(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)特許出願公開番号
特開2008-9869
(P2008-9869A)
(43)公開日　平成20年1月17日(2008.1.17)

(51)Int.Cl.		F I			テーマコード（参考）
G05B 19/042	(2006.01)	G05B	19/042		5B176
G06F 9/44	(2006.01)	G06F	9/06	620A	5H220

審査請求 未請求　請求項の数 11　ＯＬ　（全 33 頁）

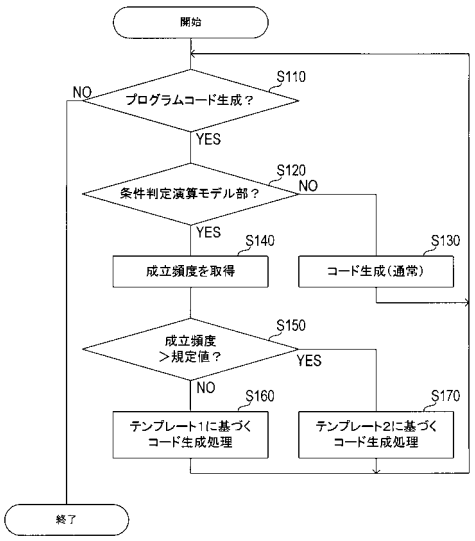
(21)出願番号　特願2006-181650 (P2006-181650)
(22)出願日　平成18年6月30日 (2006.6.30)

(71)出願人　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　植松　義貴
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　荻野　哲也
愛知県刈谷市昭和町１丁目１番地　株式会社デンソー内

Ｆターム(参考)　5B176 DD02
5H220 AA01　AA04　BB12　CC07　CX02
JJ12　JJ47　KK01

(54)【発明の名称】プログラムコード生成装置

(57)【要約】
【課題】電子制御装置での制御プログラムの実行時間や処理負荷が低減できる制御プログラムのプログラムコードを、制御仕様を表す制御モデルから容易に生成できるプログラムコード生成装置を提供する。
【解決手段】入力された制御モデルから、所定の処理と、特定条件が成立するか否かを判定する判定処理とを行い、判定処理により特定条件が成立するか或いは特定条件が成立しないと判定されると、所定の処理の演算結果を訂正するための追加処理を行うように設定された演算モデル部を検出すると（Ｓ１１０，Ｓ１２０：ＹＥＳ）、判定処理で判定される特定条件の成立頻度を取得し（Ｓ１４０）、その取得した成立頻度に基づき（Ｓ１５０）、追加処理の実行頻度が低くなるような前記演算モデル部に対応するプログラムコードを生成する（Ｓ１６０或いはＳ１７０）。
【選択図】図４



【特許請求の範囲】
【請求項１】
所定の制御対象の制御仕様を表す制御モデルであって、制御のための演算処理を表す演算モデル部を少なくとも有する制御モデルを取得するための制御モデル取得手段と、
前記制御モデル取得手段により取得された制御モデル（以下、取得制御モデルと言う）に対応する制御プログラムのプログラムコードを、予め定められたルールに基づいて生成する生成手段と、
を備えたプログラムコード生成装置において、
前記生成手段は、
前記演算モデル部のうち、その処理内容が、コンピュータに、第１の処理と、特定の条件が成立するか否かを判定する判定処理とを実行させると共に、その判定処理により前記特定条件が成立したと判定されたなら、前記第１の処理を無効化して第２の処理を実行させる第１プログラムと、コンピュータに、前記第２の処理と、前記判定処理とを実行させると共に、その判定処理により前記特定条件が成立しないと判定されたなら、その第２の処理を無効化して前記第１の処理を実行させる第２プログラムとの何れでも表すことのできる演算モデル部（以下、条件判定演算モデル部と言う）を、前記取得制御モデルから検索するモデル解析手段と、
前記モデル解析手段により検索された条件判定演算モデル部について前記特定条件の成立頻度を取得するための成立頻度取得手段と、
前記成立頻度取得手段により取得される前記特定条件の成立頻度が既定値より高いか否かを判定する成立頻度判定手段とを備え、さらに、
前記成立頻度判定手段の判定結果に基づき、前記第１プログラムのプログラムコード（以下、第１プログラムコードと言う）と前記第２プログラムのプログラムコード（以下、第２プログラムコードと言う）との何れかを、前記条件判定演算モデル部のプログラムコードとして生成するようになっていることを特徴とするプログラムコード生成装置。
【請求項２】
請求項１に記載のプログラムコード生成装置において、
前記生成手段は、
前記成立頻度判定手段により前記特定条件の成立頻度が規定値以下と判定されたならば、前記条件判定演算モデル部のプログラムコードとして、前記第１プログラムコードを生成し、前記成立頻度判定手段により前記特定条件の成立頻度が規定値より高いと判定されたならば、前記条件判定演算モデル部のプログラムコードとして、前記第２プログラムコードを生成するようになっていることを特徴とするプログラムコード生成装置。
【請求項３】
請求項２に記載のプログラムコード生成装置において、
前記制御モデルに含まれる可能性のある前記条件判定演算モデル部のそれぞれについて、前記第１プログラムコードと前記第２プログラムコードとを記憶するコード記憶手段を備え、
前記生成手段は、前記成立頻度判定手段により前記特定条件の成立頻度が既定値以下と判定されたならば、前記コード記憶手段から、前記モデル解析手段により検索された前記条件判定演算モデル部に対応する前記第１プログラムコードを読み出し、前記成立頻度判定手段により成立頻度が既定値よりも高いと判定されたならば、前記コード記憶手段から、前記モデル解析手段により検索された前記条件判定演算モデル部に対応する前記第２プログラムコードを読み出すようになっていることを特徴とするプログラムコード生成装置。
【請求項４】
請求項２又は請求項３に記載のプログラムコード生成装置において、
前記制御モデルに含まれる可能性のある前記条件判定演算モデル部のそれぞれについて、その条件判定演算モデル部を表すテンプレートとして、前記第１プログラムコードに対応する第１モデルと前記第２プログラムコードに対応する第２モデルとを記憶するモデル

記憶手段を備え、
前記生成手段は、前記成立頻度判定手段により前記特定条件の成立頻度が既定値以下と判定されたならば、前記モデル記憶手段から、前記モデル解析手段により検索された前記条件判定演算モデル部に対応する前記第１モデルを読み出して、該読み出した第１モデルに対応するプログラムコードを生成し、前記成立頻度判定手段により前記特定条件の成立頻度が既定値よりも高いと判定されたならば、前記モデル記憶手段から、前記モデル解析手段により検索された前記条件判定演算モデル部に対応する前記第２モデルを読み出して、該読み出した第２モデルに対応するプログラムコードを生成するようになっていることを特徴とするプログラムコード生成装置。
【請求項５】
請求項１ないし請求項４の何れか１項に記載のプログラムコード生成装置において、
情報を表示する表示手段と、
前記成立頻度判定手段により前記特定条件の成立頻度が既定値以下と判定されたならば、前記第１プログラムコードに対応する条件判定演算モデル部の画像を前記表示手段に表示させ、前記成立頻度判定手段により前記特定条件の成立頻度が既定値よりも高いと判定されたならば、前記第２プログラムコードに対応する条件判定演算モデル部の画像を前記表示手段に表示させる画像表示制御手段と、
を備えていることを特徴とするプログラムコード生成装置。
【請求項６】
請求項１ないし請求項５の何れか１項に記載のプログラムコード生成装置において、
情報を入力するための入力手段を備え、
前記成立頻度取得手段は、前記入力手段を介して入力される使用者からの情報に基づき、前記条件判定演算モデル部における前記判定処理で判定される前記特定条件の成立頻度を取得することを特徴とするプログラムコード生成装置。
【請求項７】
請求項１ないし請求項６の何れか１項に記載のプログラムコード生成装置において、
前記取得制御モデルに対応する制御プログラムを実行して、前記モデル解析手段により検索された前記条件判定演算モデル部における前記判定処理で判定される前記特定条件が成立するか否かを検出する検出手段を備え、
前記成立頻度取得手段は、前記モデル解析手段により検索された前記条件判定演算モデル部における前記判定処理で判定される前記特定条件の成立頻度を、前記検出手段の検出結果に基づき取得することを特徴とするプログラムコード生成装置。
【請求項８】
請求項１ないし請求項７の何れか１項に記載のプログラムコード生成装置において、
前記条件判定演算モデル部は少なくとも１つ以上あり、
前記条件判定演算モデル部における前記判定処理で判定される前記特定条件の成立頻度を、前記モデル解析手段により検索される条件判定演算モデル部と関連づけて記憶する成立頻度記憶手段を備え、
前記成立頻度取得手段は、前記成立頻度記憶手段に記憶された成立頻度から、前記モデル解析手段により検索された前記条件判定演算モデル部における前記判定処理で判定される前記特定条件の成立頻度を取得するようになっていることを特徴とするプログラムコード生成装置。
【請求項９】
所定の制御対象の制御仕様を表す制御モデルであって、制御のための演算処理を表す演算モデル部を少なくとも有する制御モデルを取得するための制御モデル取得手段と、
前記制御モデル取得手段により取得された制御モデル（以下、取得制御モデルと言う）に対応する制御プログラムのプログラムコードを、予め定められたルールに基づいて生成する生成手段と、
を備えたプログラムコード生成装置において、
前記生成手段は、

前記演算モデル部のうち、その処理内容が、互いに排他的な複数通りの条件のうちの何れが成立するかによって実行所要時間がそれぞれ異なる複数種類のプログラムの何れでも表すことのできる演算モデル部（以下、特定演算モデル部と言う）を前記取得制御モデルから検索するモデル解析手段と、

前記複数通りの条件のうち、最も成立頻度の高い条件を取得する条件取得手段とを備え、

前記複数種類のプログラムのうち、前記条件取得手段が取得した条件が成立した場合に実行所要時間が最も短くなるプログラムのプログラムコードを、前記特定演算モデル部のプログラムコードとして生成するようになっていることを特徴とするプログラムコード生成装置。

【請求項１０】

請求項９に記載のプログラムコード生成装置において、

前記制御モデルに含まれる可能性のある前記特定演算モデル部のそれぞれについて、前記複数種類のプログラムのそれぞれのプログラムコードを記憶するコード記憶手段を備え、

前記生成手段は、前記コード記憶手段から、前記モデル解析手段により検索された前記特定演算モデル部に対応すると共に、前記条件取得手段が取得した条件が成立した場合に実行所要時間が最も短くなるプログラムのプログラムコードを読み出すようになっていることを特徴とするプログラムコード生成装置。

【請求項１１】

請求項９又は請求項１０に記載プログラムコード生成装置において、

前記制御モデルに含まれる可能性のある前記特定演算モデル部のそれぞれについて、前記複数種類のプログラムのそれぞれを表すテンプレートを記憶するテンプレート記憶手段を備え、

前記生成手段は、前記テンプレート記憶手段から、前記モデル解析手段により検索された前記特定演算モデル部に対応すると共に、前記条件取得手段が取得した条件が成立した場合に実行所要時間が最も短くなるプログラムのテンプレートを読み出し、該読み出したテンプレートが表すプログラムのプログラムコードを、前記モデル解析手段により検索された前記特定演算モデル部のプログラムコードとして生成するようになっていることを特徴とするプログラムコード生成装置。

【発明の詳細な説明】

【技術分野】

【０００１】

本発明は、入力された制御モデルに対応する制御プログラムのプログラムコードを生成するプログラムコード生成装置に関する。

【背景技術】

【０００２】

従来より、使用者が、演算を行う演算ブロックを含む制御モデル等を入力すると共に、プログラムコードの生成指令を入力すると、予め設定された生成ルールに基づいて、入力された制御モデルに対応したプログラムコードを出力するプログラムコード生成装置が知られている（例えば、特許文献１参照）。

【０００３】

このようなプログラムコード生成装置では、例えば図１７（ａ）に示すような制御モデルが使用者により入力される。

図１７（ａ）の制御モデルは、車両に搭載される電子制御装置が車両の各部を制御する制御モデルの一例であり、後述するように、数値を表すブロックや、各種演算処理を行うブロック等から構成される。

【０００４】

まず、符号Ａ，Ｂ，Ｃが付されたブロック（以下、それぞれ、ブロックＡ、ブロックＢ、ブロックＣと記載する）は、変数を表すブロックである。また、変数は、本例の制御に

おける制御量を表すものである。尚、ブロックＡ，Ｂは入力側（入力１及び入力２）であり、ブロックＣは出力側（出力１）である。

【０００５】

そして、ブロックの領域内に符号Ｄ，Ｅが記載されたＣｏｎｓｔａｎｔブロック（以下、それぞれ、ブロックＤ、ブロックＥと記載する）は、定数を表すブロックである。

尚、以下、ブロックＡが表す変数（制御量）を制御量Ａと記載する。ブロックＢ，Ｃについても同様とする。また、ブロックＤが表す定数を定数Ｄと記載する。ブロックＥについても同様とする。

【０００６】

次に、ブロックの領域内に大小を表す符号（「＞」）が記載されたＢＬＫ１ブロックは、比較演算処理を行うブロックである。具体的に、ＢＬＫ１ブロックには、数値が入力される入力ポートして、ＢＬＫ第１ポートとＢＬＫ第２ポートとがあり、このＢＬＫ１ブロックでは、ＢＬＫ第１ポートとＢＬＫ第２ポートとに入力される数値について、大小が比較される。そして、ＢＬＫ第１ポートの数値がＢＬＫ第２ポートの数値よりも大きいという条件、つまり、「ＢＬＫ第１ポートの数値＞ＢＬＫ第２ポートの数値」という条件が成立する場合には、出力ポートの出力が「１」になり、一方、その条件が成立しない場合には、出力ポートの出力が「０」になるようになっている。

【０００７】

ここで、図１７（ａ）の例では、ＢＬＫ１ブロックに到達する矢印が２つ記述されており、一方は、ブロックＡを始点としてＢＬＫ第１ポートに到達するように、また、他方は、ブロックＢを始点としてＢＬＫ第２ポートに到達するように記述されている。これは、制御量ＡがＢＬＫ第１ポートに入力され、制御量ＢがＢＬＫ第２ポートに入力されるということを表している。

【０００８】

つまり、このＢＬＫ１ブロックでは、制御量Ａと制御量Ｂとの大小が比較され、「制御量Ａ＞制御量Ｂ」という条件が成立する場合には、出力が「１」になり、「制御量Ａ＞制御量Ｂ」という条件が成立しない場合、つまり「制御量Ａ≦制御量Ｂ」となる場合には、出力が「０」となるような比較演算処理が行われる。

【０００９】

次に、ブロックＤ，Ｅ及びＢＬＫ１ブロックから延びる矢印の先端にあるＳｗｉｔｃｈブロックは、出力を切り換える切換処理を行うブロックである。具体的に、Ｓｗｉｔｃｈブロックには、数値が入力される入力ポートとして、第１～第３ポートがあり、このＳｗｉｔｃｈブロックでは、第２ポートの入力が「０」の場合は、第３ポートに入力される数値が出力ポートから出力され、第２ポートの入力が「１」の場合は、第１ポートに入力される数値が出力ポートから出力されるように、出力が切り換えられる。

【００１０】

そして、図１７（ａ）の例では、ブロックＤを始点とする矢印が第１ポートに到達するように、ブロックＥを始点とする矢印が第３ポートに到達するように、さらに、ＢＬＫ１ブロックの出力ポートを始点とする矢印が第２ポートに到達するように記述されている。これは、第１ポートに定数Ｄが入力され、第３ポートに定数Ｅが入力され、第２ポートにＢＬＫ１ブロックからの出力値（「０」或いは「１」）が入力されることを表している。

【００１１】

つまり、Ｓｗｉｔｃｈブロックでは、ＢＬＫ１ブロックの出力（第２ポートの入力）が「０」の場合は、出力ポートから定数Ｅが出力され、ＢＬＫ１ブロックの出力（第２ポートの入力）が「１」の場合は、出力ポートから定数Ｄが出力される、という切換処理が行われる。

【００１２】

そして、図１７（ａ）では、Ｓｗｉｔｃｈブロックの出力ポートを始点とし、ブロックＣに到達するように、矢印が記述されている。これは、制御量Ｃの数値は、Ｓｗｉｔｃｈブロックの出力ポートから出力される数値、つまり、定数Ｄ或いは定数Ｅの何れかである

ことを表すものである。
【００１３】
以上のように構成された本制御モデルでは、次のような制御が実現されることとなる。つまり、「制御量Ｃに定数Ｅを代入する。ただし、制御量Ａが制御量Ｂよりも大きい（制御量Ａ＞制御量Ｂ）という条件が成立する場合は、制御量Ｃに定数Ｄを代入する」という制御（以下、制御１と言う）である。
【００１４】
そして、従来のプログラムコード生成装置において、このような制御モデルに基づいてプログラムコードを生成する場合、図１７（ｂ）に示すようなプログラムコードが生成される。つまり、制御量Ｃに定数Ｅを代入するという意味の「Ｃ＝Ｅ」、制御量Ａ＞制御量Ｂの条件が成立する場合は制御量Ｃに定数Ｄを代入するという意味の「ｉｆ（Ａ＞Ｂ）｛Ｃ＝Ｄ｝」、という記述がされたプログラムコードが生成される。
【特許文献１】特願２００３－１７３２５６号公報
【発明の開示】
【発明が解決しようとする課題】
【００１５】
ところで、上述の図１７（ｂ）のようなプログラムコードが車両の電子制御装置に実装される場合において、制御量Ａ＞制御量Ｂという条件（以下、条件Ｓと言う）が成立しない、つまり、制御量Ａ≦制御量Ｂとなる場合には、「制御量Ｃに定数Ｄを代入する」という部分に該当する処理（以下、処理Ｙと言う）はしなくてよい。
【００１６】
一方、条件Ｓが成立する場合には、処理Ｙを実行して、制御量Ｃに定数Ｄを代入し直す必要がある。
そうすると、条件Ｓが成立する場合には、処理Ｙを実行して制御量Ｃに定数Ｄを代入し直す分、電子制御装置が実行すべき処理ステップ数が増加することとなり、プログラムの実行時間や、処理負荷が増大する。
【００１７】
このため、条件Ｓの成立する頻度（確率）が高い場合（具体的に、５０％を超える場合）には、以下の制御を実現するプログラムコードのほうが有利である。つまり、「制御量Ｃに定数Ｄを代入する。ただし、制御量Ａ≦制御量Ｂという条件が成立する場合は、制御量Ｃに定数Ｅを代入する。」という制御（以下、制御２と言う）である。このように変更すれば、上記の制御１と同じ制御結果が得られつつ、条件（ここでは、制御量Ａ≦制御量Ｂという条件）が成立することにより実行される処理（制御量Ｃに定数Ｅを代入し直す処理）の実行頻度を抑えることができるため、プログラムの実行時間や処理負荷を低減できる。
【００１８】
そして、従来は、上述のような制御モデルにおいて、制御１及び制御２の何れを採用するかは、制御モデルを作成する設計者により決定されていた。また、設計者等は、実験やシミュレーションにより、対象とする条件（例えば条件Ｓ）の成立頻度を検証し、その成立頻度に応じて、制御モデルやプログラムコードを修正していた。一方、上述のＢＬＫ１ブロックのような比較演算処理を行うブロックは、車両の各部の制御を実現する制御モデルの中に多数存在する。
【００１９】
このため、電子制御装置でのプログラムの実行時間や処理負荷を低減するために最適な制御プログラムを開発する場合、制御モデルやプログラムコードの作成、修正に手間がかかり、開発工数が増大してしまう、ということが問題となっていた。
【００２０】
本発明は、こうした問題に鑑みなされたもので、電子制御装置に実装するための制御プログラムのプログラムコードを自動生成するプログラムコード生成装置において、電子制御装置での制御プログラムの実行時間や処理負荷が低減できる制御プログラムを、容易に

生成できるようにすることを目的とする。
【課題を解決するための手段】
【００２１】
上記目的を達成するためになされた請求項１に記載のプログラムコード生成装置においては、制御モデル取得手段が、所定の制御対象の制御仕様を表す制御モデルを取得し、生成手段が、制御モデル取得手段により取得された制御モデル（以下、取得制御モデルと言う）に対応する制御プログラムのプログラムコードを、予め定められたルールに基づいて生成する。そして、制御モデル取得手段が取得する制御モデルは、制御のための演算処理を表す演算モデル部を少なくとも有している。
【００２２】
ここで特に、請求項１のプログラムコード生成装置において、生成手段は、モデル解析手段と、成立頻度取得手段と、成立頻度判定手段とを備えている。
モデル解析手段は、演算モデル部のうち、その処理内容が、コンピュータに、第１の処理と、特定の条件が成立するか否かを判定する判定処理とを実行させると共に、その判定処理により特定条件が成立したと判定されたなら、第１の処理を無効化して第２の処理を実行させる第１プログラムと、コンピュータに、第２の処理と、判定処理とを実行させると共に、その判定処理により特定条件が成立しないと判定されたなら、その第２の処理を無効化して第１の処理を実行させる第２プログラムとの何れでも表すことのできる演算モデル部（以下、条件判定演算モデル部と言う）を、取得制御モデルから検索する。
【００２３】
また、成立頻度取得手段は、モデル解析手段により検索された条件判定演算モデル部について、判定処理で判定される特定条件の成立頻度を取得し、成立頻度判定手段は、成立頻度取得手段により取得される特定条件の成立頻度が既定値より高いか否かを判定する。
【００２４】
ところで、条件判定演算モデル部において、特定条件が成立すると、第１の処理を無効化して第２の処理を実行させるようになっている場合、その特定条件の成立頻度が高いのであれば、第１の処理を無効化して第２の処理を実行させる頻度も高い。逆に、特定条件の成立頻度が低ければ、第１の処理を無効化して第２の処理を実行させる頻度も低い。
【００２５】
また、条件判定演算モデル部において、特定条件が成立しないと、第２の処理を無効化して第１の処理を実行させるようになっている場合、その特定条件の成立頻度が低ければ、第２の処理を無効化して第１の処理を実行させる頻度は高い。逆に、特定条件の成立頻度が高ければ、第２の処理を無効化して第１の処理を実行させる頻度は低い。
【００２６】
そして、最初の処理を無効化して別の処理を実行させることとなると、その分、演算処理の処理ステップ数が増加することとなるため、最初の処理を無効化して別の処理を実行させる頻度が高くなることは、実行時間や処理負荷の点で不利である。
【００２７】
そこで、本装置では、生成手段は、成立頻度判定手段による判定結果、つまり、特定条件の成立頻度が規定値より高いか否かの判定結果に基づき、前記第１プログラムのプログラムコード（以下、第１プログラムコードと言う）と前記第２プログラムのプログラムコード（以下、第２プログラムコードと言う）との何れかを、条件判定演算モデル部のプログラムコードとして生成する。
【００２８】
これによれば、特定条件の成立頻度が高い場合と低い場合とのそれぞれに応じて、最初の処理を無効化して別の処理を実行する、ということがなるべく生じないようなプログラムコードを生成するように構成することができる。そして、このため、プログラムコードが所定の装置に実装された際のその装置におけるプログラムの実行時間や処理負荷を抑えることができるプログラムコードを、容易に生成できるようになる。
【００２９】

そして、より詳しくは、請求項２のように構成すればよい。

請求項２のプログラムコード生成装置は、請求項１のプログラムコード生成装置において、生成手段は、成立頻度判定手段により特定条件の成立頻度が規定値以下と判定されたならば、条件判定演算モデル部のプログラムコードとして、第１プログラムコードを生成する。一方、生成手段は、成立頻度判定手段により特定条件の成立頻度が規定値より高いと判定されたならば、条件判定演算モデル部のプログラムコードとして、第２プログラムコードを生成する。

【００３０】

その第１プログラムコードによれば、特定条件の成立頻度が規定値以下である場合、第１の処理を無効化して第２の処理を実行させる頻度が低くなり、また、第２プログラムコードによれば、特定条件の成立頻度が規定値より高い場合、第２の処理を無効化して第１の処理を実行させる頻度が低くなることとなる。

【００３１】

また、このような第１，第２プログラムコードを生成するようにする場合、請求項３のように構成すればよい。

請求項３のプログラムコード生成装置は、請求項２のプログラムコード生成装置において、制御モデルに含まれる可能性のある条件判定演算モデル部のそれぞれについて、第１プログラムコードと第２プログラムコードとを記憶するコード記憶手段を備えている。尚、第１，第２プログラムコードは、予め生成しておき、コード記憶手段に記憶させておけばよい。

【００３２】

そして、生成手段は、成立頻度判定手段により特定条件の成立頻度が既定値以下と判定されたならば、コード記憶手段から、モデル解析手段により検索された条件判定演算モデル部に対応する第１プログラムコードを読み出す。また、成立頻度判定手段により成立頻度が既定値よりも高いと判定されたならば、コード記憶手段から、モデル解析手段により検索された条件判定演算モデル部に対応する第２プログラムコードを読み出すようになっている。

【００３３】

これによれば、コード記憶手段から、モデル解析手段により検索された条件判定演算モデル部に対応する第１プログラムコード或いは第２プログラムコードを読み出すだけでよいため、その第１プログラムコード或いは第２プログラムコードを生成するための処理を省略することができる点で有利である。

【００３４】

また一方、請求項４のように構成しても良い。

請求項４のプログラムコード生成装置は、請求項２又は請求項３のプログラムコード生成装置において、制御モデルに含まれる可能性のある条件判定演算モデル部のそれぞれについて、その条件判定演算モデル部を表すテンプレートとして、第１プログラムコードに対応する第１モデルと第２プログラムコードに対応する第２モデルとを記憶するモデル記憶手段を備えている。尚、第１，第２モデルは、予め生成しておき、モデル記憶手段に記憶させておけばよい。

【００３５】

そして、生成手段は、成立頻度判定手段により特定条件の成立頻度が既定値以下と判定されたならば、モデル記憶手段から、モデル解析手段により検索された条件判定演算モデル部に対応する第１モデルを読み出して、該読み出した第１モデルに対応するプログラムコードを生成する。また、成立頻度判定手段により特定条件の成立頻度が既定値よりも高いと判定されたならば、モデル記憶手段から、モデル解析手段により検索された条件判定演算モデル部に対応する第２モデルを読み出して、該読み出した第２モデルに対応するプログラムコードを生成するようになっている。

【００３６】

このような請求項４のプログラムコード生成装置によれば、例えば所望のプログラムコ

ードが生成されるように修正を加える場合、テンプレートとしての第１，第２モデルを修正すれば済む。

【００３７】

また、このような場合、特に、第１モデル或いは第２モデルが使用者に画像で認識できるようにすれば、使用者にとって使い勝手がよい。

つまり、プログラムコード生成装置においては、使用者等が、生成手段により生成されるプログラムコードに対応する制御モデルを視覚で認識できることが好ましい。

【００３８】

そこで、請求項５のプログラムコード生成装置は、請求項１～４のプログラムコード生成装置において、情報を表示する表示手段と、画像表示制御手段とを備えている。

画像表示制御手段は、成立頻度判定手段により特定条件の成立頻度が既定値以下と判定されたならば、第１プログラムコードに対応する条件判定演算モデル部の画像を表示手段に表示させ、また、成立頻度判定手段により特定条件の成立頻度が既定値よりも高いと判定されたならば、第２プログラムコードに対応する条件判定演算モデル部の画像を表示手段に表示させる。

【００３９】

これによれば、使用者は、第１プログラムコード或いは第２プログラムコードのそれぞれについて、対応する条件判定演算モデル部の画像を、表示手段を介して確認することができるため、使い勝手がよくなる。

【００４０】

次に、請求項６のプログラムコード生成装置は、請求項１～５のプログラムコード生成装置において、情報を入力するための入力手段を備えている。そして、成立頻度取得手段は、入力手段を介して入力される使用者からの情報に基づき、条件判定演算モデル部における判定処理で判定される特定条件の成立頻度を取得する。

【００４１】

つまり、本請求項６の装置によれば、使用者が、成立頻度を入力手段を介して入力することができる。またこの場合、条件判定演算モデル部が複数あれば、その複数の条件判定演算モデル部に対して個々に成立頻度を入力できる構成としても良いし、特定の入力値が、その複数の条件判定演算モデル部の一部（少なくとも２つ以上）或いは全部に同時に適用される構成としてもよい。これによれば、使用者にとって使い勝手が良くなる。

【００４２】

また、成立頻度は、請求項７のように、自動で検出されるようにすれば、より使い勝手がよくなる。つまり、請求項７のプログラムコード生成装置は、請求項１～６のプログラムコード生成装置において、取得制御モデルに対応する制御プログラムを実行して、モデル解析手段により検索された条件判定演算モデル部における判定処理で判定される特定条件が成立するか否かを検出する検出手段を備えている。

【００４３】

ここで、制御プログラムを実行する構成としては、コンピュータ装置に制御対象を接続し、コンピュータ装置と制御対象との間で制御のための各種信号が入出力されるようにした実際の状態で、コンピュータ装置が制御プログラムを実行する構成が考えられる。一方、コンピュータ装置に制御対象を接続せず、そのコンピュータ装置上のみで、制御プログラムを模擬的に実行する（所謂シミュレーションである）構成が考えられる。

【００４４】

そして、請求項７では、シミュレーションが実行される後者のように構成することが好ましい。これによれば、容易に、特定条件が成立するか否かを表すデータ（以下、条件成立データと言う）が得られる。また、シミュレーションを繰り返し実行するようにすれば、条件成立データを多数取得できることとなる。このため、その条件成立データの信頼性を向上させることができる。

【００４５】

そして、成立頻度取得手段は、モデル解析手段により検索された条件判定演算モデル部

における判定処理で判定される特定条件の成立頻度を、検出手段の検出結果に基づき取得する。

【００４６】

尚、条件成立データは、特定条件が成立する回数及び成立しない回数を表すデータとすることが考えられる。この場合、成立頻度取得手段は、その回数を表すデータに基づき、特定条件の成立頻度を取得するようにすればよい。具体的に、特定条件が成立する回数と成立しない回数とから、成立する割合、即ち成立頻度を算出するようにできる。また、特定条件が成立する回数が成立しない回数よりも多ければ、頻度が高いという旨の成立頻度を取得し、逆に、特定条件が成立する回数が成立しない回数よりも少なければ、頻度が低いという旨の成立頻度を取得するようにできる。

【００４７】

次に、請求項８のプログラムコード生成装置は、請求項１～７のプログラムコード生成装置において、条件判定演算モデル部における判定処理で判定される特定条件の成立頻度を、モデル解析手段により検索される条件判定演算モデル部と関連づけて記憶する成立頻度記憶手段を備えている。

【００４８】

これによれば、条件判定演算モデル部のそれぞれについての特定条件の成立頻度を、成立頻度記憶手段に記憶させるようにすることができる。

そして、成立頻度取得手段は、成立頻度記憶手段に記憶された成立頻度から、モデル解析手段により検索された条件判定演算モデル部における判定処理で判定される特定条件の成立頻度を取得するようになっている。

【００４９】

このような本請求項８の装置によれば、条件判定演算モデル部が複数ある場合でも、その条件判定演算モデル部のそれぞれについて、前述したように、成立頻度に応じたプログラムコードを生成するようにできる。つまり、条件判定演算モデル部のそれぞれについて、追加処理の実行頻度が低くなるようなプログラムコードを生成することができる。

【００５０】

次に、請求項９のプログラムコード生成装置は、所定の制御対象の制御仕様を表す制御モデルであって、制御のための演算処理を表す演算モデル部を少なくとも有する制御モデルを取得するための制御モデル取得手段を備えている。また、制御モデル取得手段により取得された制御モデル（以下、取得制御モデルと言う）に対応する制御プログラムのプログラムコードを、予め定められたルールに基づいて生成する生成手段を備えている。

【００５１】

この生成手段は、モデル解析手段と、条件取得手段とを備えている。

モデル解析手段は、演算モデル部のうち、その処理内容が、互いに排他的な複数通りの条件のうちの何れが成立するかによって実行所要時間がそれぞれ異なる複数種類のプログラムの何れでも表すことのできる演算モデル部（以下、特定演算モデル部と言う）を前記取得制御モデルから検索する。

【００５２】

一方、条件取得手段は、複数通りの条件のうち、最も成立頻度の高い条件を取得する。

そして、生成手段は、複数種類のプログラムのうち、条件取得手段が取得した条件が成立した場合に実行所要時間が最も短くなるプログラムのプログラムコードを、特定演算モデル部のプログラムコードとして生成するようになっている。

【００５３】

つまり、本請求項９のプログラムコード生成装置では、複数種類のプログラムの何れでも表すことのできる特定演算モデル部について、条件の成立頻度に応じて、実行所要時間が最も短くなるなるようなプログラムコードを生成する。

【００５４】

このため、本請求項９の装置によれば、プログラムコードが所定の装置に実装された際のその装置におけるプログラムの実行時間や処理負荷を抑えることができるプログラムコ

ードを、容易に生成できるようになる。
【００５５】
　また、この場合、請求項１０のように構成することが好ましい。
　請求項１０のプログラムコード生成装置は、請求項９の装置において、制御モデルに含まれる可能性のある特定演算モデル部のそれぞれについて、複数種類のプログラムのそれぞれのプログラムコードを記憶するコード記憶手段を備え、生成手段は、コード記憶手段から、モデル解析手段により検索された特定演算モデル部に対応すると共に、条件取得手段が取得した条件が成立した場合に実行所要時間が最も短くなるプログラムのプログラムコードを読み出すようになっている。
【００５６】
　これによれば、モデル解析手段により検索された特定演算モデル部に対応し、かつ条件取得手段が取得した条件が成立した場合に実行所要時間が最も短くなるプログラムのプログラムコードを、コード記憶手段から読み出すだけでよいため、そのプログラムコードを生成するための処理を省略することができる点で有利である。
【００５７】
　また一方、請求項１１のように構成しても良い。
　請求項１１のプログラムコード生成装置は、請求項９又は請求項１０の装置において、制御モデルに含まれる可能性のある特定演算モデル部のそれぞれについて、複数種類のプログラムのそれぞれを表すテンプレートを記憶するテンプレート記憶手段を備えている。
【００５８】
　そして、生成手段は、テンプレート記憶手段から、モデル解析手段により検索された特定演算モデル部に対応すると共に、条件取得手段が取得した条件が成立した場合に実行所要時間が最も短くなるプログラムのテンプレートを読み出し、該読み出したテンプレートが表すプログラムのプログラムコードを、モデル解析手段により検索された特定演算モデル部のプログラムコードとして生成するようになっている。
【００５９】
　このような請求項１１のプログラムコード生成装置によれば、例えば所望のプログラムコードが生成されるように修正を加える場合、テンプレートを修正すれば済む。
　ところで、テンプレートは、どのような形式のテンプレートでもよく、例えば、プログラムコードの形式で記述されたテンプレートでも良いし、モデルの形式で記述されたテンプレートでも良い。
【００６０】
　尚、請求項５～７はそれぞれ、請求項９～１１に順次従属させることができる。
【発明を実施するための最良の形態】
【００６１】
　以下に、本発明の実施形態を図面に基づき説明する。
［実施形態１］
　図１は、本発明が適用されたプログラムコード生成装置３の構成図である。
【００６２】
　プログラムコード生成装置３は、例えば車両制御用の制御プログラムのプログラムコードを生成するものであり、周知のコンピュータシステムとして構成されている。そして、そのハードウェア構成は従来より当業者によく知られたものである。このため、一部図示を省略するが、プログラムコード生成装置３は、ＣＰＵ、ＲＯＭ、ＲＡＭ、Ｉ／Ｏおよびこれらを接続するバスラインをはじめ、周辺機器としてのハードディスク装置、キーボード・ポインティングデバイス等の各種入力部（後述するモデル入力部５１、設定入力部５２、開始指令入力部５３等に相当）、ディスプレイ（後述する表示部４０に相当）、及び外部との間で通信を行うための通信装置等を備えている。
【００６３】
　そして、図１では、プログラムコード生成装置３の機能をブロックにて表すようにしている。以下、具体的に説明する。

本実施形態のプログラムコード生成装置３は、制御仕様を表す後述の制御モデル２０から、その制御モデル２０に対応する制御プログラムのプログラムコード（ソースコード）を生成するコード生成ツール１０と、制御モデル２０を入力するためのモデル入力部５１と、制御モデル２０やコード生成ツール１０に対して各種設定を行うための設定入力部５２と、コード生成ツール１０にプログラムコードの生成やシミュレーションの実行のための各種指令を入力する開始指令入力部５３と、コード生成ツール１０により生成されたプログラムコードをオブジェクトコードに変換するコンパイラ・リンカ３１と、表示部４０とを備えている。そして、コンパイラ・リンカ３１により作成されたオブジェクトコードは、電子制御装置（以下「ＥＣＵ」という）のフラッシュＲＯＭ３２に書き込まれる。また、表示部４０には、モデル入力部５１を介して入力された制御モデル２０が表示されるようになっている。

【００６４】

コード生成ツール１０は、抽出エンジン１１と、コード生成処理部１２と、シミュレーション処理部１３と、生成ルール記憶部１４とを備えている。尚、プログラムコード生成装置３において演算が実行される際には、より高精度な演算を行うために浮動小数点演算により実行される。

【００６５】

抽出エンジン１１は、制御モデル２０を読み込んで中間ファイルを出力する。具体的に、抽出エンジン１１は、制御モデル２０から、後述する演算ブロックの情報や、これらの演算ブロックに入力されるパラメータ及びそのパラメータ名の情報等、プログラムコードの生成のために必要な各種情報を抽出し、その抽出した各種情報を含む中間ファイルを出力する。

【００６６】

コード生成処理部１２は、抽出エンジン１１から出力される中間ファイルに対応するプログラムコードを生成する。尚、詳細は後述する。

そして、生成ルール記憶部１４は、コード生成処理部１２が中間ファイルに対応するプログラムコードを生成するための生成ルールを記憶する。また、生成ルール記憶部１４は、後述する比較演算ブロック２３における判定処理で判定される特定条件の成立頻度を記憶する成立頻度記憶部１５と、プログラムコードのテンプレートを記憶するコード記憶部１６とを有している。

【００６７】

また、シミュレーション処理部１３は、抽出エンジン１１から出力される前述の中間ファイルに基づきシミュレーションを行う。言い換えると、制御モデル２０に対応する制御プログラムをソフトウエア上で実行し、その実行結果を表す各種データをログデータとして検出する。そして、シミュレーションにより得られたそのログデータは、図示しないＲＡＭ、或いはハードディスク装置の所定の格納領域に記憶される。

【００６８】

次に、プログラムコード生成装置３において、プログラムコードの生成対象となる制御モデル２０について説明する。

制御モデル２０は、使用者等（例えば、設計者）により、前述のモデル入力部５１を介して入力されるものである。そして、図１に示すように、制御モデル２０は、加減算を行う加減算ブロック２１、乗除算を行う乗除算ブロック２２、比較演算を行う比較演算ブロック２３、入力値に応じて動作を換える切換ブロック２４、及び論理演算を行う論理演算ブロック２５等の演算ブロックや、変数ブロック２８及び定数ブロック２９等のパラメータを表すブロック（以下、単にパラメータブロックとも言う）から構成される。

【００６９】

ここで、図２は、制御モデル２０の具体例を表す図面である。

図２に示す制御モデル２０ａは、図１７（ａ）に示す制御モデルと同じである。尚、ＢＬＫ１ブロックは比較演算ブロック２３に相当し、Ｓｗｉｔｃｈブロックは切換ブロック２４に相当し、ブロックＡ，Ｂ，Ｃは変数ブロック２８に相当し、ブロックＤ，Ｅは定数

ブロック２９に相当している。

【００７０】

そして、この制御モデル２０ａが表す制御仕様も図１７（ａ）の制御モデルと同じである。つまり、制御モデル２０ａは、「制御量Ｃに定数Ｅを代入する。ただし、制御量Ａが制御量Ｂよりも大きい（制御量Ａ＞制御量Ｂ）という条件が成立する場合は、制御量Ｃに定数Ｄを代入する」という制御仕様（以下、第１の制御とも言う）を表している。

【００７１】

そして、図３（ａ）は、このような制御モデル２０ａに対応するプログラムコードである。つまり、図３（ａ）は、制御量Ｃに定数Ｅを代入するという意味の「Ｃ＝Ｅ」、制御量Ａ＞制御量Ｂの条件（以下、特定条件Ｓと言う）が成立するか否かを判定し、成立すると判定すると、制御量Ｃに定数Ｄを代入し直すという意味の「ｉｆ（Ａ＞Ｂ｛Ｃ＝Ｄ｝」、という記述がされたプログラムコードである。

【００７２】

一方、［発明が解決しようとする課題］にて説明したように、図２の制御モデル２０ａについて、特定条件Ｓの成立頻度が高い場合、以下のようにしたほうが有利である。つまり、「制御量Ｃに定数Ｄを代入する。ただし、制御量Ａが制御量Ｂよりも大きい（制御量Ａ＞制御量Ｂ）という条件が成立しない場合（制御量Ａ≦制御量Ｂが成立する場合）は、制御量Ｃに定数Ｅを代入し直す」という制御仕様（以下、第２の制御とも言う）にする。特定条件Ｓの成立頻度が高い場合、この第２の制御によれば、制御量Ｃに定数Ｅを代入し直すという処理の実行頻度が低くなるためである。尚、以下、制御量Ｃに定数Ｄを代入し直す処理及び制御量Ｃに定数Ｅを代入し直す処理のように、代入し直す処理のことを、追加処理とも記載する。

【００７３】

そして、この第２の制御に対応するプログラムコードが、図３（ｂ）のプログラムコードである。つまり、図３（ｂ）のプログラムコードは、制御量Ｃに定数Ｄを代入するという意味の「Ｃ＝Ｄ」、制御量Ａ≦制御量Ｂの条件が成立するか否かを判定し、その条件が成立すると判定すると（特定条件Ｓが成立しないと判定すると）、制御量Ｃに定数Ｅを代入し直すという意味の「ｉｆ（Ａ＜＝Ｂ）｛Ｃ＝Ｅ｝」、という記述がされたプログラムコードである。

【００７４】

そして、本実施形態のプログラムコード生成装置３によれば、制御モデル２０ａについて、特定条件Ｓの成立頻度に応じて、追加処理の実行頻度が低くなるように、図３（ａ）のプログラムコード、或いは図３（ｂ）のプログラムコードの何れかが生成される。以下、具体的に説明する。

【００７５】

まず、図４は、プログラムコード生成装置３のコード生成処理部１２が、制御モデル２０に対応する制御プログラムのプログラムコードを生成する処理の流れを表すフローチャートである。そして、図４の処理は、以下のタイミングで開始される。

【００７６】

まず、使用者により、制御モデル２０がモデル入力部５１を介して入力され、また、開始指令入力部５３を介してプログラムコードの生成を指令する生成指令が、コード生成ツール１０に入力されると、抽出エンジン１１が、制御モデル２０を読み込んで、図示しない処理にて前述の中間ファイルを出力する。そして、その中間ファイルは、コード生成処理部１２に入力されるようになっており、コード生成処理部１２は、その中間ファイルが入力されると、図４の処理を開始する。尚、使用者により制御モデル２０が入力されると、自動的に、生成指令がコード生成ツール１０に入力されるように構成してもよい。

【００７７】

そして、図４の処理において、コード生成処理部１２は、まず、Ｓ１１０で、プログラムコードを生成するか否か、言い換えると、プログラムコードの生成対象があるか否かを、中間ファイルの情報に基づき判定する。

【００７８】
ところで、制御モデル２０は、１つ以上の演算ブロックやパラメータブロックから構成され、所定の処理を実現するひとかたまりのモデル部を少なくとも有する。例えば、モデル部としては、加減算ブロック２１と、変数ブロック２８と、定数ブロック２９とから構成され、所定の変数及び定数に対して加減算を行うモデル部が考えられる。また、図２の制御モデル２０ａも、モデル部である。特に、制御モデル２０ａは、「Ｃ＝Ｅ」という処理と、特定条件Ｓが成立するか否かを判定する判定処理とを行うとともに、判定処理により特定条件Ｓが成立すると判定されると、「Ｃ＝Ｄ」という追加処理を行うように構成されたモデル部である。
【００７９】
尚、以下、所定の処理と、特定の条件が成立するか否かを判定する判定処理とを行うとともに、判定処理により特定条件が成立すると判定されるか、或いは判定処理により特定条件が成立しないと判定されるかの何れかの場合に、所定の処理の演算結果を訂正するための追加処理を行うように設定されたモデル部を、条件判定演算モデル部とも言う。
【００８０】
そして、Ｓ１１０では、プログラムコードの生成対象のモデル部があるか否かを判定する。
そして、生成対象のモデル部があると判断すると、プログラムコードを生成すると判定し（Ｓ１１０：ＹＥＳ）、Ｓ１２０へ移行する。
【００８１】
Ｓ１２０では、生成対象のモデル部が、前述の条件判定演算モデル部であるか否かを判定する。そして、条件判定演算モデル部でないと判定すると（Ｓ１２０：ＮＯ）、Ｓ１３０へ移行し、その条件判定演算モデル部でないと判定したモデル部に対応するプログラムコードを生成する。尚、Ｓ１３０の処理は一般的なものであり、ここでは詳しい説明を省略する。そして、その後、再びＳ１１０へ戻る。
【００８２】
一方、Ｓ１２０で、条件判定演算モデル部であると判定すると（Ｓ１２０：ＹＥＳ）、次にＳ１４０へ移行する。
Ｓ１４０では、条件判定演算モデル部における判定処理（特定条件が成立するか否かを判定する処理）で判定される特定条件の成立頻度を、前述の成立頻度記憶部１５から読み出す。また、この成立頻度は、後述するが、設定入力部５２を介して使用者が入力することができるように構成されている。つまり、成立頻度は、予め使用者に入力され、成立頻度記憶部１５に記憶される。また、シミュレーション処理部１３が、シミュレーションにより特定条件が成立するか否かを検出し、その検出結果を、成立頻度記憶部１５に記憶させておく構成とすることもできる（後述する第２実施形態）。この場合、その検出結果に基づき、成立頻度を取得するようにすればよい。
【００８３】
そして、Ｓ１４０で特定条件の成立頻度を読み出すと、Ｓ１５０へ進み、その読み出した成立頻度が規定値より高いか否かを判定する。尚、規定値は、予め使用者等により設定され、ＲＯＭやハードディスク装置等の所定の領域に記憶されるが、例えば５０％（５割）と設定しておくことが考えられる。
【００８４】
そして、成立頻度が規定値以下であると判定すると（Ｓ１５０：ＮＯ）、Ｓ１６０へ移行し、一方、成立頻度が規定値より高いと判定すると（Ｓ１５０：ＹＥＳ）、Ｓ１７０へ移行する。Ｓ１６０及びＳ１７０ではそれぞれ、条件判定演算モデル部のプログラムコードを生成するが、具体的に以下のようにする。
【００８５】
まず、本実施形態において、コード記憶部１６（図１参照）には、プログラムコードのテンプレートとして、図５に示すようなテンプレートが記憶されている。
図５（ａ）のテンプレート１では、ｏｕｔにｉｎ１を代入するという意味の「ｏｕｔ＝

ｉｎ１」、ｉｎ２＞ｉｎ３という特定条件が成立するか否かを判定し、特定条件が成立すると判定すると、ｏｕｔにｉｎ４を代入し直す処理を行うという意味の「ｉｆ（ｉｎ２＞ｉｎ３）｛ｏｕｔ＝ｉｎ４｝」、という記述がされている。

【００８６】

また、図５（ｂ）のテンプレート２は、ｏｕｔにｉｎ４を代入するという意味の「ｏｕｔ＝ｉｎ４」、ｉｎ２≦ｉｎ３という条件が成立するか否かを判定し、その条件が成立すると判定すると（ｉｎ２＞ｉｎ３という特定条件が成立しないと判定すると）、ｏｕｔにｉｎ１を代入し直す処理を行うという意味の「ｉｆ（ｉｎ２＜＝ｉｎ３）｛ｏｕｔ＝ｉｎ１｝」、という記述がされている。尚、ｏｕｔ，ｉｎ１，ｉｎ２，ｉｎ３，ｉｎ４にはそれぞれ、具体的なパラメータ名が記述される。

【００８７】

そして、Ｓ１６０では、コード記憶部１６からテンプレート１を読み出し、そのテンプレート１に基づき、プログラムコードを生成する。つまり、「ｉｎ２＞ｉｎ３」の成立頻度が規定値以下である場合には（Ｓ１５０：ＮＯ）、「ｉｎ２＞ｉｎ３」が成立する場合に追加処理（ｏｕｔ＝ｉｎ４）が行われるようになるテンプレート１を採用する。そして、Ｓ１６０の後、再びＳ１１０へ戻る。

【００８８】

一方、Ｓ１７０では、コード記憶部１６からテンプレート２を読み出し、そのテンプレート２に基づき、プログラムコードを生成する。つまり、「ｉｎ２＞ｉｎ３」の成立頻度が規定値より高い場合には（Ｓ１５０：ＮＯ）、「ｉｎ２＞ｉｎ３」が成立しない場合（ｉｎ２≦ｉｎ３が成立する場合）に追加処理（ｏｕｔ＝ｉｎ１）が行われるようになるテンプレート２を採用する。そして、Ｓ１７０の後、再びＳ１１０へ戻る。

【００８９】

そして、Ｓ１１０で、プログラムコードを生成すべき対象のモデル部がないと判定すると（Ｓ１１０：ＮＯ）、当該処理を終了する。

このような本実施形態のプログラムコード生成装置３において、前述の図２の制御モデル２０ａが入力されると、以下のようにして、その制御モデル２０ａに対応するプログラムコードが生成される。

【００９０】

制御モデル２０ａが入力されると共に生成指令が入力され、図４の処理が開始されると、まず、生成対象のモデル部があると判定し（Ｓ１１０：ＹＥＳ）、また、生成対象のモデル部が条件判定演算モデル部であると判定する（Ｓ１２０：ＹＥＳ）。そしてＳ１４０へ移行して、「制御量Ａ＞制御量Ｂ」という特定条件Ｓの成立頻度を成立頻度記憶部１５から読み出す。

【００９１】

ここで、図６は、成立頻度が入力できるように構成された制御モデル２０ａを表す図面であり、成立頻度は、予め、制御モデル２０ａを入力する際に入力することができる。この図６では、ＢＬＫ１ブロックで判定される特定条件Ｓの成立頻度ＦＲＥＱを設定するためのパラメータ設定領域２３ａが、ＢＬＫ１ブロックと接続線で接続された形態で設けられている。そして、このパラメータ設定領域２３ａは、設定入力部５２を介して、ＢＬＫ１ブロックに対して設けることができるようになっている。また、ＢＬＫ１ブロックが複数ある場合には、それぞれ設けることができるし、Ｓｗｉｔｃｈブロックに対しても設けることができる。さらに、パラメータ設定領域２３ａを介して設定されるパラメータの種類は、成立頻度ＦＲＥＱに限らず、使用者が任意に選択できる。

【００９２】

そして、特定条件Ｓの成立頻度ＦＲＥＱは、設定入力部５２を介して、パラメータ設定領域２３ａ内に記述することができ、本例では、％（パーセント）を表す数値で記述するようにされている。尚、成立頻度ＦＲＥＱの記述方法は、任意に設定することができる。例えば、「高い」或いは「低い」の何れかで記述する方法でもよいし、度数を表す数字（例えば、５であれば５割を表す）で記述する方法でもよいし、成立回数及び不成立回数を

記述するような方法でもよく、どのような形態を用いても良い。後述するパラメータ設定領域２３ｂについても同様である。

【００９３】

そして、パラメータ設定領域２３ａに記述された成立頻度ＦＲＥＱは、成立頻度記憶部１５に、その設定対象のＢＬＫ１ブロックと関連づけて記憶される。

そして、Ｓ１４０で、特定条件Ｓの成立頻度を成立頻度記憶部１５から読み出すと、その読み出した成立頻度が規定値より高いか否かを判定し（Ｓ１５０）、成立頻度が規定値以下であると判定するとＳ１６０へ移行し、一方、成立頻度が規定値より高いと判定するとＳ１７０へ移行する。尚、成立頻度記憶部１５に記憶されている成立頻度が、例えば「高い」、「低い」という頻度である場合には、Ｓ１５０では、成立頻度が「高い」か否かを判定するようにすればよい。

【００９４】

そして、Ｓ１６０では、コード記憶部１６から、図５（ａ）のテンプレート１を読み出し、そのテンプレート１に基づき、図２の制御モデル２０ａに対応するプログラムコードを生成する。具体的に、中間ファイルの情報及び生成ルール記憶部１４に記憶された生成ルールに基づき、図５（ａ）のテンプレート１におけるｏｕｔ，ｉｎ１，ｉｎ２，ｉｎ３，ｉｎ４に対して、それぞれ順に、図２の制御モデル２０ａのパラメータ名として、Ｃ，Ｅ，Ａ，Ｂ，Ｄを当てはめる。これにより、図３（ａ）のプログラムコードが生成される。そしてその後、Ｓ１１０へ戻ると共に、当該処理を終了する（Ｓ１１０：ＮＯ）。

【００９５】

一方、Ｓ１７０では、コード記憶部１６から、図５（ｂ）のテンプレート２を読み出し、そのテンプレート２に基づき、図２の制御モデル２０ａに対応するプログラムコードを生成する。パラメータ名の当てはめについては、前述の通りである。これにより、図３（ｂ）のプログラムコードが生成される。そしてその後、Ｓ１１０へ戻ると共に、当該処理を終了する（Ｓ１１０：ＮＯ）。

【００９６】

尚、コード記憶部１６に、図３（ａ），（ｂ）のプログラムコードがそれぞれ、テンプレート１，２として記憶されるようにし、Ｓ１６０、Ｓ１７０では、その記憶されたテンプレート１或いはテンプレート２、即ち、図３（ａ）或いは図３（ｂ）のプログラムコードを読み出すように構成してもよい。

【００９７】

このように、本装置では、図２の制御モデル２０ａについて、特定条件Ｓの成立頻度が規定値以下である場合には（Ｓ１５０：ＮＯ）、特定条件Ｓが成立する場合に追加処理（「Ｃ＝Ｄ」）が行われるようになる図３（ａ）のプログラムコードが生成される（Ｓ１６０）。つまり、特定条件Ｓの成立頻度が規定値以下である場合、その図３（ａ）のプログラムコードによれば、追加処理の実行頻度は低くなることとなる。

【００９８】

一方、本装置では、特定条件Ｓの成立頻度が規定値より高い場合には（Ｓ１５０：ＮＯ）、特定条件Ｓが成立しない場合（制御量Ａ≦制御量Ｂの条件が成立する場合）に追加処理（「Ｃ＝Ｅ」）が行われるようになる図３（ｂ）のプログラムコードが生成される（Ｓ１７０）。つまり、特定条件Ｓの成立頻度が規定値より高い場合、その図３（ｂ）のプログラムコードによれば、追加処理の実行頻度は低くなることとなる。

【００９９】

次に、図２の制御モデル２０ａとは別の制御モデルの例を用いて、説明する。

図７は、制御モデル２０（図１参照）の他の具体例を表す図面である。また、この図７の制御モデル２０ｂも、車両に搭載される電子制御装置が車両の各部を制御する場合の例である。

【０１００】

図７の制御モデル２０ｂは、まず、変数を表すブロックＡ，Ｂ，Ｃ，Ｄを有している。また、ブロックＡ，Ｂ，Ｃ，Ｄはそれぞれ、入力側（入力１，２，３，４）である。尚、

ブロックＡが表す変数（制御量）を、制御量Ａと記載する。ブロックＢ，Ｃ，Ｄについても同様とする。

【０１０１】

次に、制御モデル２０ｂは、制御モデル２０ａと同じＢＬＫ１ブロックを有している。また、制御モデル２０ｂにおいて、ブロックＡを始点としてＢＬＫ１ブロックのＢＬＫ第１ポートに到達する矢印と、ブロックＢを始点としてＢＬＫ１ブロックのＢＬＫ第２ポートに到達する矢印とが記述されている。つまり、ＢＬＫ１ブロックのＢＬＫ第１ポートに制御量Ａが入力され、ＢＬＫ１ブロックのＢＬＫ第２ポートに制御量Ｂが入力されて、ＢＬＫ１ブロックでは、制御量Ａ＞制御量Ｂの条件（特定条件Ｓ）が成立するか否かが判定される。そして、ＢＬＫ１ブロックにおいて、特定条件Ｓが成立する場合には、出力が「１」になり、特定条件Ｓが成立しない場合には、出力が「０」となる。

【０１０２】

また、制御モデル２０ｂは、大小を表す符号として「＜」が記載されたＢＬＫ２ブロックを有している。また、制御モデル２０ｂにおいて、ブロックＣを始点としてＢＬＫ２ブロックのＢＬＫ第１ポートに到達する矢印と、ブロックＤを始点としてＢＬＫ２ブロックのＢＬＫ第２ポートに到達する矢印とが記述されている。つまり、ＢＬＫ２ブロックのＢＬＫ第１ポートに制御量Ｃが入力され、ＢＬＫ２ブロックのＢＬＫ第２ポートに制御量Ｄが入力されて、ＢＬＫ２ブロックでは、制御量Ｃ＜制御量Ｄの条件（以下、特定条件Ｔと言う）が成立するか否かが判定される。そして、ＢＬＫ２ブロックにおいて、特定条件Ｔが成立する場合には、出力が「１」になり、特定条件Ｔが成立しない場合には、出力が「０」となる。

【０１０３】

さらに、制御モデル２０ｂは、「ＡＮＤ」が記載されたＡＮＤブロックを有している。このＡＮＤブロックには、入力ポートとして、第１ポートと第２ポートとがある。そして、このＡＮＤブロックにおいて、第１ポートの入力が「１」であり、かつ第２ポートの入力も「１」であれば、出力ポートの出力は「１」となる。一方、第１ポートの入力が「１」であっても、第２ポートの入力が「０」であれば出力は「０」になり、また、第１ポートの入力が「０」であれば出力は「０」となる。つまり、ＡＮＤブロックは、論理積演算を行うブロックである。

【０１０４】

そして、制御モデル２０ｂにおいては、ＢＬＫ１ブロックの出力ポートを始点としてＡＮＤ回路の第１ポートに到達する矢印と、ＢＬＫ２ブロックの出力ポートを始点としてＡＮＤ回路の第２ポートに到達する矢印とが記述されている。つまり、ＡＮＤブロックでは、ＢＬＫ１ブロック，ＢＬＫ２ブロックからの出力値に対し、上記のような論理積演算が行われる。

【０１０５】

さらに、制御モデル２０ｂにおいては、符号Ｘが付されたＸブロックが記述されている。このＸブロックには、入力ポートがあり、また、このＸブロック内には、所定の処理を表す記述がなされるようになっている。そして、Ｘブロックは、入力ポートの入力が「１」である間、Ｘブロック内に記述された処理を行うようになっている。逆に、入力ポートの入力が「０」であれば、その記述された処理は行わない。

【０１０６】

ここで、図７に示すように、制御モデル２０ｂでは、ＡＮＤブロックの出力ポートを始点としてＸブロックの入力ポートに到達する矢印が記述されている。また、Ｘブロック内には、ハイ信号を出力する処理を表す記述がされている。つまり、Ｘブロックは、ＡＮＤブロックの出力が「１」である間、ハイ信号を出力する処理（以下、処理Ｘと言う）を行う。尚、Ｘブロックは、ＡＮＤブロックの出力が「０」の間、ロー信号を出力する。

【０１０７】

以上のように構成された図７の制御モデル２０ｂは、次のような制御仕様を表す。つまり、「ロー信号を出力し、制御量Ａが制御量Ｂよりも大きい（制御量Ａ＞制御量Ｂ）とい

う条件が成立し、かつ、制御量Ｃが制御量Ｄよりも小さい（制御量Ｃ＜制御量Ｄ）という条件が成立すると、その間ハイ信号を出力する」という制御仕様（以下、第３の制御と言う）である。尚、このような制御モデル２０ｂは、上述した条件判定演算モデル部に相当する。つまり、ロー信号を出力する処理が所定の処理に相当し、制御量Ａ＞制御量Ｂかつ制御量Ｃ＜制御量Ｄが成立するか否かを判定する処理が判定処理に相当し、ハイ信号を出力する処理が追加処理に相当する。そして、本例では、その判定処理は、さらに、制御量Ａ＞制御量Ｂが成立するか否かの判定処理と、制御量Ｃ＜制御量Ｄが成立するか否かの判定処理とから構成され、本装置では、それぞれの判定処理についての成立頻度に基づき、プログラムコードが生成されるようになっている。

【０１０８】

そして、このような第３の制御に対応するプログラムコードが、図８（ａ）に示すプログラムコードである。つまり、図８（ａ）のプログラムコードは、制御量Ａ＞制御量Ｂかつ制御量Ｃ＜制御量Ｄが成立するか否かを判定し、成立すると判定すると（「ｉｆ（（Ａ＞Ｂ）＆＆（Ｃ＜Ｄ））」）、処理Ｘを実行する（「Ｘ（　）；」）ことを表すものである。

【０１０９】

一方、第３の制御仕様と同じ制御仕様を表すものとして、以下のような制御仕様が考えられる。つまり、「ロー信号を出力し、制御量Ｃが制御量Ｄよりも小さい（制御量Ｃ＜制御量Ｄ）という条件が成立し、かつ、制御量Ａが制御量Ｂよりも大きい（制御量Ａ＞制御量Ｂ）という条件が成立すると、その間ハイ信号を出力する」という制御（以下、第４の制御と言う）である。

【０１１０】

そして、このような第４の制御に対応するプログラムコードが、図８（ｂ）に示すプログラムコードである。つまり、図８（ｂ）のプログラムコードは、制御量Ｃ＜制御量Ｄかつ制御量Ａ＞制御量Ｂが成立するか否かを判定し、成立すると判定すると（「ｉｆ（（Ｃ＜Ｄ）＆＆（Ａ＞Ｂ））」）、処理Ｘを実行する（「Ｘ（　）；」）ことを表すものである。

【０１１１】

ところで、第３の制御では、特定条件Ｓ（制御量Ａ＞制御量Ｂ）が成立するか否かを判定する判定処理（以下、判定処理Ｓと言う）を実行し、判定処理Ｓにより特定条件Ｓが成立すると判定された場合に、特定条件Ｔ（制御量Ｃ＜制御量Ｄ）が成立するか否かを判定する判定処理（以下、判定処理Ｔと言う）を実行する。逆に、判定処理Ｓにより特定条件Ｓが成立しないと判定された場合は、判定処理Ｔは実行しない。このため、特定条件Ｓの成立頻度（以下、成立頻度Ｓと言う）が高い場合、判定処理Ｔの実行頻度は高くなる。逆に、成立頻度Ｓが低い場合、判定処理Ｔの実行頻度も低くなる。

【０１１２】

また一方、第４の制御では、判定処理Ｔにより特定条件Ｔが成立すると判定された場合に、判定処理Ｓを実行し、逆に、判定処理Ｔにより特定条件Ｔが成立しないと判定された場合は、判定処理Ｓは実行しない。このため、特定条件Ｔの成立頻度（以下、成立頻度Ｔと言う）が高い場合、判定処理Ｓの実行頻度は高くなる。逆に、成立頻度Ｔが低い場合、判定処理Ｓの実行頻度も低くなる。

【０１１３】

そして、成立頻度Ｓが成立頻度Ｔよりも高い場合、第３の制御と第４の制御とでは、第４の制御のほうが、２回目の判定処理（ここでは、判定処理Ｓ）の実行頻度を抑えられる。また、成立頻度Ｓが成立頻度Ｔよりも低い場合、第３の制御と第４の制御とでは、第３の制御のほうが、２回目の判定処理（ここでは、判定処理Ｔ）の実行頻度を抑えられる。尚、成立頻度Ｓと成立頻度Ｔが同じである場合には、第３の制御と第４の制御とでは、２回目の判定処理の実行頻度は同じである。

【０１１４】

そして、本実施形態のプログラムコード生成装置３は、図７の制御モデル２０ｂについ

て、成立頻度Ｓ及び成立頻度Ｔに応じて、２回目の判定処理の実行頻度が低くなるような、制御モデル２０ｂに対応するプログラムコードを生成する。尚、２回目の判定処理の実行頻度が低くなることは、処理Ｘ（追加処理に相当）の実行頻度が低くなることを意味する。以下、図４のフローチャートに基づき説明する。

【０１１５】

図７の制御モデル２０ｂが入力されると共に生成指令が入力され、図４の処理が開始されると、まず、生成対象のモデル部があると判定し（Ｓ１１０：ＹＥＳ）、生成対象が条件判定演算モデル部であると判定する（Ｓ１２０：ＹＥＳ）。そして、Ｓ１４０へ移行して、成立頻度Ｓ及び成立頻度Ｔをそれぞれ、成立頻度記憶部１５から読み出す。

【０１１６】

ここで、成立頻度Ｓ及び成立頻度Ｔは、図６に示す場合と同様に、予め、使用者により設定されるように構成されている。図９に、その構成を示す。

図９では、ＢＬＫ１ブロックとＢＬＫ２ブロックとのそれぞれに対して、成立頻度ＦＲＥＱを設定するためのパラメータ設定領域２３ｂが設けられている。

【０１１７】

そして、設定入力部５２を介して、ＢＬＫ１ブロックについてのパラメータ設定領域２３ｂに成立頻度Ｓを設定し、ＢＬＫ２ブロックについてのパラメータ設定領域２３ｂに成立頻度Ｔを設定できるようになっている。

【０１１８】

そして、パラメータ設定領域２３ｂに設定された成立頻度Ｓ及び成立頻度Ｔは、成立頻度ＳがＢＬＫ１ブロックと関連づけられ、成立頻度ＴがＢＬＫ２ブロックと関連づけられて、成立頻度記憶部１５に記憶される。

【０１１９】

そして、Ｓ１４０で、成立頻度Ｓ及び成立頻度Ｔを成立頻度記憶部１５から読み出すと、次のＳ１５０では、この例では、成立頻度Ｓが成立頻度Ｔより高いか否かを判定する。そして、成立頻度Ｓが成立頻度Ｔ以下であると判定すると（Ｓ１５０：ＮＯ）、Ｓ１６０へ移行し、一方、成立頻度Ｓが成立頻度Ｔより高いと判定すると（Ｓ１５０：：ＹＥＳ）、Ｓ１７０へ移行する。

【０１２０】

ここで、コード記憶部１６には、図１０に示すようなテンプレートが記憶されている。図１０（ａ）のテンプレート１は、「ｉｎ１＞ｉｎ２かつｉｎ３＜ｉｎ４の場合、処理Ｘを実行する」というプログラムコードを表すテンプレートであり、前述の第３の制御に対応する。また、図１０（ｂ）のテンプレート２は、「ｉｎ３＜ｉｎ４かつｉｎ１＞ｉｎ２の場合、処理Ｘを実行する」というプログラムコードを表すテンプレートであり、前述の第４の制御に対応する。尚、ｉｎ１，ｉｎ２，ｉｎ３，ｉｎ４にはそれぞれ、具体的なパラメータ名が記述される。

【０１２１】

そして、Ｓ１６０では、コード記憶部１６から、図１０（ａ）のテンプレート１を読み出し、そのテンプレート１に基づき、図７の制御モデル２０ｂに対応するプログラムコードを生成する。具体的に、中間ファイルの情報及び生成ルール記憶部１４に記憶された生成ルールに基づき、図１０（ａ）のテンプレート１におけるｉｎ１，ｉｎ２，ｉｎ３，ｉｎ４に対して、それぞれ順に、図７の制御モデル２０ｂのパラメータ名として、Ａ，Ｂ，Ｃ，Ｄを当てはめる。これにより、図８（ａ）のプログラムコードが生成される。そしてその後、Ｓ１１０へ戻ると共に、当該処理を終了する（Ｓ１１０：ＮＯ）。

【０１２２】

一方、Ｓ１７０では、コード記憶部１６から、図１０（ｂ）のテンプレート２を読み出し、そのテンプレート２に基づき、図７の制御モデル２０ａに対応するプログラムコードを生成する。パラメータ名の当てはめについては、前述の通りである。これにより、図８（ｂ）のプログラムコードが生成される。そしてその後、Ｓ１１０へ戻ると共に、当該処理を終了する（Ｓ１１０：ＮＯ）。

【０１２３】
尚、コード記憶部１６に、図８（ａ），（ｂ）のプログラムコードがそれぞれ、テンプレート１，２として記憶されるようにし、Ｓ１６０、Ｓ１７０では、その記憶されたテンプレート１或いはテンプレート２、即ち、図８（ａ）或いは図８（ｂ）のプログラムコードを読み出すように構成してもよい。
【０１２４】
このように、本装置では、図７の制御モデル２０ｂについて、成立頻度Ｓが成立頻度Ｔ以下である場合には（Ｓ１５０：ＮＯ）、まず判定処理Ｓが実行され、判定処理Ｓで特定条件Ｓが成立すると判定されると、次に判定処理Ｔが実行されるようになる図８（ａ）のプログラムコードが生成される（Ｓ１６０）。つまり、成立頻度Ｓが成立頻度Ｔ以下である場合、その図８（ａ）のプログラムコードによれば、２回目の判定処理Ｔの実行頻度が低くなることとなる。また、処理Ｘの実行頻度も低くなることとなる。
【０１２５】
一方、本装置では、成立頻度Ｓが成立頻度Ｔより高い場合には（Ｓ１５０：：ＹＥＳ）、まず判定処理Ｔが実行され、判定処理Ｔで特定条件Ｔが成立すると判定されると、次に判定処理Ｓが実行されるようになる図８（ｂ）のプログラムコードが生成される（Ｓ１６０）。つまり、成立頻度Ｓが成立頻度Ｔより高い場合、その図８（ｂ）のプログラムコードによれば、２回目の判定処理Ｓの実行頻度が低くなることとなる。また、処理Ｘの実行頻度も低くなることとなる。
【０１２６】
尚、本実施形態において、抽出エンジン１１が制御モデル取得手段に相当し、コード生成処理部１２が生成手段に相当し、Ｓ１１０及びＳ１２０の処理がモデル解析手段に相当し、Ｓ１４０の処理が成立頻度取得手段に相当し、Ｓ１５０の処理が成立頻度判定手段に相当し、コード記憶部１６がコード記憶手段に相当し、設定入力部５２が入力手段に相当し、成立頻度記憶部１５が成立頻度記憶手段に相当している。
【０１２７】
以上のように、本実施形態によれば、例えば、制御モデル２０ａに対応するプログラムコードとして、追加処理の実行頻度が低くなるように、特定条件Ｓの成立頻度が規定値より高い場合に追加処理の実行頻度が低くなるプログラムコードと、特定条件Ｓの成立頻度が規定値以下である場合に追加処理の実行頻度が低くなるプログラムコードとの何れかが、特定条件Ｓの実行頻度に応じて生成される。
【０１２８】
一方、追加処理の実行頻度が低くなるようなプログラムコードがＥＣＵに実装された場合、そのＥＣＵにおいて、追加処理の実行頻度が低くなる分、プログラムの実行時間や処理負荷が低減されるため有利となる。
【０１２９】
また、本実施形態によれば、例えば、制御モデル２０ｂに対応するプログラムコードとして、２回目の判定処理の実行頻度が低くなるように、特定条件Ｓの成立頻度Ｓが特定条件Ｔの成立頻度Ｔより高い場合に２回目の判定処理の実行頻度が低くなるプログラムコードと、特定条件Ｓの成立頻度Ｓが特定条件Ｔの成立頻度Ｔ以下である場合に２回目の判定処理の実行頻度が低くなるプログラムコードとの何れかが、成立頻度Ｓ及び成立頻度Ｔに応じて生成される。尚、２回目の判定処理の実行頻度が低くなると、２回目の判定処理の後の処理Ｘの実行頻度も低くなることとなる。
【０１３０】
一方、２回目の判定処理の実行頻度が低くなるようなプログラムコードがＥＣＵに実装された場合、そのＥＣＵにおいて、２回目の判定処理や処理Ｘの実行頻度が低くなる分、プログラムの実行時間や処理負荷が低減されるため有利となる。
【０１３１】
そして、本実施形態のプログラムコード生成装置３によれば、ＥＣＵに実装した際、実行時間や処理負荷の点で有利となるプログラムコードを、容易に生成できるようになる。

また、本実施形態によれば、制御モデル２０ａ，２０ｂに対応するプログラムコードのテンプレートがそれぞれ、コード記憶部１６に記憶されており、コード生成処理部１２は、特定条件の成立頻度に応じて、コード記憶部１６に記憶されたテンプレートを読み出して、その読み出したテンプレートに基づき、プログラムコードを生成する。つまり、コード生成処理部１２は、テンプレートに基づきプログラムコードを生成すればよく、これによれば、制御モデル２０ａ，２０ｂに基づきプログラムコードを生成する場合と比較して、プログラムコード生成装置３の処理負荷を抑えることができる。

【０１３２】

そして、本実施形態によれば、制御モデル２０ａ，２０ｂに限られず、モデル入力部５１を介して入力される制御モデル２０について、特定条件の成立頻度に応じて、追加処理の実行頻度が低くなるようなその制御モデル２０に対応するプログラムコードが生成されるようにすることができる。また、その制御モデル２０に対応するプログラムコードのテンプレートをコード記憶部１６に記憶させておき、特定条件の成立頻度に応じて、コード記憶部１６に記憶されたテンプレートが読み出され、その読み出されたテンプレートに基づき、プログラムコードが生成されるようにすることができる。

【０１３３】

また、本実施形態において、制御量Ａ＞制御量Ｂの条件、及び制御量Ａ≦制御量Ｂの条件は、請求項９の互いに排他的な複数通りの条件に相当する。そして、制御モデル２０ａは、請求項９の特定演算モデル部に相当する。つまり、本実施形態は、請求項９～１１にも対応するものである。

【０１３４】

そして、抽出エンジン１１は請求項９の制御モデル取得手段にも相当し、コード生成処理部１２は請求項９の生成手段にも相当し、Ｓ１１０及びＳ１２０の処理は請求項９のモデル解析手段にも相当し、Ｓ１４０及びＳ１５０の処理は請求項９の条件取得手段にも相当し、コード記憶部１６は請求項１０のコード記憶手段にも相当する。

［第２実施形態］

次に、本発明の第２実施形態について説明する。

【０１３５】

本第２実施形態のプログラムコード生成装置３は、第１実施形態のプログラムコード生成装置３と同じ構成である（図１参照）。また、図４の処理が実行される点も同じである。

【０１３６】

一方、本第２実施形態のプログラムコード生成装置３は、特定条件（例えば特定条件Ｓ，Ｔ）が成立するか否かをシミュレーションにより検出し、その検出結果に基づき、成立頻度を取得するようになっている点が第１実施形態と異なっている。以下、図２の制御モデル２０ａに場合について具体的に説明する。

【０１３７】

まず、図１１は、制御モデル２０ａについて、パラメータ設定領域２３ａが設けられた構成を表す図面である。そして、本第２実施形態の場合、図１１に示すように、パラメータ設定領域２３ａ内には、特定条件Ｓの成立回数ＣＮＴＴと不成立回数ＣＮＴＦとがそれぞれ記述される構成となっている。

【０１３８】

具体的に、特定条件Ｓの成立回数及び不成立回数がシミュレーションにより検出され、その検出された成立回数の値が成立回数ＣＮＴＴとして、また、検出された不成立回数の値が不成立回数ＣＮＴＦとして、それぞれパラメータ設定領域２３ａ内に記述される。加えて、成立頻度記憶部１５に、ＢＬＫ１ブロックと関連づけて、成立回数ＣＮＴＴと不成立回数ＣＮＴＦとが記憶される。以下、説明する。

【０１３９】

図１２は、シミュレーション処理部１３が行う処理の流れを表すフローチャートである。この処理では、制御モデル２０（本例では、制御モデル２０ａ）に対応する制御プログ

ラムのシミュレーションが実行されると共に、シミュレーション結果を表すログデータが解析されて、特定条件（本例では、特定条件Ｓ）の成立回数及び不成立回数が検出される。そして、以下のタイミングで実施される。

【０１４０】

まず、使用者により、制御モデル２０がモデル入力部５１を介して入力され、また、開始指令入力部５３を介してシミュレーションの実行を指令する実行指令が、コード生成ツール１０に入力されると、抽出エンジン１１が、制御モデル２０を読み込んで、図示しない処理にて制御モデル２０に対応する制御プログラムの情報を含む中間ファイルを出力する。そして、その中間ファイルは、シミュレーション処理部１３に入力されるようになっており、シミュレーション処理部１３は、その中間ファイルが入力されると、図１２の処理を開始する。尚、使用者により制御モデル２０が入力されると、自動的に、実行指令がコード生成ツール１０に入力されるように構成してもよい。

【０１４１】

図１２の処理では、まず、Ｓ２１０にて、制御モデル２０ａに対応する制御プログラムをシミュレーションする。つまり、その制御プログラムをソフトウエア上で実行する。尚、このＳ２１０において、シミュレーションにより得られたシミュレーション結果を表すログデータは、図示しないＲＡＭに一旦記憶されると共に、ハードディスク装置に記憶される。

【０１４２】

そして、次にＳ２２０では、シミュレーションを終了するか否かを判定する。具体的に、シミュレーションを実行する回数が予め定められており、このＳ２２０では、シミュレーションが、予め定められた回数（規定回数）実行されたか否かを判定する。

【０１４３】

そして、シミュレーションが規定回数実行されていないと判断すると、シミュレーションを終了しないと判定して（Ｓ２２０：ＮＯ）、再びＳ２１０へ戻り、シミュレーションを実行する。

【０１４４】

一方、Ｓ２２０で、シミュレーションが規定回数実行されたと判断すると、シミュレーションを終了すると判定して（Ｓ２２０：ＹＥＳ）、Ｓ２３０へ移行する。Ｓ２３０では、以下の処理を行う。

【０１４５】

シミュレーション処理部１３では、シミュレーションにより検出される特定条件Ｓの成立回数ＮＴと不成立回数ＮＦとがそれぞれ、図示しないカウンタによりカウントされるようになっており、このＳ２３０では、そのカウンタにおける成立回数ＮＴの値及び不成立回数ＮＦの値を０（初期値）にする。尚、成立回数ＮＴ及び不成立回数ＮＦが、ＲＡＭ等の所定の格納領域に更新記憶されるように構成してもよい。

【０１４６】

そして、Ｓ２３０の次は、Ｓ２４０へ進み、Ｓ２１０で得られたログデータのうち、解析すべきログデータ、言い換えると、解析していないログデータがあるか否かを判定する。そして、解析すべきログデータがあると判定すると（Ｓ２４０：ＹＥＳ）、Ｓ２５０へ移行し、その解析すべきログデータを解析して、そのログデータから、特定条件Ｓが成立したか否かを表すデータ（以下、成立頻度検出データと言う）を読み込む。また、成立頻度検出データの値は、０，１の何れかであり、０は特定条件Ｓが成立していないことを表し、１は特定条件Ｓが成立したことを表す。

【０１４７】

そして、Ｓ２５０で成立頻度検出データを読み込むと、Ｓ２６０へ進み、その読み込んだ成立頻度検出データの値が１であるか否かを判定する。そして、１であると判定すると（Ｓ２６０：ＹＥＳ）、特定条件Ｓが成立したと判断してＳ２７０へ移行し、成立回数ＮＴの値を１増加させる。そして、再びＳ２４０へ戻る。

【０１４８】

一方、Ｓ２６０で成立頻度検出データが１でない、即ち、０であると判定すると（Ｓ２６０：ＮＯ）、特定条件Ｓが成立していないと判断してＳ２８０へ移行し、不成立回数ＮＦの値を１増加させる。そして、再びＳ２４０へ戻る。

【０１４９】

また、Ｓ２４０で解析すべきログデータがないと判定すると（Ｓ２４０：ＮＯ）、Ｓ２９０へ移行し、Ｓ２３０～Ｓ２７０で検出された成立回数ＮＴの値を、成立回数ＣＮＴＴとして設定し、Ｓ２３０～Ｓ２６０及びＳ２８０で検出された不成立回数ＮＦの値を、不成立回数ＣＮＴＦとして設定する。尚、設定するとは、前述のように、成立回数ＣＮＴＴ及び不成立回数ＣＮＴＦを、パラメータ設定領域２３ａ内に記述すると共に、成立頻度記憶部１５に、ＢＬＫ１ブロックと関連づけて記憶させることである。そしてその後、当該処理を終了する。

【０１５０】

そして、本第２実施形態において、図４の処理は、この図１２の処理が終了し、使用者により、開始指令入力部５３を介して前述の生成指令が入力されると開始される。尚、図１２の処理が終了すると、図４の処理が自動で開始されるように構成してもよい。

【０１５１】

そして、本第２実施形態の図４のＳ１４０では、成立頻度記憶部１５から成立回数ＣＮＴＴ及び不成立回数ＣＮＴＦを読み出して、その読み出した成立回数ＣＮＴＴ及び不成立回数ＣＮＴＦに基づき、成立頻度を取得する。例えば成立回数ＣＮＴＴが３００（回）であり、不成立回数ＣＮＴＦが１８０（回）であれば、成立頻度は３００／（３００＋１８０）＝６２．５％と算出することができる。尚、成立回数ＣＮＴＴと不成立回数ＣＮＴＦの大小を比較し、成立頻度が「高い」或いは「低い」と算出するようにしてもよい。

【０１５２】

尚、本第２実施形態において、シミュレーション処理部１３及びＳ２４０～Ｓ２９０の処理が検出手段に相当している。

以上のように、本第２実施形態によれば、シミュレーション処理部１３が、制御モデル２０ａに対応する制御プログラムについてシミュレーションを実行すると共に、特定条件Ｓの成立回数及び不成立回数を検出し、コード生成処理部１２は、その検出結果に基づき、特定条件Ｓの成立頻度を取得する。このため、使用者が制御モデル２０ａに対して成立頻度を設定したりしなくてもよく、成立頻度が自動で容易に得られるようにすることができ、使用者にとって使い勝手がよくなる。また、シミュレーションを繰り返し実行するようにすれば、特定条件Ｓが成立するか否かを表すデータを多数取得できることとなり、これによれば、シミュレーションで得られる成立頻度の信頼性を向上させることができる。このため、追加処理の実行頻度が低くなるプログラムコードが確実に生成されるようにすることができる。

［実施形態３］

次に、本発明の第３実施形態について説明する。

【０１５３】

図１３は、本第３実施形態のプログラムコード生成装置３の構成図である。

本第３実施形態のプログラムコード生成装置３は、第１実施形態のプログラムコード生成装置３（図１参照）と比較して、生成ルール記憶部１４に、モデル記憶部１７を備えている点が異なっている。

【０１５４】

モデル記憶部１７は、制御モデル２０を表すためのモデルのテンプレートを記憶するものである。具体的に、モデル記憶部１７には、図１４及び図１５に示すようなモデルのテンプレートが記憶されている。

【０１５５】

図１４（ａ）のテンプレート１は、前述した第１の制御に対応する制御モデルのテンプレートであり、図１４（ｂ）のテンプレート２は、前述した第２の制御に対応する制御モデルのテンプレートである。

【０１５６】
ここで、図１４（ａ）は、図１７（ａ）と同じであるため、ここでは詳しい説明を省略する。一方、図１４（ｂ）について説明すると、図１４（ａ）と異なり、ＢＬＫ１ブロックには、大小を表す符号として、「≦」が記述され、また、Ｓｗｉｔｃｈブロックの第１ポートには定数Ｅが入力され、Ｓｗｉｔｃｈブロックの第３ポートには定数Ｄが入力されるように構成されている。この構成によれば、「制御量Ｃに定数Ｄを代入する。ただし、制御量Ａ≦制御量Ｂという条件が成立する場合は、制御量Ｃに定数Ｅを代入する」という前述の第２の制御が実現される。
【０１５７】
また、図１５（ａ）のテンプレート１は、前述した第３の制御に対応する制御モデルのテンプレートであり、図１５（ｂ）のテンプレート２は、前述した第４の制御に対応する制御モデルのテンプレートである。
【０１５８】
ここで、図１５（ａ）は、図７と同じであるため、ここでは詳しい説明を省略する。一方、図１５（ｂ）について説明すると、図１５（ａ）と異なり、ＢＬＫ１ブロックの出力値はＡＮＤ回路の第２ポートに入力され、ＢＬＫ２ブロックの出力値はＡＮＤ回路の第１ポートに入力されるように構成されている。この構成によれば、「ロー信号を出力し、制御量Ｃ＜制御量Ｄという条件が成立し、かつ、制御量Ａ＞制御量Ｂという条件が成立する間は、ハイ信号を出力する」という前述の第４の制御が実現される。
【０１５９】
そして、本第３実施形態では、コード生成処理部１２は、図４の処理に代えて、図１６の処理を実行する。尚、この図１６の処理が開始されるタイミングは、図４の処理が実行される場合と同じであり、つまり、制御モデル２０が入力されると共に生成指令が入力され、抽出エンジン１１から所定の中間ファイルがコード生成処理部１２に入力されたタイミングである。
【０１６０】
図１６の処理では、まずＳ３１０にて、抽出エンジン１１からの中間ファイルに基づき、プログラムコードの生成対象のモデル部を検索してそのモデル部があるか否かを判定し、生成対象のモデル部がないと判定すると（Ｓ３１０：ＮＯ）、そのまま当該処理を終了する。一方、対象のモデル部があると判定すると（Ｓ３１０：ＹＥＳ）、Ｓ３２０へ移行し、その生成対象のモデル部が、条件判定演算モデル部であるか否かを判定する。
【０１６１】
Ｓ３２０で、条件判定演算モデル部でないと判定すると（Ｓ３２０：ＮＯ）、Ｓ３８０へ移行し、その条件判定演算モデル部でないと判定したモデル部について、プログラムコードを生成する。尚、Ｓ３８０の処理の内容については、ここでは詳しい説明を省略する。
【０１６２】
一方、Ｓ３２０で、条件判定演算モデル部であると判定すると（Ｓ３２０：：ＹＥＳ）、Ｓ３３０へ移行する。Ｓ３３０では、その条件判定演算モデル部における判定処理で判定される条件の成立頻度を取得する。尚、成立頻度は、図６を用いて前述したように、予め、使用者により設定入力部５２を介して入力され、成立頻度記憶部１５に記憶される。そして、Ｓ３３０では、成立頻度記憶部１５から、Ｓ３１０で検索された条件判定演算モデル部に対応する成立頻度を読み出す。
【０１６３】
また、図１１及び図１２を用いて説明したように構成してもよい。つまり、シミュレーション処理部１３が、入力された制御モデル２０に対応する制御プログラムのシミュレーションを実行すると共に、条件判定演算モデル部で判定される条件の成立回数及び不成立回数を検出し、その検出値を、成立頻度記憶部１５に記憶させる。そして、Ｓ３３０では、その検出値を成立頻度記憶部１５から読み出すと共に、その読み出した検出値に基づき、成立頻度を取得するように構成してもよい。

【０１６４】
そして、Ｓ３３０の後、Ｓ３４０へ移行し、取得した成立頻度が規定値より高いか否かを判定する。そして、成立頻度が規定値以下であると判定すると（Ｓ３４０：ＮＯ）、Ｓ３５０へ移行し、一方、成立頻度が規定値より高いと判定すると（Ｓ３４０：：ＹＥＳ）、Ｓ３６０へ移行する。
【０１６５】
Ｓ３５０では、モデル記憶部１７から、テンプレート１（図１４（ａ）或いは図１５（ａ））を読み出すと共に、その後、Ｓ３７０へ移行し、その読み出したテンプレート１を、表示部４０に表示させる。この場合、読み出したテンプレート１が、入力された制御モデル２０の一部を構成するものである場合、その入力された制御モデル２０のテンプレート１に対応する部分とＳ３５０で読み出したテンプレート１とを置き換えて、その置き換えを行った後の制御モデル２０全体を、表示部４０に表示させるようにすればよい。また、入力された制御モデル２０と、読み出したテンプレート１とをそれぞれ別々に、１画面上に表示させるようにしてもよい。さらに、読み出したテンプレート１のみを表示させるようにしてもよい。テンプレート２についても同様である。
【０１６６】
そして、Ｓ３７０の処理の後、Ｓ３８０へ移行し、Ｓ３７０で表示部４０に表示させたテンプレート１に対応するプログラムコードを生成する。そして再び、Ｓ３１０へ戻る。
一方、Ｓ３６０では、モデル記憶部１７から、テンプレート２（図１４（ｂ）或いは図１５（ｂ））を読み出すと共に、その後、Ｓ３７０へ移行し、その読み出したテンプレート２を、表示部４０に表示させる。そして、Ｓ３８０へ移行し、Ｓ３７０で表示部４０に表示させたテンプレート２に対応するプログラムコードを生成する。そして再び、Ｓ３１０へ戻る。
【０１６７】
次に、具体例を用いて説明する。
まず、図２の制御モデル２０ａの場合について説明する。
制御モデル２０ａが入力されると共に生成指令が入力されて、図１６の処理が開始されると、生成対象のモデル部があると判定し（Ｓ３１０：ＹＥＳ）、また、その生成対象のモデル部が、条件判定演算モデル部であると判定する（Ｓ３２０：：ＹＥＳ）。
【０１６８】
そして、成立頻度記憶部１５から、制御モデル２０ａにおいて判定される特定条件Ｓの成立頻度を取得し（Ｓ３３０）、その後、取得した成立頻度が規定値より高いか否かを判定する（Ｓ３４０）。尚、規定値は、予め使用者等により設定され、ＲＯＭやハードディスク装置等の所定の領域に記憶されるが、例えば５０％（５割）と設定しておくことが考えられる。
【０１６９】
そして、成立頻度が規定値以下であると判定すると（Ｓ３４０：ＮＯ）、モデル記憶部１７から、図１４（ａ）のテンプレート１を読み出す（Ｓ３５０）。このテンプレート１は、前述のように、第１の制御を表すものであり、第１の制御によれば、特定条件Ｓ（制御量Ａ＞制御量Ｂ）の成立頻度が低い（規定値以下）の場合、「Ｃ＝Ｄ」の追加処理、即ち、ＣにＤを代入し直すという追加処理の実行頻度が低くなる。
【０１７０】
そしてその後、読み出した図１４（ａ）のテンプレート１を、表示部４０に表示させると共に（Ｓ３７０）、その表示させたテンプレート１に対応するプログラムコードを生成する（Ｓ３８０）。これによれば、図３（ａ）に示すプログラムコードが生成される。そしてその後、再びＳ３１０へ戻る。
【０１７１】
一方、Ｓ３４０で成立頻度が規定値より高いと判定すると（Ｓ３４０：：ＹＥＳ）、モデル記憶部１７から、図１４（ｂ）のテンプレート２を読み出す（Ｓ３６０）。このテンプレート２は、前述のように、第２の制御を表すものであり、第２の制御によれば、特定

条件Ｓ（制御量Ａ＞制御量Ｂ）の成立頻度が高い（規定値より高い）場合、「Ｃ＝Ｅ」の追加処理、即ち、ＣにＥを代入し直すという追加処理の実行頻度が低くなる。

【０１７２】

そしてその後、読み出した図１４（ｂ）のテンプレート２を、表示部４０に表示させると共に（Ｓ３７０）、そのテンプレート２に対応するプログラムコードを生成する（Ｓ３８０）。これによれば、図３（ｂ）に示すプログラムコードが生成される。そしてその後、再びＳ３１０へ戻る。

【０１７３】

つまり、本第３実施形態のプログラムコード生成装置３によれば、第１及び第２実施形態のプログラムコード生成装置３と同様に、制御モデル２０ａについて、特定条件Ｓの成立頻度が規定値以下である場合には、図３（ａ）のプログラムコードが生成され、特定条件Ｓの成立頻度が規定値より高い場合には、図３（ｂ）プログラムコードが生成される。つまり、追加処理の実行頻度が低くなるようなプログラムコードが生成される。

【０１７４】

次に、図７の制御モデル２０ｂの場合について説明する。

制御モデル２０ｂが入力されると共に生成指令が入力されて、図１６の処理が開始されると、生成対象のモデル部があると判定し（Ｓ３１０：ＹＥＳ）、また、その生成対象のモデル部が、条件判定演算モデル部であると判定する（Ｓ３２０：：ＹＥＳ）。

【０１７５】

そして、成立頻度記憶部１５から、制御モデル２０ｂで判定される特定条件Ｓの成立頻度Ｓと、特定条件Ｔの成立頻度Ｔとを読み出し（Ｓ３３０）、その後、本例では、成立頻度Ｓが成立頻度Ｔより高いか否かを判定する。

【０１７６】

そして、成立頻度Ｓが成立頻度Ｔ以下であると判定すると（Ｓ３４０：ＮＯ）、モデル記憶部１７から、図１５（ａ）のテンプレート１を読み出す。このテンプレート１は、前述のように、第３の制御を表すものであり、第３の制御によれば、成立頻度Ｓが成立頻度Ｔ以下である場合、２回目の判定処理（判定処理Ｔ）の実行頻度が低くなる。

【０１７７】

そしてその後、読み出した図１５（ａ）のテンプレート１を、表示部４０に表示させると共に（Ｓ３７０）、その表示させたテンプレート１に対応するプログラムコードを生成する（Ｓ３８０）。これによれば、図８（ａ）に示すプログラムコードが生成される。そしてその後、再びＳ３１０へ戻る。

【０１７８】

一方、Ｓ３４０で成立頻度Ｓが成立頻度Ｔより高いと判定すると（Ｓ３４０：：ＹＥＳ）、Ｓ３６０へ移行し、モデル記憶部１７から、図１５（ｂ）のテンプレート２を読み出す。このテンプレート２は、前述のように、第４の制御を表すものであり、第４の制御によれば、成立頻度Ｓが成立頻度Ｔより高い場合、２回目の判定処理（判定処理Ｓ）の実行頻度が低くなる。

【０１７９】

そしてその後、Ｓ３７０へ移行し、読み出した図１５（ｂ）のテンプレート２を、表示部４０に表示させると共に、Ｓ３８０へ移行し、そのテンプレート２に対応するプログラムコードを生成する。これによれば、図８（ｂ）に示すプログラムコードが生成される。そしてその後、再びＳ３１０へ戻る。

【０１８０】

つまり、本第３実施形態のプログラムコード生成装置３によれば、第１及び第２実施形態のプログラムコード生成装置３と同様に、制御モデル２０ｂについて、成立頻度Ｓが成立頻度Ｔ以下である場合には、図８（ａ）のプログラムコードが生成され、成立頻度Ｓが成立頻度Ｔより高い場合には、図８（ｂ）プログラムコードが生成される。つまり、２回目の判定処理の実行頻度が低くなるようなプログラムコードが生成される。また、２回目の判定処理の実行頻度が低くなることは、２回目の判定処理の後の処理Ｘ（追加処理に相

当）の実行頻度が低くなることを意味する。

【０１８１】

尚、本第３実施形態において、モデル記憶部１７がモデル記憶手段に相当し、表示部４０が表示手段に相当し、Ｓ３７０の処理が画像表示制御手段に相当している。

以上のように、本第３実施形態によれば、特定条件の成立頻度に応じて、モデル記憶部１７から、モデルのテンプレートが読み出されて、その読み出されたモデルのテンプレートが、表示部４０に表示されるようになっている。このため、使用者は、どのようなモデルに基づきプログラムコードが生成されるのかが、視覚で認識できるようになる。よって、使用者にとって使い勝手がよくなる。例えば、入力した制御モデルと、その入力した制御モデルに基づき作成されたプログラムコードが対応しないというような不都合が生じることを防止することができる。

［実施形態４］

次に、本発明の第４実施形態について説明する。

【０１８２】

第４実施形態のプログラムコード生成装置の構成は、第３実施形態のプログラムコード生成装置３と同じ構成である（図１３参照）。

そして、本第４実施形態では、少なくとも３つ以上の複数種類のプログラムの何れでも表すことができる制御モデル（以下、特定演算モデル部）のプログラムコードを生成するようになっている。ここでは、特定演算モデル部が、車両制御用の制御プログラムを表し、特に、所定の制御量（例えば燃料噴射量やスロットル開度等）を補正するための補正値をエンジンの回転数に応じて算出するためのマップデータを、そのマップデータが記憶された記憶領域から読み出す処理の制御プログラムを表す場合について説明する。

【０１８３】

本第４実施形態の例では、エンジンの回転数が高回転の時に用いるマップデータ（以下、第１マップデータと言う）と、中回転の時に用いるマップデータ（以下、第２マップデータと言う）と、低回転の時に用いるマップデータ（以下、第３マップデータと言う）とがあり、特定演算モデル部は、その第１～３のマップデータを読み出す処理の制御プログラムを表すものである。尚、この特定演算モデル部が表す制御プログラムにより読み出されたマップデータは、所定の格納領域に格納されると共に、特定の補正実施条件が成立すると、別のプログラムで読み込まれ、その時のエンジン回転数に応じた補正値を算出するために使用される。

【０１８４】

以下、どのように特定演算モデル部に対応するプログラムコードを生成するかについて説明する。

まず、使用者により、特定演算モデル部を含む制御モデル２０（図１３参照）がモデル入力部５１を介して入力され、また、開始指令入力部５３を介してシミュレーションの実行を指令する実行指令が、コード生成ツール１０に入力されると、抽出エンジン１１が、制御モデル２０を読み込んで、図示しない処理にて制御モデル２０に対応する制御プログラムの情報を含む中間ファイルを出力する。そして、その中間ファイルは、シミュレーション処理部１３に入力されるようになっており、シミュレーション処理部１３は、その中間ファイルが入力されると、その中間ファイルに基づき、制御モデル２０に対応する制御プログラムをシミュレーションする。尚、使用者により制御モデル２０が入力されると、自動的に、実行指令がコード生成ツール１０に入力されるように構成してもよい。また、このシミュレーションの際、エンジンの実機が実際に接続され、そのエンジンの回転数のデータが利用される。

【０１８５】

ここで、まず、そのシミュレーションを含め、以下に説明する処理は、シミュレーション処理部１３が行う処理である。

ところで、シミュレーションにより得られたシミュレーション結果を表すログデータは、図示しないＲＡＭに一旦記憶されると共に、ハードディスク装置に記憶される。また、

シミュレーションは、予め定められた回数実行される。

【０１８６】

そして、シミュレーション処理部１３は、シミュレーション結果を表すログデータから、前記補正実施条件成立時のエンジン回転数について、高回転、中回転、低回転となる割合を算出する。言い換えると、高回転となる頻度、中回転となる頻度、低回転となる頻度をそれぞれ算出する。

【０１８７】

次に、最も成立の頻度が高くなるエンジンの回転状態の情報（以下、回転状態情報と言う）を取得する。例えば、高回転となる割合（頻度）が最も高ければ、「高回転」である旨の情報を取得する。尚、回転状態情報は、図示しないＲＡＭ等に記憶される。

【０１８８】

以上が、シミュレーション処理部１３が行う処理である。

次に説明する処理は、コード生成処理部１２が行う処理である。

コード生成処理部１２は、特定演算モデル部について、回転状態情報に基づき、実行所要時間が最も短くなるようなプログラムコードを生成する処理を行う。

【０１８９】

具体的に、回転状態情報をＲＡＭ等から読み出すと共に、その回転状態情報が表す回転状態（高回転、中回転、低回転の何れか）の時に用いるマップデータ（第１～第３マップデータの何れか）を、最初に読み出すようになるプログラムコードを生成する。

【０１９０】

例えば、回転状態情報が「高回転」を表す場合、高回転の時に用いる第１マップデータを最初に読み出すようなプログラムコードを生成する。また、第１マップデータを読み出した後は、第２マップデータ、第３マップデータの順で読み出すように構成する。尚、第２，第３マップデータについて、第３マップデータ、第２マップデータの順で読み出すように構成してもよい。

【０１９１】

また、回転状態情報が「中回転」を表す場合、中回転の時に用いる第２マップデータを最初に読み出すようなプログラムコードを生成する。また、第２マップデータを読み出した後は、第１マップデータ、第３マップデータの順で読み出すように構成する。尚、第１，第３マップデータについて、第３マップデータ、第１マップデータの順で読み出すように構成してもよい。

【０１９２】

さらに、回転状態情報が「低回転」を表す場合、低回転の時に用いる第３マップデータを最初に読み出すようなプログラムコードを生成する。また、第３マップデータを読み出した後は、第１マップデータ、第２マップデータの順で読み出すように構成する。尚、第１，第２マップデータについて、第２マップデータ、第１マップデータの順で読み出すように構成してもよい。

【０１９３】

以上のように、コード生成処理部１２は、シミュレーション処理部１３の処理により得られた回転状態情報に基づき、実行所要時間が最も短くなるプログラムコードを生成する。

【０１９４】

このような構成によれば、最も成立の頻度が高くなるエンジンの回転状態の時に使用されるマップデータが、まず最初に読み出されるようなプログラムコードを生成することができる。つまり、補正値を算出する際に使用しないマップデータを最初に読み出してしまう、ということが生じる頻度を抑えることができる。このため、実行所要時間や処理負荷を抑えることができるプログラムコードを生成できるようになる。

【０１９５】

ところで、第１マップデータが最初に読み出されるようになるプログラムコード（以下、第１プログラムコードと言う）、第２マップデータが最初に読み出されるようになるプ

ログラムコード（以下、第２プログラムコードと言う）、及び第３マップデータが最初に読み出されるようになるプログラムコード（以下、第３プログラムコードと言う）をそれぞれ、コード記憶部１６に記憶しておき、回転状態情報に基づいて、そのコード記憶部１６から、第１～第３プログラムコードの何れかを読み出すように構成してもよい。つまり、回転状態情報が「高回転」を表すものであれば、第１プログラムコードを読み出し、「中回転」を表すものであれば、第２プログラムコードを読み出し、「低回転」を表すものであれば、第３プログラムコードを読み出すように構成してもよい。

【０１９６】

また、第１～第３プログラムコードに代えて、第１～第３プログラムコードを表すテンプレート（以下、それぞれ、第１，第２，第３テンプレートと言う）を、例えばモデル記憶部１７に記憶しておくようにしてもよい。そして、この場合、回転状態情報に基づいて、モデル記憶部１７から、第１～第３テンプレートの何れかを読み出して、その読み出した第１～第３テンプレートの何れかに基づき、プログラムコードを生成するように構成すればよい。つまり、回転状態情報が「高回転」を表すものであれば、第１テンプレートを読み出すと共に、その第１テンプレートに基づいて第１プログラムコードを生成し、「中回転」を表すものであれば、第２テンプレートを読み出すと共に、その第２テンプレートに基づいて第２プログラムコードを生成し、「低回転」を表すものであれば、第３テンプレートを読み出すと共に、その第３テンプレートに基づいて第３プログラムコードを生成するように構成すればよい。

【０１９７】

尚、本第４実施形態において、コード生成処理部１２が請求項９～１１の生成手段に相当し、シミュレーション処理部１３の回転状態情報を取得する処理が条件取得手段に相当し、コード記憶部１６が請求項１０のコード記憶手段に相当し、モデル記憶部１７が請求項１１のテンプレート記憶手段に相当している。

【０１９８】

以上、本発明の一実施形態について説明したが、本発明は上記実施形態に限定されるものではなく、本発明の技術範囲内で種々の形態をとることができる。

例えば、上記第１，第２実施形態において、コード生成処理部１２がコード記憶部１６からテンプレートを読み出すと、その読み出したテンプレートに対応する制御モデルが表示部４０に表示されるように構成してもよい。

【０１９９】

また、条件判定演算モデル部における判定処理は、２つ以上の判定処理で構成されていてもよい。

また、図７の例において、成立頻度Ｓ＞成立頻度Ｔという条件についての成立頻度Ｕを取得するようにし、その成立頻度Ｕに応じて、プログラムコードが生成されるような構成としてもよい。

【図面の簡単な説明】

【０２００】

【図１】第１実施形態のプログラムコード生成装置３の構成図である。

【図２】制御モデルの一例を表す図面である（例１）。

【図３】第１の例の制御モデルのプログラムコードを表す図面である（例１）。

【図４】コード生成処理部１２が実行する処理の流れを表すフローチャートである。

【図５】プログラムコードのテンプレートを表す図面である（例１）。

【図６】パラメータ設定の構成を表す図面である（例１）。

【図７】制御モデルの一例を表す図面である（例２）。

【図８】制御モデルのプログラムコードを表す図面である（例２）。

【図９】パラメータ設定の構成を表す図面である（例２）。

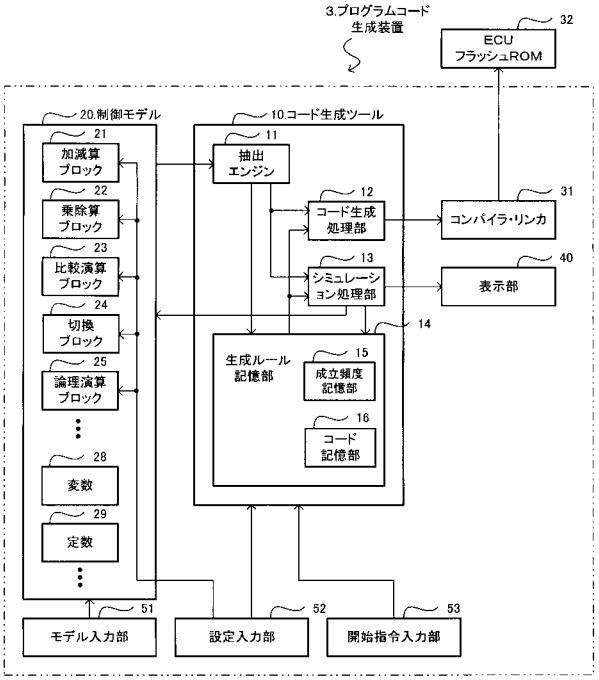
【図１０】プログラムコードのテンプレートを表す図面である（例２）。

【図１１】パラメータ設定の構成を表す図面である（例３）。

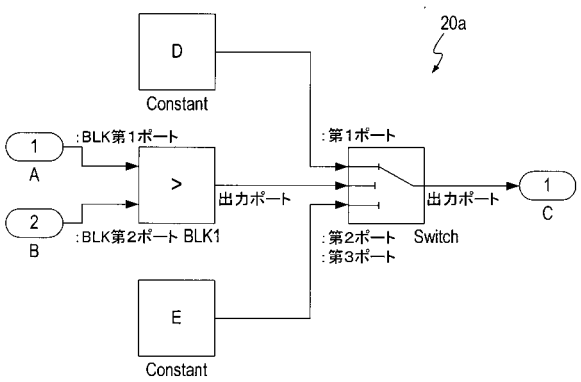
【図１２】シミュレーション処理部１３が実行する処理の流れを表すフローチャートであ

る。
【図１３】第３実施形態のプログラムコード生成装置３の構成図である。
【図１４】制御モデルのテンプレートを表す図面である（例１）。
【図１５】制御モデルのテンプレートを表す図面である（例２）。
【図１６】第３実施形態のコード生成処理部１２が実行する処理の流れを表すフローチャートである。
【図１７】従来のプログラムコード生成方法を説明するための図面である。
【符号の説明】
【０２０１】
３…プログラムコード生成装置、１０…コード生成ツール、１１…抽出エンジン、１２…コード生成処理部、１３…シミュレーション処理部、１４…生成ルール記憶部、１５…成立頻度記憶部、１６…コード記憶部、１７…モデル記憶部、２０，２０ａ，２０ｂ…制御モデル、２１…加減算ブロック、２２…乗除算ブロック、２３…比較演算ブロック、２３ａ，２３ｂ…パラメータ設定領域、２４…切換ブロック、２５…論理演算ブロック、２８…変数ブロック、２９…定数ブロック、３１…コンパイラ・リンカ、４０…表示部、５１…モデル入力部、５２…設定入力部、５３…開始指令入力部、３２…フラッシュＲＯＭ。

【図１】



【図２】



【図３】

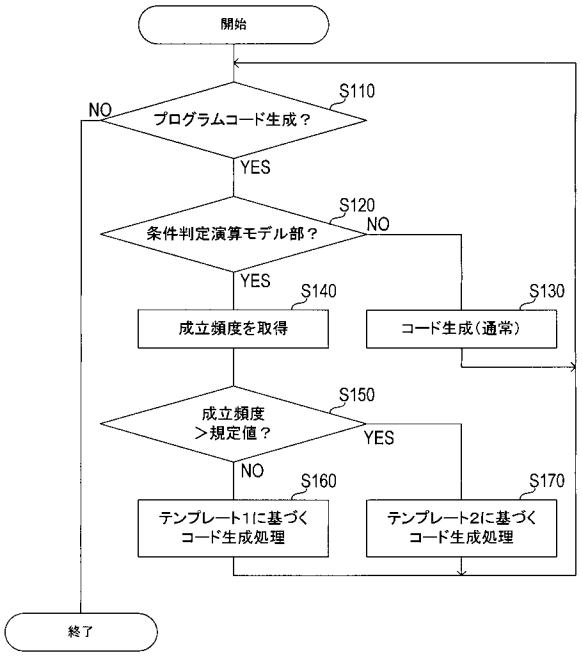
(a)

```
C = E ;
if ( A > B ){
  C = D ;
}
```

(b)

```
C = D ;
if ( A <= B ){
  C = E ;
}
```

【図４】



【図５】

(a)

```
out = in1 ;
if ( in2 > in3 ){
   out = in4 ;
}
```

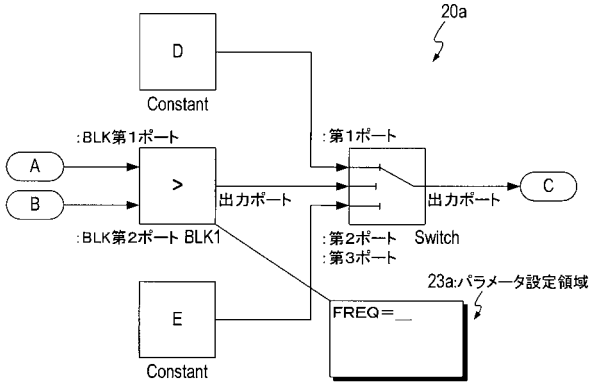
テンプレート1

(b)

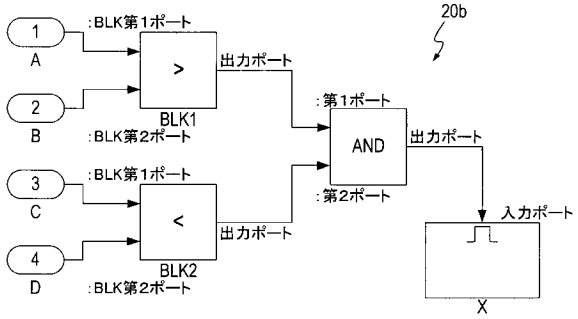
```
out = in4 ;
if ( in2 <= in3 ){
   out = in1 ;
}
```

テンプレート2

【図６】



【図７】



【図８】

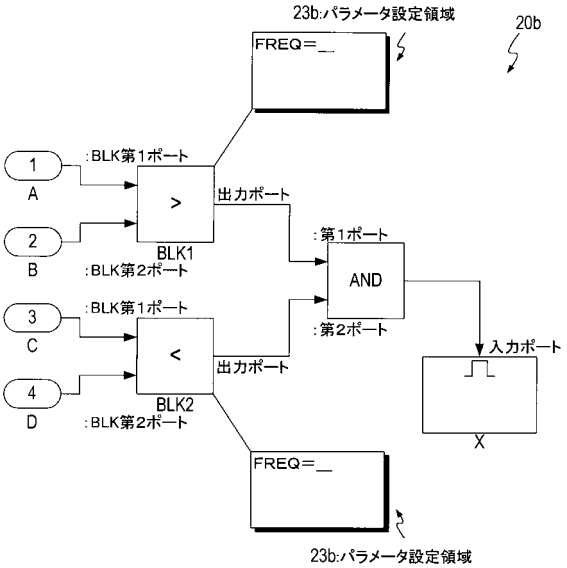
(a)

```
if (( A > B )&&( C < D )) {
    X( );
}
```

(b)

```
if (( C < D )&&( A > B )) {
    X( );
}
```

【図９】



【図１０】

(a)

```
if (( in1 > in2 )&&( in3 < in4 )) {
    X( );
}
```

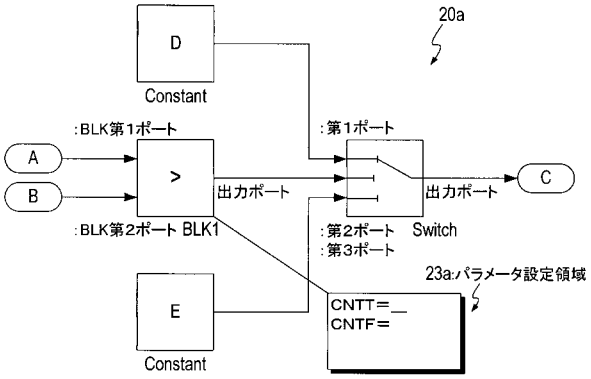
テンプレート1

(b)

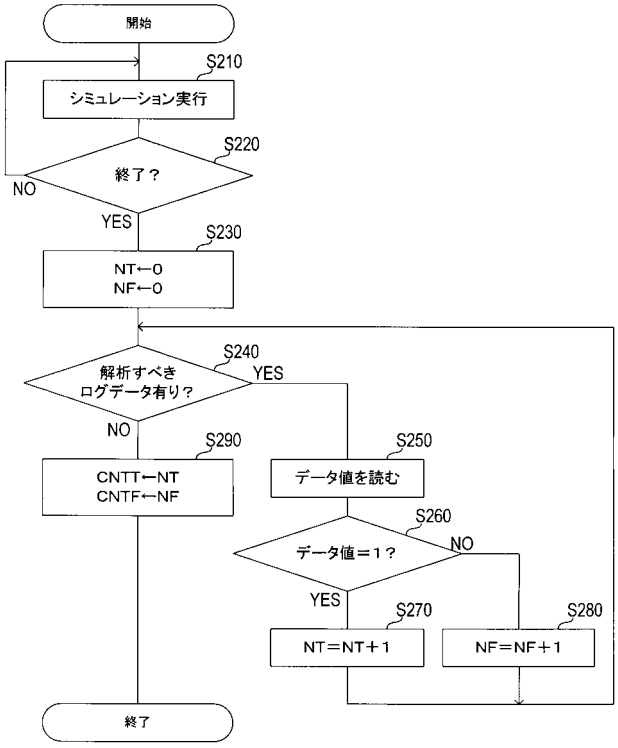
```
if (( in3 < in4 )&&( in1 > in2 )) {
    X( );
}
```

テンプレート2

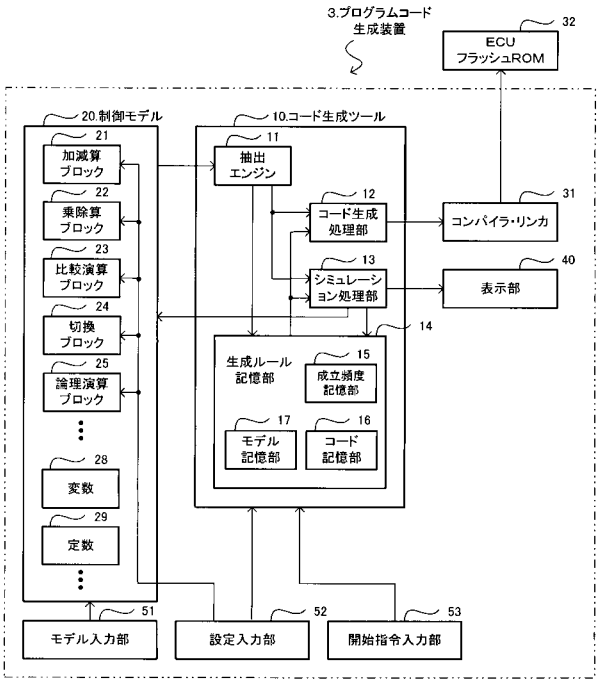
【図１１】



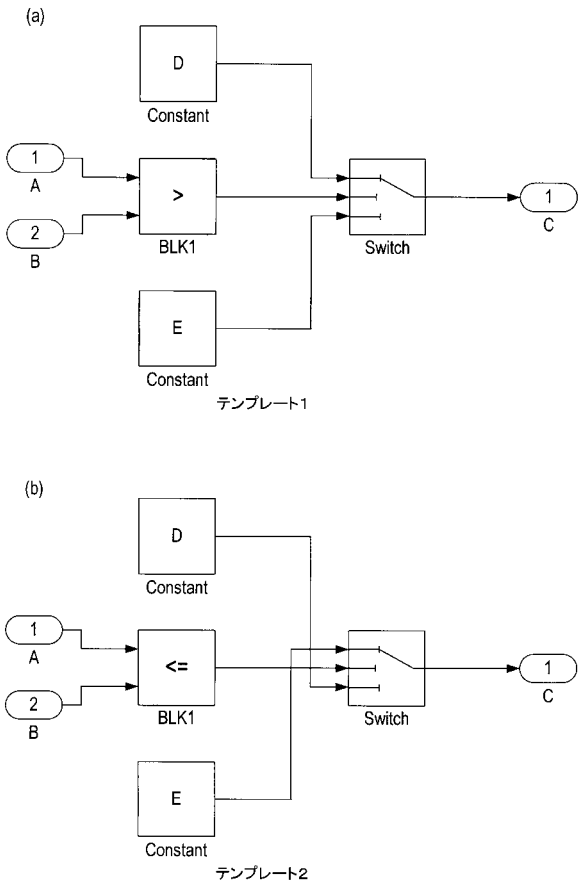
【図１２】



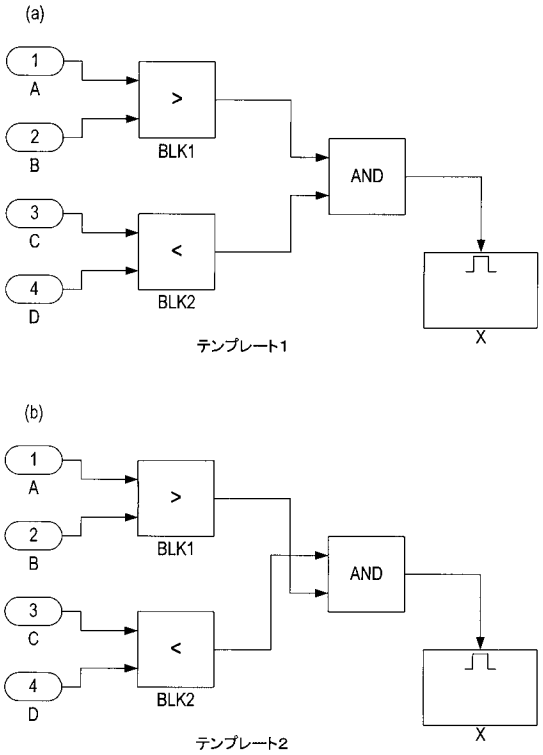
【図１３】



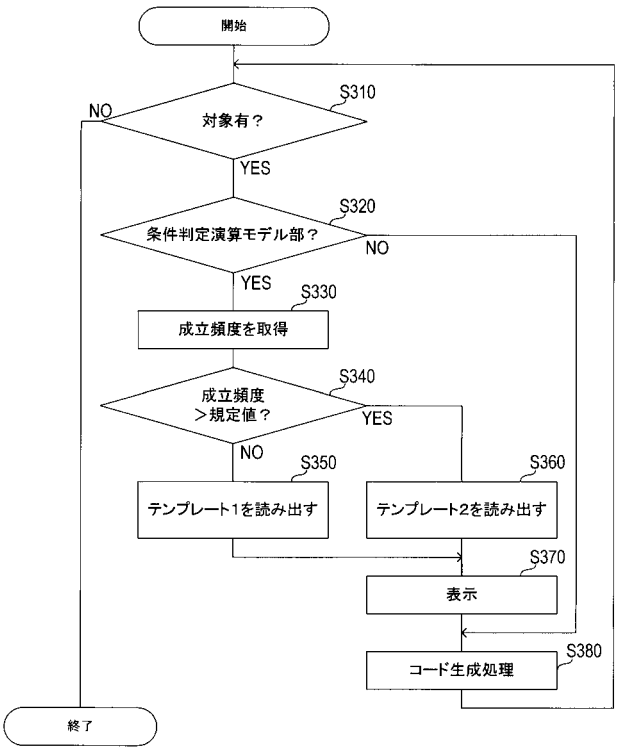
【図１４】



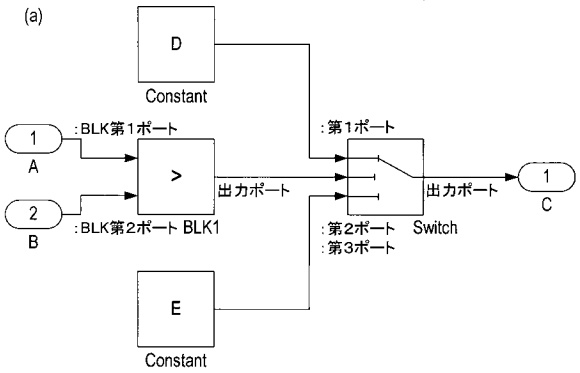
【図１５】



【図１６】



【図１７】



(b)

```
C = E ;
if ( A > B ){
   C = D ;
}
```