ｄを初期化する。

【００５６】

第２コア２０は、ステップＳ２３で非数が発生したと判定していない場合、ステップＳ２７へ進む。ステップＳ２７では、第１フラグがオンであるか否かを判定する（監視部）。第２コア２０は、第１共有領域３１を確認することで、すなわち、第１フラグを確認することで、他処理部である第１コア１０での演算で非数が発生しているか否かを監視する。第２コア２０は、第１フラグがオンであると判定した場合、ステップＳ２８へ進み、第１フラグがオンであると判定しなかった場合、ステップＳ２９へ進む。

【００５７】

ステップＳ２８では、変数ｄを初期化する（第２初期化部）。第２コア２０は、ステップＳ２７で第１フラグがオンであり、第１コア１０での演算で非数が発生していたと判定した場合、非数が発生している変数ｃに関連する変数ｄを初期化する。つまり、第２コア２０は、他コア１０が演算した変数ｃに非数が発生していた場合、自コア２０が変数ｃを用いて演算する変数ｄを初期化する。第２コア２０は、例えばｄ＝０とする。第２コア２０は、例えばプログラムの内容などによって、変数ｄが変数ｃに関連するか否かを確認できる。

【００５８】

なお、第２コア２０は、変数ｄとは異なる変数に関しても、変数ｃを用いて演算を行うものであってもよい。この第２コア２０が変数ｃを用いて演算を行う変数は、第２ＲＡＭ２２や共有ＲＡＭ３０に記憶されている。この場合、ステップＳ２８では、第２コア２０は、変数ｄだけでなく、変数ｃを用いて演算を行う全ての変数を初期化してもよい。

【００５９】

また、第２コア２０は、ステップＳ２２で変数ｄを演算した場合、ステップＳ２７よりも先に、変数領域２２ａの変数ｄを演算によって得られた値に更新してもよい。しかしながら、本実施形態では、ステップＳ２７で第１フラグがオンであると判定した場合、変数領域２２ａの変数ｄを演算によって得られた値に更新する前に、変数領域２２ａの変数ｄ

を初期化する。これによって、マイコン１００は、変数領域２２ａの変数ｄが二度更新されることを抑制できる。また、マイコン１００は、変数領域２２ａの変数ｄが、非数が発生した変数ｃを用いて演算された値に更新されることを抑制できる。

【００６０】

なお、各コア１０，２０で演算された変数ｃ，ｄは、各種制御に用いられるものであってもよい。この場合、マイコン１００は、変数領域２２ａの変数ｄが、非数が発生した変数ｃを用いて演算された値に更新されることを抑制できるため、変数ｄを用いる制御の信頼性を向上できる。

【００６１】

ステップＳ２９では、変数ｄに演算結果を書き込む。第２コア２０は、ステップＳ２２での演算結果を変数領域２２ａに書き込む。

【００６２】

次に、図２のシーケンス図を用いて、第１コア１０と第２コア２０の処理動作を説明する。タイミングｔ１では、第１コア１０と第２コア２０とが処理を開始する。よって、第１コアは、ステップＳ１０の処理を実行し、第２コア２０がステップＳ２０を実行する。

【００６３】

タイミングｔ２では、第１コア１０がステップＳ１１、Ｓ１２を実行し、第２コア２０が非数状態の通知を受けてステップＳ２１を実行する。タイミングｔ３では、第１コア１０がステップＳ１３を実行し、第２コア２０がステップＳ２２を実行する。このとき、第２コア２０は、ステップＳ２０でラッチした変数ｃを用いて、変数ｄを演算することになる。

【００６４】

タイミングｔ４では、第１コア１０がステップＳ１４を実行し、第２コア２０がステップＳ２３を実行する。タイミングｔ５では、第２コア２０がステップＳ２７を実行する。そして、タイミングｔ６では、ステップＳ２８を実行する。なお、タイミングｔ６では、第１コア１０はアイドル状態となる。

【００６５】

このように、マイコン１００は、第１コア１０における演算に非数が発生したか否かを判定し、非数が発生したと判定した場合、非数が発生したと判定された演算で得られた自コア１０の変数ｃを初期化する。同様に、マイコン１００は、第２コア２０における演算に非数が発生したか否かを判定し、非数が発生したと判定した場合、非数が発生したと判定された演算で得られた自コア２０の変数ｄを初期化する。

【００６６】

また、マイコン１００は、非数が発生したと判定された場合、非数が発生したことを第１コア１０と第２コア２０との間で共有するために、非数が発生したことを共有ＲＡＭ３０に記憶する。そして、マイコン１００は、各コア１０，２０が共有ＲＡＭ３０を確認して、他コアでの演算に非数が発生しているか否かを監視する。

【００６７】

このため、各コア１０，２０は、自コアにおける演算で用いる変数が、非数が発生している演算で得られた変数であるか否かを把握することができる。そして、マイコン１００は、上記のように監視して非数が発生していた場合、非数が発生している演算で得られた変数ｃに関連する自コア２０の変数ｄを初期化する。よって、マイコン１００は、変数の演算を実行する複数のコア１０，２０が、非数が発生した演算で得られた変数ｃに加えて、この変数ｃを用いて演算された変数ｄを初期化することができる。つまり、マイコン１００は、他コアの非数発生に応じて、関連する自コアの変数を一括初期化できるとも言える。

【００６８】

また、各コア１０，２０で演算された変数ｃ，ｄは、各種制御に用いられるものであってもよい。上記のように、マイコン１００は、非数が発生した変数ｃを初期化できるだけでなく、他コア２０が変数ｃを用いて演算した変数ｄを初期化できる。このため、マイコ

ン１００は、非数が発生した変数ｃを用いて演算された変数ｄが初期化されないまま用いられるよりも、変数ｄを用いる制御の信頼性を向上できる。

【００６９】

なお、本実施形態では、処理部としてのコア１０，２０を備えたデュアルコアマイコンに適用した例を採用した。しかしながら、本発明はこれに限定されない。本発明は、処理部としてのマイコンを複数備えた電子装置であっても適用できる。

【００７０】

また、第２ＲＡＭ２２には、変数ｄの他にも、変数ｃを用いて演算されない複数の変数を記憶している。第２コア２０は、ステップＳ２８で変数ｄを初期化する際に、変数ｄだけではなく、変数ｃを用いて演算されない他の変数も含めて全ての変数を初期化してもよい。つまり、第２コア２０は、ステップＳ２８で第２ＲＡＭ２２に記憶されている全ての変数を初期化してもよい。この場合、第２コア２０は、変数ｃを用いて演算する変数ｄを第２ＲＡＭ２２から選択する必要がない。よって、マイコン１００は、変数ｃを用いて演算する変数を第２ＲＡＭ２２から選択するよりも容易に変数ｄを初期化することができる。

【００７１】

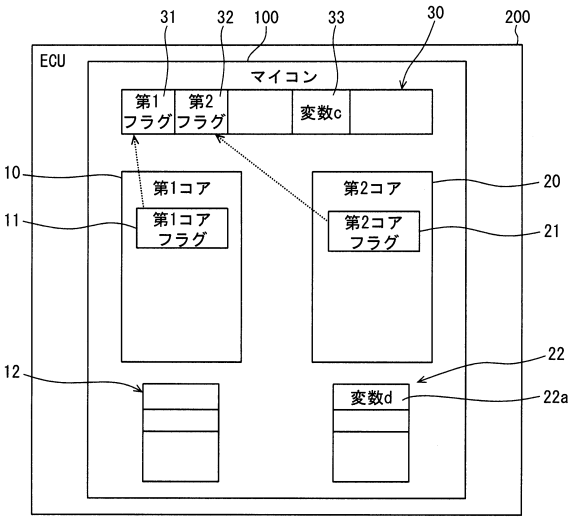
以上、本発明の好ましい実施形態について説明した。しかしながら、本発明は、上記実施形態に何ら制限されることはなく、本発明の趣旨を逸脱しない範囲において、種々の変形が可能である。

【符号の説明】

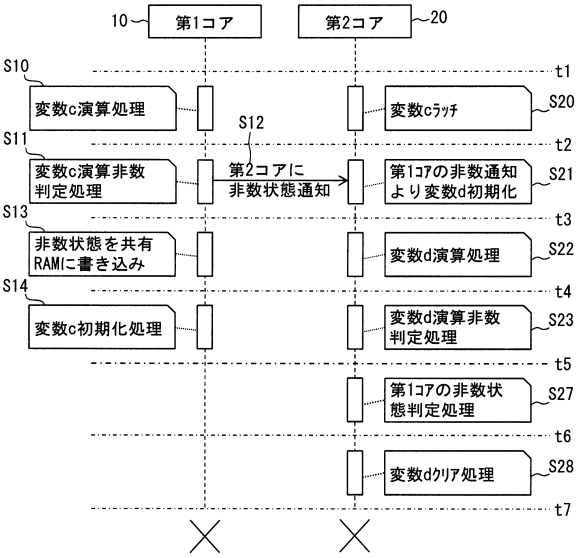
【００７２】

１０…第１コア、１１…第１フラグ領域、１２…第１コア用ＲＡＭ、２０…第２コア、２１…第２フラグ領域、２２…第２コア用ＲＡＭ、２２ａ…変数領域、３０…共有ＲＡＭ、３１…第１共有領域、３２…第２共有領域、３３…第３共有領域、１００…マイコン、２００…ＥＣＵ

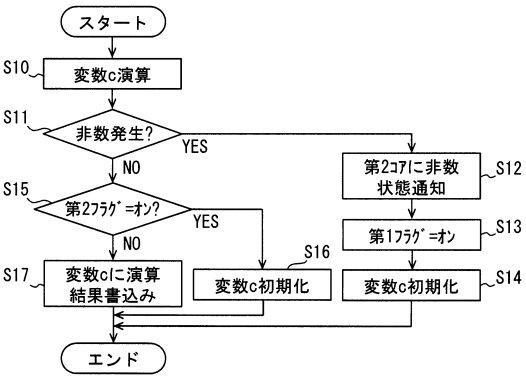
【図１】


ECU
マイコン
第1
フラグ
第2
フラグ
変数c
第1コア
第1コア
フラグ
第2コア
第2コア
フラグ
変数d
10
11
12
20
21
22
22a
30
31
32
33
100
200

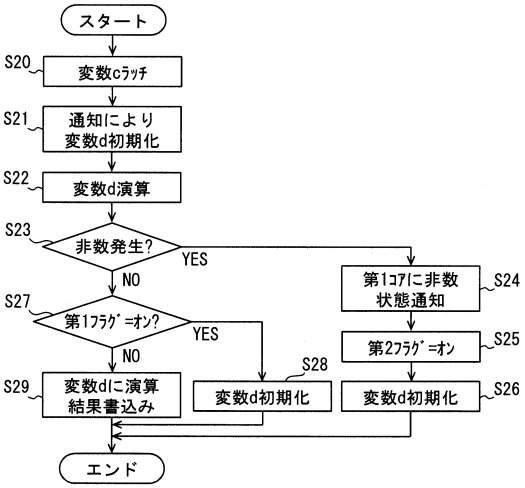
【図２】


10
第1コア
第2コア
20
S10
変数c演算処理
S11
変数c演算非数
判定処理
S12
第2コアに
非数状態通知
S13
非数状態を共有
RAMに書き込み
S14
変数c初期化処理
変数cﾗｯﾁ
S20
第1ｺｱの非数通知
より変数d初期化
S21
変数d演算処理
S22
変数d演算非数
判定処理
S23
第1ｺｱの非数状
態判定処理
S27
変数dｸﾘｱ処理
S28
t1
t2
t3
t4
t5
t6
t7

【図３】


スタート
S10
変数c演算
S11
非数発生?
YES
NO
S12
第2ｺｱに非数
状態通知
S15
第2ﾌﾗｸﾞ=ｵﾝ?
YES
NO
S13
第1ﾌﾗｸﾞ=ｵﾝ
S16
S17
変数cに演算
結果書込み
変数c初期化
S14
変数c初期化
エンド

【図４】


スタート
S20
変数cﾗｯﾁ
S21
通知により
変数d初期化
S22
変数d演算
S23
非数発生?
YES
NO
S24
第1ｺｱに非数
状態通知
S27
第1ﾌﾗｸﾞ=ｵﾝ?
YES
NO
S25
第2ﾌﾗｸﾞ=ｵﾝ
S28
S29
変数dに演算
結果書込み
変数d初期化
S26
変数d初期化
エンド

フロントページの続き

(56)参考文献 特開２０１１－１６４８１４（ＪＰ，Ａ）
特開２００４－２７８４８３（ＪＰ，Ａ）
特開２０１０－０６１３２４（ＪＰ，Ａ）
特開２００６－３１８４０１（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　１５／７８
Ｇ０６Ｆ　　７／４８３
Ｇ０６Ｆ　　９／４８
Ｇ０６Ｆ　１１／１４
Ｇ０６Ｆ　１５／１７７
Ｂ６０Ｒ　１６／０２
Ｆ０２Ｄ　４５／００