(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2020-8941
(P2020-8941A)
(43)公開日　令和2年1月16日(2020.1.16)

(51)Int.Cl.			F I			テーマコード（参考）
G06F	9/455	(2006.01)	G06F	9/455	100	5B081
G05B	19/05	(2006.01)	G05B	19/05	A	5H220

審査請求 未請求　請求項の数 11　ＯＬ　（全 14 頁）

(21)出願番号　特願2018-126934 (P2018-126934)
(22)出願日　平成30年7月3日 (2018.7.3)

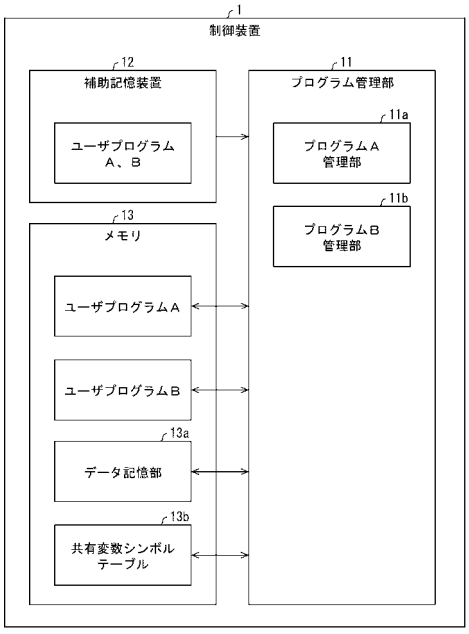
(71)出願人　000002945
オムロン株式会社
京都府京都市下京区塩小路通堀川東入南不動堂町８０１番地
(74)代理人　100155712
弁理士　村上　尚
(72)発明者　荒井　航
京都府京都市下京区塩小路通堀川東入南不動堂町８０１番地　オムロン株式会社内

Ｆターム(参考)　5B081 AA10 CC51
5H220 AA04 BB12 CC07 CX01 DD01 DD04 FF01 FF03 FF05 JJ12 JJ26

(54)【発明の名称】制御装置および制御方法

(57)【要約】
【課題】共有変数を定義するプログラムが複数存在する場合、変数名が衝突し意図せずにデータを上書きすることを防止する。
【解決手段】制御装置（１）は、プログラム管理部（１１）と、データ記憶部（１３ａ）と、共有変数シンボルテーブルと（１３ｂ）を備え、プログラム管理部（１１）は、第１ユーザプログラムの識別子を取得し、前記第１ユーザプログラムにおいて定義されている共有変数の変数名と、前記識別子とを含む共有変数名を生成し、前記共有変数名と、前記共有変数のアドレスとを対応付けて、共有変数シンボルテーブル（１３ｂ）に記録する。
【選択図】図１

図１



【特許請求の範囲】
【請求項１】
第１ユーザプログラムおよび第２ユーザプログラムを実行するプログラム管理部と、
前記第１ユーザプログラムおよび前記第２ユーザプログラムの両方から参照可能な共有変数のデータを記憶するデータ記憶部と、
前記データ記憶部での前記共有変数のアドレスを記憶する共有変数シンボルテーブルとを備え、
前記プログラム管理部は、
前記第１ユーザプログラムの識別子を取得し、
前記第１ユーザプログラムにおいて定義されている前記共有変数の変数名と、前記識別子とを含む共有変数名を生成し、
前記共有変数名と、前記共有変数の前記アドレスとを対応付けて、前記共有変数シンボルテーブルに記録することを特徴とする制御装置。
【請求項２】
前記プログラム管理部は、
前記第２ユーザプログラムから前記共有変数名をキーとして前記共有変数を読み込みまたは書き込みする命令を受けると、前記共有変数シンボルテーブルから前記共有変数名に対応する前記アドレスを取得し、
前記アドレスを用いて前記データ記憶部の前記共有変数を読み込みまたは書き込みすることを特徴とする請求項１に記載の制御装置。
【請求項３】
前記第１ユーザプログラムは、機械語にコンパイルされたものであり、
前記第１ユーザプログラムは、前記共有変数の前記変数名の情報と前記アドレスの情報とを含み、
前記プログラム管理部は、前記第１ユーザプログラムから、前記共有変数の前記変数名と前記アドレスとを取得することを特徴とする請求項１または２に記載の制御装置。
【請求項４】
前記第１ユーザプログラムは、インタプリタ型プログラムであり、
前記第１ユーザプログラムは、前記共有変数の前記変数名の情報を含み、
前記プログラム管理部は、前記第１ユーザプログラムのインタプリタとして機能し、
前記プログラム管理部は、前記第１ユーザプログラムから前記共有変数の前記変数名を取得すると、
前記共有変数名を生成し、
前記共有変数シンボルテーブルに前記共有変数名が記録されている場合、前記共有変数の前記アドレスを取得し、
前記共有変数シンボルテーブルに前記共有変数名が記録されていない場合、前記共有変数に割り当てる前記アドレスを決定することを特徴とする請求項１または２に記載の制御装置。
【請求項５】
前記第１ユーザプログラムは、前記識別子の情報を含み、
前記プログラム管理部は、前記第１ユーザプログラムから、前記識別子を取得することを特徴とする請求項３または４に記載の制御装置。
【請求項６】
前記プログラム管理部は、設定ファイルから、前記識別子を取得することを特徴とする請求項３または４に記載の制御装置。
【請求項７】
前記プログラム管理部は、前記第２ユーザプログラムを実行すると、
非共有変数と前記非共有変数のアドレスとを記録した非共有シンボルテーブルを作成し、
前記第２ユーザプログラムにおいて参照されている変数の変数名が前記非共有シンボ

ルテーブルに記録されている場合、該変数は非共有変数であると判断し、
　前記第２ユーザプログラムにおいて参照されている前記変数の変数名が前記非共有シンボルテーブルに記録されていない場合、該変数は共有変数であると判断することを特徴とする請求項１から６のいずれか一項に記載の制御装置。
【請求項８】
　前記第１ユーザプログラムと、前記第２ユーザプログラムとは、互いに異なるプログラミング言語で構成されたものであることを特徴とする請求項１から７のいずれか一項に記載の制御装置。
【請求項９】
　前記識別子は、前記第１ユーザプログラムのパスを含むことを特徴とする請求項１から４のいずれか一項に記載の制御装置。
【請求項１０】
　前記識別子は、前記第１ユーザプログラムのファイル名を含むことを特徴とする請求項１から４のいずれか一項に記載の制御装置。
【請求項１１】
　第１ユーザプログラムおよび第２ユーザプログラムを実行する方法であって、
　前記第１ユーザプログラムの識別子を取得し、
　前記第２ユーザプログラムから参照可能かつ前記第１ユーザプログラムにおいて定義されている共有変数の変数名と、前記識別子とを含む共有変数名を生成し、
　前記共有変数名と、前記共有変数のアドレスとを対応付けて、共有変数シンボルテーブルに記録し、
　前記共有変数のデータを、データ記憶部における前記アドレスに対応する領域に記憶させることを特徴とする制御方法。
【発明の詳細な説明】
【技術分野】
【０００１】
　本発明は、ユーザプログラム間で共有するデータの意図せずに上書きすることを防止するための制御装置およびこの制御装置による制御方法に関する。
【背景技術】
【０００２】
　従来では、以下のようなコントロール制御装置が知られている（特許文献１を参照）。すなわち、ユーザプログラムが複数あるとき、それらの間のデータのやり取りはデータ記憶部を介して行う。プログラム間でデータのやり取りを行うにあたっては、データに変数名を与え、その変数名をキーにして各ユーザプログラムがデータ記憶部からデータの書き込みや読み取りを行う。
【０００３】
　また、従来では、異なるアプリケーションフレームワーク間のデータサービスのためのプラットフレームが知られている（特許文献２を参照）。
【０００４】
　また、従来では、仕様記述もしくは言語記述を処理する言語処理システムにおける入力／出力仕様記述中もしくは入力ソースプログラム／出力オブジェクトプログラム中の名標を管理する言語管理システムの名標管理方式が知られている（特許文献３を参照）。
【先行技術文献】
【特許文献】
【０００５】
【特許文献１】特開２０００－１３２２０８号公報
【特許文献２】特開２００６－２４４４８８号公報
【特許文献３】特開平４－２６０１３４号公報
【発明の概要】
【発明が解決しようとする課題】

【０００６】
しかしながら、上記の何れの技術も、プログラムが複数存在する場合、変数名が衝突し意図せずにデータを上書きしてしまうことを防止する技術ではない。
【０００７】
本発明の一態様は、共有変数を定義するプログラムが複数存在する場合、変数名が衝突し意図せずにデータを上書きすることを防止することを目的とする。
【課題を解決するための手段】
【０００８】
上記の課題を解決するために、本発明の一態様に係る制御装置は、第１ユーザプログラムおよび第２ユーザプログラムを実行するプログラム管理部と、前記第１ユーザプログラムおよび前記第２ユーザプログラムの両方から参照可能な共有変数のデータを記憶するデータ記憶部と、前記データ記憶部での前記共有変数のアドレスを記憶する共有変数シンボルテーブルとを備え、前記プログラム管理部は、前記第１ユーザプログラムの識別子を取得し、前記第１ユーザプログラムにおいて定義されている前記共有変数の変数名と、前記識別子とを含む共有変数名を生成し、前記共有変数名と、前記共有変数の前記アドレスとを対応付けて、前記共有変数シンボルテーブルに記録することを特徴とする。
【０００９】
上記の態様によれば、共有変数を定義するプログラムが複数存在する場合、変数名が衝突し意図せずにデータを上書きすることを防止することができる。
【００１０】
制御装置の前記プログラム管理部は、前記第２ユーザプログラムから前記共有変数名をキーとして前記共有変数を読み込みまたは書き込みする命令を受けると、前記共有変数シンボルテーブルから前記共有変数名に対応する前記アドレスを取得し、前記アドレスを用いて前記データ記憶部の前記共有変数を読み込みまたは書き込みすることを特徴とする。
【００１１】
上記の態様によれば、前記プログラム管理部は、プログラムＡによって既に生成されていた前記データ記憶部の前記共有変数を読み込みまたは書き込みすることができる。
【００１２】
制御装置では、前記第１ユーザプログラムは、機械語にコンパイルされたものであり、前記第１ユーザプログラムは、前記共有変数の前記変数名の情報と前記アドレスの情報とを含み、前記プログラム管理部は、前記第１ユーザプログラムから、前記共有変数の前記変数名と前記アドレスとを取得することを特徴とする。
【００１３】
上記の態様によれば、前記プログラム管理部は、前記第１ユーザプログラムから、前記共有変数の前記変数名と前記アドレスとを取得することができる。
【００１４】
制御装置では、前記第１ユーザプログラムは、インタプリタ型プログラムであり、前記第１ユーザプログラムは、前記共有変数の前記変数名の情報を含み、前記プログラム管理部は、前記第１ユーザプログラムのインタプリタとして機能し、前記プログラム管理部は、前記第１ユーザプログラムから前記共有変数の前記変数名を取得すると、前記共有変数名を生成し、前記共有変数シンボルテーブルに前記共有変数名が記録されている場合、前記共有変数の前記アドレスを取得し、前記共有変数シンボルテーブルに前記共有変数名が記録されていない場合、前記共有変数に割り当てる前記アドレスを決定することを特徴とする。
【００１５】
上記の態様によれば、前記プログラム管理部は、共有変数の重複を防ぎ、共有変数を適切に管理することができる。
【００１６】
制御装置では、前記第１ユーザプログラムは、前記識別子の情報を含み、前記プログラム管理部は、前記第１ユーザプログラムから、前記識別子を取得することを特徴とする。

【００１７】
制御装置では、前記プログラム管理部は、設定ファイルから、前記識別子を取得することを特徴とする。
【００１８】
制御装置では、前記プログラム管理部は、前記第２ユーザプログラムを実行すると、非共有変数と前記非共有変数のアドレスとを記録した非共有シンボルテーブルを作成し、前記第２ユーザプログラムにおいて参照されている変数の変数名が前記非共有シンボルテーブルに記録されている場合、該変数は非共有変数であると判断し、前記第２ユーザプログラムにおいて参照されている前記変数の変数名が前記非共有シンボルテーブルに記録されていない場合、該変数は共有変数であると判断することを特徴とする。
【００１９】
上記の態様によれば、前記プログラム管理部は、前記第２ユーザプログラムにおいて参照される変数の変数名が前記非共有シンボルテーブルに記録されているか否かに応じて、該変数が非共有変数であるか共有変数であるかを好適に判断することができる。
【００２０】
制御装置では、前記第１ユーザプログラムと、前記第２ユーザプログラムとは、互いに異なるプログラミング言語で構成されたものであることを特徴とする。
【００２１】
上記の態様によれば、互いに異なるプログラミング言語で構成された第１ユーザプログラムと第２ユーザプログラムとの間で、変数を共有することができる。
【００２２】
制御装置では、前記識別子は、前記第１ユーザプログラムのパスを含むことを特徴とする。
【００２３】
制御装置では、前記識別子は、前記第１ユーザプログラムのファイル名を含むことを特徴とする。
【００２４】
上記の課題を解決するために、本発明の一態様に係る制御方法は、第１ユーザプログラムおよび第２ユーザプログラムを実行する方法であって、前記第１ユーザプログラムの識別子を取得し、前記第２ユーザプログラムから参照可能かつ前記第１ユーザプログラムにおいて定義されている共有変数の変数名と、前記識別子とを含む共有変数名を生成し、前記共有変数名と、前記共有変数のアドレスとを対応付けて、共有変数シンボルテーブルに記録し、前記共有変数のデータを、データ記憶部における前記アドレスに対応する領域に記憶させることを特徴とする。
【００２５】
上記の態様によれば、共有変数を定義するプログラムが複数存在する場合、変数名が衝突し意図せずにデータを上書きすることを防止することができる。
【発明の効果】
【００２６】
本発明の一態様によれば、共有変数を定義するプログラムが複数存在する場合、変数名が衝突し意図せずにデータを上書きすることを防止することができる。
【図面の簡単な説明】
【００２７】
【図１】本発明に係る制御装置の構成の一例を模式的に示すブロック図である。
【図２】図１の共有変数シンボルテーブルの詳細を例示する図である。
【図３】プログラム管理部の構成例２のプログラム管理部が実行する工程のフロチャートである。
【図４】プログラムＡの構成例２のプログラム管理部が実行する工程のフロチャートである。
【図５】プログラム管理部の構成例３のプログラム管理部が実行する工程のフロチャート

である。
【図６】本発明の実施形態１に係る制御方法における各工程を示すフロチャートである。
【発明を実施するための形態】
【００２８】
§１　適用例
図１を用いて、本実施形態に係る制御装置の適用場面の一例について説明する。図１は、本実施形態に係る制御装置の構成の一例を模式的に示すブロック図である。本実施形態に係る制御装置１は、共有変数を定義するユーザプログラムが複数存在する場合、変数名が衝突し意図せずにデータを上書きすることを防止するための装置である。
【００２９】
図１に示すように、制御装置１は、プログラム管理部１１と、補助記憶装置１２と、メモリ１３とを備える。
【００３０】
プログラム管理部１１は、プログラムＡ管理部１１ａと、プログラムＢ管理部１１ｂとを含む。本実施形態では、プログラム管理部１１により第１ユーザプログラム（以下プログラムＡと称する）および第２ユーザプログラム（以下プログラムＢと称する）を実行する。
【００３１】
補助記憶装置１２は、プログラムＡおよびプログラムＢを格納する。メモリ１３は、プログラムＡおよびプログラムＢを一時的に記憶する。メモリ１３は、データ記憶部１３ａと、共有変数シンボルテーブル１３ｂとを備える。データ記憶部１３ａは、プログラムＡおよびプログラムＢの両方から参照可能な共有変数のデータを記憶している。共有変数シンボルテーブル１３ｂは、データ記憶部１３ａでの前記共有変数のアドレスを記憶している。
【００３２】
プログラムＡ管理部１１ａは、プログラムＡで定義される共有変数に対して、該共有変数の変数名とプログラムＡの識別子とを含む共有変数名を生成する。プログラムＡ管理部１１ａは、共有変数名とアドレスとを対応付けて共有変数シンボルテーブルで管理する。プログラムＢから該共有変数にアクセス（読み込み／書き込み）する場合、上記共有変数名を用いてアクセスする。
【００３３】
本実施形態に係る制御装置によれば、共有変数を定義するプログラムが複数存在する場合、変数名が衝突し意図せずにデータを上書きすることを防止することができる。
【００３４】
§２　構成例
（制御装置１の概略構成）
図１～図２を用いて、制御装置１の概略構成の一例について説明する。図１は、本実施形態に係る制御装置の構成の一例を模式的に示すブロック図である。図１に示すように、制御装置１は、プログラム管理部１１と、補助記憶装置１２と、メモリ１３とを備える。制御装置１は、たとえばＰＬＣ（プログラマブルロジックコントローラ）等の各種の機器の動作を制御するものが挙げられる。
【００３５】
図１に例示したように、プログラム管理部１１は、プログラムＡ管理部１１ａと、プログラムＢ管理部１１ｂとを含む。本実施形態では、プログラム管理部１１によりプログラムＡおよびプログラムＢを実行する。具体的には、プログラムＡ管理部１１ａによりプログラムＡを実行し、プログラムＢ管理部１１ｂによりプログラムＢを実行する構成を例示するが、これは本実施形態を限定するものではない。
【００３６】
また、プログラムＡおよびプログラムＢに詳細について後述する。
【００３７】

補助記憶装置１２は、プログラムＡおよびプログラムＢを格納するためのものである。通常、プログラムＡおよびプログラムＢは補助記憶装置１２に格納されている。補助記憶装置１２は、例えば、不揮発性のメモリ、またはハードディスク等であるが、これに限らない。補助記憶装置１２を備えることにより、データ消失を確実に防止することができる。

【００３８】

メモリ１３は、プログラムＡおよびプログラムＢを一時的に記憶する。メモリ１３は、例えば揮発性のメモリ等であるが、これに限らない。メモリ１３は、データ記憶部１３ａと、共有変数シンボルテーブル１３ｂとを備える。データ記憶部１３ａは、プログラムＡおよびプログラムＢの両方から参照可能な共有変数のデータを記憶している。共有変数シンボルテーブル１３ｂは、データ記憶部１３ａでの前記共有変数のアドレスを記憶している。

【００３９】

ここでいう共有変数とは、複数のユーザプログラムから参照可能な変数のことを指す。共有変数シンボルテーブル１３ｂは、前記データ記憶部１３ａに記憶されている前記共有変数の記憶場所（アドレス）を記憶している。

【００４０】

（プログラム管理部の構成例１）

図２は、図１の共有変数シンボルテーブル１３ｂの詳細を例示する図である。プログラム管理部の構成例１として、本実施形態では、プログラム管理部１１は、ユーザからプログラムＡを実行する指示を受けると、補助記憶装置１２からプログラムＡをロードし、メモリ１３にプログラムＡを記憶させる。ここでは、例えばプログラムＡはコンパイラ型のプログラムであるとする。補助記憶装置１２に格納されたプログラムＡは、機械語にコンパイルされたプログラムである。コンパイルされたプログラムＡは、共有変数と該共有変数のアドレスとの対応関係を示すシンボルテーブルを含む。非共有変数は、プログラムＡの中だけで使用され、プログラムＡのみが参照可能な変数である。コンパイルされたプログラムＡにおいて、非共有変数のアドレスが規定されている。コンパイルされたプログラムＡにおいて、非共有変数の変数名の情報は残っていなくてもよい。ただし、プログラムＡは、コンパイルされた後でも、共有変数の変数名の情報とアドレスの情報とを含む。

【００４１】

プログラム管理部１１は、共有変数の変数名の情報、アドレスの情報、および、プログラムＡの識別子を取得する。プログラム管理部１１は、プログラムＡにおいて定義されている前記共有変数の変数名と、前記識別子とを含む共有変数名を生成する。プログラム管理部１１は、前記共有変数名と、前記共有変数の前記アドレスとを対応付けて、共有変数シンボルテーブル１３ｂに記録する。また、プログラム管理部１１は、データ記憶部１３ａの前記アドレスに、前記共有変数のデータを書き込む。

【００４２】

換言すると、プログラム管理部１１により生成される共有変数名は、前記共有変数の変数名と、前記識別子とを含む。プログラム管理部１１は、前記共有変数名と、前記共有変数の前記アドレスとを対応付けて、共有変数シンボルテーブル１３ｂに記録する。

【００４３】

ユーザプログラムの識別子は、該ユーザプログラムに固有の文字列または数値である。複数のユーザプログラムの識別子は、互いに重複しないように設定される。例えば、ユーザプログラムのプログラム名、ユーザプログラムの配置先ディレクトリのパス、ユーザが任意に設定する文字列、ＵＵＩＤ（universally unique identifier）、またはドメイン名を、該ユーザプログラムの識別子としてもよい。

【００４４】

図２の例示では、プログラムＡの識別子が「Ｐｒｅｓｓ」である。プログラムＡで定義されている共有変数の変数名は「ｓｔａｔｕｓ」である。変数名「ｓｔａｔｕｓ」は、プログラムＡのソースコードで定義された変数名である。プログラム管理部１１は、識別子

「Ｐｒｅｓｓ」と元々の変数名「ｓｔａｔｕｓ」を組み合わせることで、該共有変数の共有変数名「Ｐｒｅｓｓ.ｓｔａｔｕｓ」を生成する。共有変数名「Ｐｒｅｓｓ.ｓｔａｔｕｓ」は、他のプログラムＢから該共有変数を参照するときにキーとして使用される変数名である。ここでは、識別子と変数名との間に「.」を加えているが、代わりに任意の文字または文字列を用いてもよい。また、識別子と変数名との間の文字はなくてもよい。識別子と変数名との順番も任意である。例えば、図２に例示する「Ｃｕｔｔｅｒ.ｓｔａｔｕｓ」は、プログラムＢで定義される共有変数の、共有変数名である。プログラムＢの識別子が「Ｃｕｔｔｅｒ」である。プログラムＢで定義されている共有変数の変数名は「ｓｔａｔｕｓ」である。プログラム管理部１１は、所定の規則に沿って識別子と変数名とを含む共有変数名を生成する。これにより、複数のユーザプログラムにおいて、変数名「ｓｔａｔｕｓ」が同じ複数の共有変数が定義されていても、共有変数名の衝突を回避することができる。共有変数名はたとえばＰｒｅｓｓ.ｓｔａｔｕｓ、Ｃｕｔｔｅｒ.ｓｔａｔｕｓを挙げ、これらの共有変数名に対応するアドレスはそれぞれ０Ｘ００００００００、０Ｘ００００００１を挙げたが、これは本実施形態を限定するものではない。

【００４５】

上記の態様によれば、共有変数を定義するユーザプログラムが複数存在する場合、変数名が衝突し意図せずにデータを上書きすることを防止することができる。

【００４６】

（プログラム管理部の構成例２）

図３は、プログラム管理部の構成例２のプログラム管理部が実行する工程のフロチャートである。図３に示すように、プログラムＢ管理部１１ｂは、プログラムＢから前記共有変数名をキーとして前記共有変数を読み込みまたは書き込みする命令を受けると、共有変数シンボルテーブル１３ｂから前記共有変数名に対応する前記アドレスを取得する。プログラムＢ管理部１１ｂは、前記アドレスを用いてデータ記憶部１３ａの前記共有変数を読み込みまたは書き込みする。

【００４７】

詳細に、ステップＳ１において、プログラムＢ管理部１１ｂは、前記プログラムＢから前記共有変数名をキーとして前記共有変数を読み込みまたは書き込みする命令を受ける。続いて、ステップＳ２において、プログラムＢ管理部１１ｂは、共有変数シンボルテーブル１３ｂから前記共有変数名に対応する前記アドレスを取得する。最後に、ステップＳ３において、プログラムＢ管理部１１ｂは、前記アドレスを用いて前記データ記憶部１３ａの前記共有変数を読み込みまたは書き込みする。

【００４８】

上記の態様によれば、プログラムＢ管理部１１ｂは、プログラムＡによって既に生成されていたデータ記憶部１３ａの前記共有変数を読み込みまたは書き込みすることができる。

【００４９】

（プログラムＡの構成例１）

上述したプログラムＡの構成例１について詳細に説明する。プログラムＡは、たとえば機械語にコンパイルされたものである。プログラムＡは、機械語にコンパイルされた後でも、前記共有変数の前記変数名の情報と前記アドレスの情報とを含む。プログラムＡ管理部１１ａは、前記プログラムＡから、前記共有変数の前記変数名と前記アドレスとを取得する。プログラムＡを実行するプログラムＡ管理部１１ａは、取得された前記アドレスを用いて、データ記憶部１３ａの前記共有変数を読み込みまたは書き込みすることができる。

【００５０】

ここで、プログラムＡはラダー言語またはＣ言語等を用いたプログラムが挙げられるが、これは本実施形態を限定するものではない。ラダー言語は、制御装置１が工作機械または生産装置等の動作制御を行う場合に適したプログラミング言語である。

【００５１】

具体的には、プログラムＡ管理部１１ａは、プログラムＡからシンボルテーブルとプログラムＡの識別子を取得する。プログラムＡ管理部１１ａは、共有変数の変数名に前記識別子を付与して共有変数名を生成する。そして、プログラムＡ管理部１１ａは、共有変数シンボルテーブル１３ｂに共有変数名とアドレスとを対応付けて記録する。続いて、プログラムＡ管理部１１ａは、プログラムＡを実行する。そして、プログラムＡ管理部１１ａは、前記プログラムＡから前記アドレスをキーにしてデータ書き込み指示を受ける。最後に、プログラムＡ管理部１１ａは、メモリ１３に前記アドレスをキーにしてデータ書き込みする。ここで、共有変数名の生成は前記プログラムＡのロード時に行う。

【００５２】

上記の態様によれば、プログラムＡ管理部１１ａは、プログラムＡから、前記共有変数の前記変数名と前記アドレスとを取得することができる。

【００５３】

例えば、生産装置をラダー言語であるプログラムＡで制御し、Ｊａｖａ言語であるプログラムＢでデータ処理を行う場合について説明する。プログラムＡは、生産装置またはセンサから、生産ラインに関するデータ（センシング情報等）を得る。プログラムＡは、得られたデータを、共有変数としてデータ記憶部１３ａに記録させる。プログラムＢは、プログラムＡが記録した共有変数を読み出し、例えばセンシング情報等を用いて統計処理を行う。ラダー言語は制御に適したプログラミング言語であり、Ｊａｖａ等の汎用プログラミング言語は、情報処理に適したプログラミング言語である。このように、異なるプログラミング言語を用いて、共通のデータに対してそれぞれ適した処理を行うことができる。

【００５４】

また、既存のプログラムＡを実行し続けたまま（生産装置を稼働させ続けたまま）、既存のプログラム管理部およびプログラムＢを本実施形態のプログラム管理部１１およびプログラムＢに入れ替えてもよい。これにより、既存の制御装置を共有変数名の衝突を回避できる本実施形態の制御装置にすることができる。

【００５５】

（プログラムＡの構成例２）

図４は、プログラムＡの構成例２のプログラムＡ管理部１１ａが実行する工程のフロチャートである。例えば、プログラムＡは、たとえばインタプリタ型プログラムである。プログラムＡは、前記共有変数の前記変数名の情報を含む。プログラムＡ管理部１１ａは、前記プログラムＡのインタプリタとして機能する。

【００５６】

図４におけるステップＳ４において、プログラムＡ管理部１１ａは、前記プログラムＡから前記共有変数の前記変数名と識別子とを取得すると、前記共有変数名を生成する。続いて、ステップＳ５において、プログラムＡ管理部１１ａは、共有変数名が記録されているか否かを判定する。プログラムＡ管理部１１ａは、共有変数シンボルテーブル１３ｂに前記共有変数名が既に記録されている場合（Ｓ５でＹＥＳ）、前記共有変数の前記アドレスを取得する（ステップＳ６）。一方、プログラムＡ管理部１１ａは、共有変数シンボルテーブル１３ｂに前記共有変数名が記録されていない場合（Ｓ５でＮＯ）、前記共有変数に割り当てる前記アドレスを決定する（ステップＳ７）。プログラムＡ管理部１１ａは、決定した前記アドレスと前記共有変数名とを対応付けて共有変数シンボルテーブル１３ｂに記録する。

【００５７】

また、インタプリタ型のプログラムとしては、例えば、Ｊａｖａ（登録商標）、ＪａｖａＳｃｒｉｐｔ（登録商標）、Ｐｙｔｈｏｎなどのプログラミング言語を用いたプログラムが挙げられる。例えば、ユーザプログラムがＪａｖａプログラムである場合、補助記憶装置１２に格納されたユーザプログラムは中間コードにコンパイルされたものであり、プログラムＡ管理部１１ａは、中間コードであるユーザプログラムを解釈して実行するＪａｖａ仮想マシンとして機能する。

【００５８】

具体的には、プログラムＡ管理部１１ａは、プログラムＡから前記プログラムＡの識別子を取得する。また、プログラムＡ管理部１１ａは、前記プログラムＡを実行する。そして、プログラムＡ管理部１１ａは、変数名をキーにしてデータ書き込み指示を受ける。続いて、プログラムＡ管理部１１ａは、共有変数の変数名に前記プログラムＡの前記識別子を付加して共有変数名を生成する。そして、プログラムＡ管理部１１ａは、共有変数シンボルテーブル１３ｂを参照して、該共有変数が定義済みであるか否かを判定する。該共有変数が未定義である場合、アドレスを決定し、共有変数に前記アドレスを割り当てる。プログラムＡ管理部１１ａは、決定した前記アドレスと前記共有変数名とを対応付けて共有変数シンボルテーブル１３ｂに記録する。一方、該共有変数が定義済みである場合、割り当て済みの前記アドレスを共有変数シンボルテーブル１３ｂから取得する。最後に、プログラムＡ管理部１１ａは、データ記憶部１３ａに前記アドレスをキーにしてデータ書き込みする。ここで、プログラムＡ管理部１１ａは、共有変数であるか否かの判断を実行時に行う。プログラムＡにおいて使用される変数が共有変数であるか非共有変数であるかは、プログラムＡにおいて定義されている。例えば、プログラムＡにおいて、変数に対して共有または非共有を指定することにより、変数が共有変数であるか非共有変数であるかを定義する。また、プログラムＡ管理部１１ａは、プログラムＡの実行時に共有変数名を生成する。

【００５９】

上記の態様によれば、プログラム管理部１１は、共有変数の重複を防ぎ、共有変数を適切に管理することができる。

【００６０】

例えば、プログラムＡのソースコードにおいて、共有変数「ｓｔａｔｕｓ」への書き込みは、一般的な方法で「status="OK"」と記述されてもよい。これは共有変数「status」に値「OK」を書き込むことを意味する。ここで「＝」は値の代入を意味する演算子である。または、プログラムＡ管理部１１ａは、共有変数に書き込むためのＡＰＩ関数（application program interface）を提供してもよい。例えば、プログラムＡのソースコードにおいて、共有変数「ｓｔａｔｕｓ」への書き込みは、ＡＰＩ関数「write」を用いて「write("status","OK")」と記述されてもよい。いずれも、プログラムＡは、キーとして共有変数の変数名「ｓｔａｔｕｓ」をプログラムＡ管理部１１ａに渡す。

【００６１】

（識別子の取得例１）

上述したプログラムＡは、前記識別子の情報を含み、プログラムＡ管理部１１ａは、前記プログラムＡから、前記識別子を取得する。

【００６２】

（識別子の取得例２）

例えば、プログラムＡ管理部１１ａは、設定ファイルから、前記識別子を取得してもよい。プログラムＡの識別子を規定する設定ファイルを、所定のディレクトリに格納していてもよい。

【００６３】

（プログラム管理部の構成例３）

図５は、プログラム管理部の構成例３のプログラム管理部が実行する工程のフロチャートである。ここでは、プログラムＢはインタプリタ型のプログラムである。プログラムＡは先に実行され、プログラムＡで定義された共有変数の共有変数名およびアドレスは、共有変数シンボルテーブル１３ｂに記録されている。図５に示すように、ステップＳ８において、プログラムＢ管理部１１ｂは、プログラムＢを実行する。続いて、ステップＳ９において、プログラムＢ管理部１１ｂは、プログラムＢにおいて定義されている非共有変数と前記非共有変数のアドレスとを記録した非共有シンボルテーブルを作成する。非共有シンボルテーブルは、実行されるユーザプログラム毎に分けて作成される。例えば、プログラムＢにおいて変数が非共有変数として定義されたとき、該変数名および割り当てたアドレスを非共有シンボルテーブルに記録する。プログラムＢ管理部１１ｂは、非共有シンボ

ルテーブルを記憶している。続いて、ステップＳ１０において、プログラムＢ管理部１１ｂは、変数名が記録されているか否かを判定する。プログラムＢ管理部１１ｂは、プログラムＢにおいて参照（読み込み）される変数の変数名が前記非共有シンボルテーブルに記録されている場合（Ｓ１０でＹＥＳ）、該変数は非共有変数であると判断する（ステップＳ１１）。一方、プログラムＢ管理部１１ｂは、プログラムＢにおいて参照（読み込み）される前記変数の変数名が前記非共有シンボルテーブルに記録されていない場合（Ｓ１０でＮＯ）、該変数は共有変数であると判断する（ステップＳ１２）。

【００６４】

上記の態様によれば、プログラムＢ管理部１１ｂは、プログラムＢにおいて参照される変数の変数名が前記非共有シンボルテーブルに記録されているか否かに応じて、該変数が非共有変数であるか共有変数であるかを好適に判断することができる。例えば、プログラムＡで定義された共有変数の共有変数名「Ｐｒｅｓｓ．ｓｔａｔｕｓ」と、プログラムＢで参照された非共有変数の変数名とが偶然一致している場合でも、プログラムＢの非共有変数を誤って前記共有変数だと判断することを防ぐことができる。

【００６５】

（プログラム管理部の構成例４）

プログラム管理部１１の構成例４について、詳細に説明する。ここでは、プログラムＢはコンパイラ型のプログラムである。プログラムＡは先に実行され、プログラムＡで定義された共有変数の共有変数名およびアドレスは、共有変数シンボルテーブル１３ｂに記録されている。プログラムＢを実行するプログラムＢ管理部１１ｂは、共有変数を読み込むためのＡＰＩ関数を提供する。例えば、プログラムＢのソースコードにおいて、共有変数「Ｐｒｅｓｓ．ｓｔａｔｕｓ」の参照（読み込み）は、ＡＰＩ関数「read」を用いて「read("Press.status")」と記述されてもよい。プログラムＢは、キーとして共有変数の共有変数名「Ｐｒｅｓｓ．ｓｔａｔｕｓ」をＡＰＩ関数を介してプログラムＢ管理部１１ｂに渡す。プログラムＢ管理部１１ｂは、共有変数シンボルテーブルから対応するアドレスを取得する。プログラムＢ管理部１１ｂは、アドレスをキーにしてデータ記憶部１３ａから共有変数のデータを取得する。プログラムＢ管理部１１ｂは、共有変数のデータをプログラムＢに返す。

【００６６】

通常、コンパイラ型のユーザプログラムは、機械語に変換されている。そのため、ソースファイルで規定した変数名は、アドレスに変換されており、残っていない。上記の態様によれば、共有変数を参照するＡＰＩを用いることで、コンパイルされたプログラムＢの中の共有変数名はキーとして残る。

【００６７】

制御装置１では、前記プログラムＡと、前記プログラムＢとは、互いに異なるプログラミング言語で構成されたものであってもよい。上記の態様によれば、互いに異なるプログラミング言語で構成されたプログラムＡとプログラムＢとの間で、変数を共有することができる。

【００６８】

例えば、プログラムＡと、プログラムＢとは、同じプログラミング言語で構成されたものであってもよい。

【００６９】

（識別子の構成例１）

前記識別子は、前記プログラムＡのパスを含んでもよい。例えば、ユーザは、プログラムＡを作成する際に、プログラムＡを配置する予定であるディレクトリのパスをプログラムＡ内で規定しておく。または、プログラムＡ管理部１１ａによって実行されたプログラムＡ自身が、プログラムＡのパスを取得してもよい。

【００７０】

（識別子の構成例２）

前記識別子は、前記プログラムＡのファイル名を含んでもよい。パスと同様に、ファイ

ル名はプログラムＡ内で規定されていてもよいし、プログラムＡ管理部１１ａによって実行されたプログラムＡ自身が、プログラムＡのファイル名を取得してもよい。

【００７１】

前記識別子は、前記プログラムＡのパスおよびファイル名を含んでもよい。上述したように、前記識別子は、ユーザが任意に設定する文字列、ＵＵＩＤ（universally unique identifier）、および／またはドメイン名を含んでもよい。ここでのドメイン名とは、ユーザプログラムの提供者（提供会社）が所有するドメイン名（例えば、omron.co.jpまたはjp.co.omron）である。

【００７２】

（制御方法）

図６は、本実施形態に係る制御方法における各工程を示すフロチャートである。図６に示すように、ステップＳ１３において、プログラムＡ管理部１１ａは、前記プログラムＡの識別子を取得する。続いて、ステップＳ１４において、プログラムＡ管理部１１ａは、前記プログラムＢから参照可能かつ前記プログラムＡにおいて定義されている共有変数の変数名と、前記識別子とを含む共有変数名を生成する。続いて、ステップＳ１５において、プログラムＡ管理部１１ａは、前記共有変数名と、前記共有変数のアドレスとを対応付けて、共有変数シンボルテーブル１３ｂに記録する。続いて、ステップＳ１６において、プログラムＡ管理部１１ａは、前記共有変数のデータを、データ記憶部１３ａにおける前記アドレスに対応する領域に記憶させる。

【００７３】

上記の態様によれば、共有変数を定義するプログラムが複数存在する場合、変数名が衝突し意図せずにデータを上書きすることを防止することができる。

【００７４】

〔ソフトウェアによる実現例〕

制御装置１の制御ブロック（特にプログラム管理部１１）は、集積回路（ＩＣチップ）等に形成された論理回路（ハードウェア）によって実現してもよいし、ソフトウェアによって実現してもよい。

【００７５】

後者の場合、制御装置１は、各機能を実現するソフトウェアであるプログラムの命令を実行するコンピュータを備えている。このコンピュータは、例えば１つ以上のプロセッサを備えていると共に、上記プログラムを記憶したコンピュータ読み取り可能な記録媒体を備えている。そして、上記コンピュータにおいて、上記プロセッサが上記プログラムを上記記録媒体から読み取って実行することにより、本発明の目的が達成される。上記プロセッサとしては、例えばＣＰＵ（Central Processing Unit）を用いることができる。上記記録媒体としては、「一時的でない有形の媒体」、例えば、ＲＯＭ（Read Only Memory）等の他、テープ、ディスク、カード、半導体メモリ、プログラマブルな論理回路などを用いることができる。また、上記プログラムを展開するＲＡＭ（Random Access Memory）などをさらに備えていてもよい。また、上記プログラムは、該プログラムを伝送可能な任意の伝送媒体（通信ネットワークや放送波等）を介して上記コンピュータに供給されてもよい。なお、本発明の一態様は、上記プログラムが電子的な伝送によって具現化された、搬送波に埋め込まれたデータ信号の形態でも実現され得る。

【００７６】

本発明は上述した各実施形態に限定されるものではなく、請求項に示した範囲で種々の変更が可能であり、異なる実施形態にそれぞれ開示された技術的手段を適宜組み合わせて得られる実施形態についても本発明の技術的範囲に含まれる。

【符号の説明】

【００７７】

１　制御装置

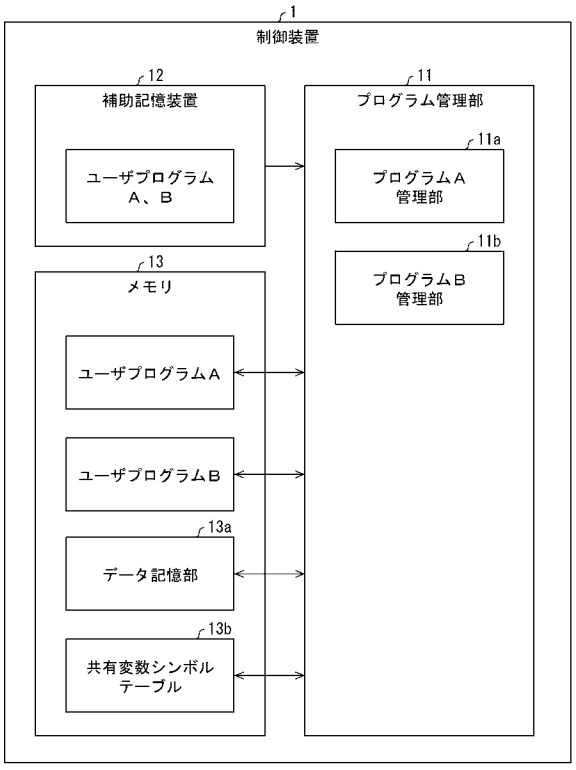
１１　プログラム管理部

１１ａ　プログラムＡ管理部

１１ｂ　プログラムＢ管理部
１２　補助記憶装置
１３　メモリ
１３ａ　データ記憶部
１３ｂ　共有変数シンボルテーブル

【図１】

図１



【図２】

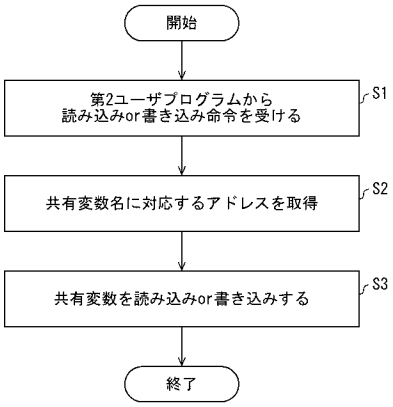
図２

13b
共有変数シンボルテーブル

共有変数名	アドレス
Press.status	0X00000000
Cutter.status	0X00000001
・	・
・	・
・	・
・	・
・	・
・	・
・	・
・	・
・	・

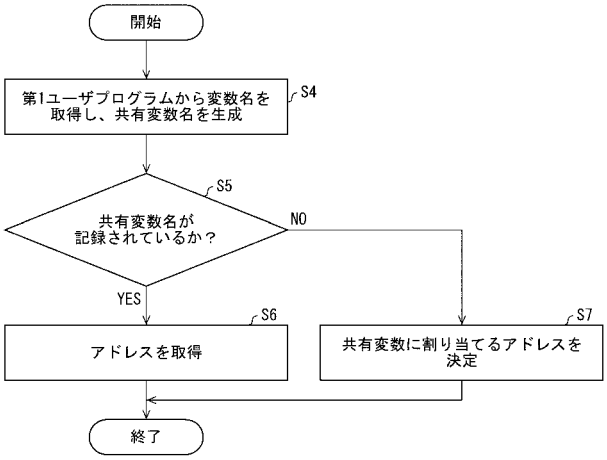
【図３】

図３



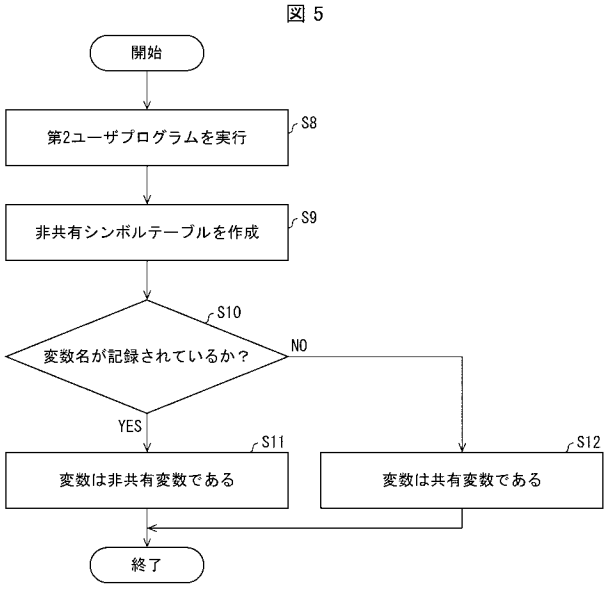
【図４】

図４



【図５】

図５



【図６】

図６