

明 細 書

発明の名称：

モーション制御プログラム、モーション制御方法及びモーション制御装置 技術分野

[0001] 本発明は、モーション制御プログラム、モーション制御方法及びモーション制御装置に関する。

背景技術

[0002] ロボット及びFA（ファクトリーオートメーション）等の分野では、ベルトコンベア的位置やアームの位置などを、意図した通りに動作させることが求められる。このような動作をさせるためには、サーボモータやステッピングモータ等の複数の制御対象装置を、高精度に同期させながら制御する必要がある。このように、複数の制御対象装置を高精度に同期させながら制御する装置は、モーションコントローラやモーション制御装置等と呼ばれる。例えば特許文献1には、安価で簡単な低速通信を利用しつつ滑らかな制御を実現可能なモーション制御用指令システムが開示されている。

先行技術文献

特許文献

[0003] 特許文献1：特開2010-170435号公報

発明の概要

発明が解決しようとする課題

[0004] 一般的に、モーションコントローラは、高精度なリアルタイム性能を要求されることから、モーション制御に特化した専用のハードウェアとして提供される。専用のハードウェアとして提供されることで、必要なリアルタイム性能を確保することができる反面、価格が高く、制御対象装置やその通信プロトコルに依存した構成となっており、汎用性に欠けるというデメリットがある。

[0005] 一方、汎用的な情報処理装置であるパーソナルコンピュータは、専用のハ

ードウェアであるモーションコントローラと比較して安価であるというメリットを有している。また、ユーザが使い慣れたデザインのユーザインタフェースを提供することが可能というメリットもある。

[0006] ここで、現在、パーソナルコンピュータのOS (Operating System) として広く利用されている非リアルタイムOS (例えばWindows (登録商標)) をインストールしたパーソナルコンピュータ上にインストール可能なリアルタイムOSが提供されている。そのため、非リアルタイムOS及びリアルタイムOSの両方をインストールしたパーソナルコンピュータをモーションコントローラとして利用することができれば、様々なメリットを享受できると考えられる。また、1つのパーソナルコンピュータ上に、通信プロトコルが異なる様々な制御対象機器を接続して制御することができれば、更にコスト軽減が図られると考えられる。

[0007] しかしながら、現在、非リアルタイムOS及びリアルタイムOSの両方をインストールした1つのパーソナルコンピュータに接続された、通信プロトコルが異なる複数の制御対象機器を制御することが可能な技術は知られていない。

[0008] そこで、本発明は、非リアルタイムOS及びリアルタイムOSがインストールされたコンピュータに、通信プロトコルが異なる複数の制御対象機器を接続して制御することが可能な技術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

[0009] 本発明の一態様に係るモーション制御プログラムは、非リアルタイムOS及びリアルタイムOSがインストールされ、複数の制御対象装置をモーション制御するためのコンピュータで実行されるモーション制御プログラムであって、コンピュータを、非リアルタイムOS上で動作する受付部と、リアルタイムOS上で動作する制御部と、リアルタイムOS上で動作する複数の通信モジュール部と、として機能させ、受付部は、複数の制御対象装置を制御するユーザ作成プログラムから、複数の制御対象装置の各々が複数のモーション制御サイクルに渡って行うべき動作を示す制御指令を受け付け、受け付

けた制御指令の内容を示す制御指令情報を制御部に通知し、制御部は、受付部から通知された制御指令情報に基づいて、複数の制御対象装置の各々に対してモーション制御サイクルごとに実行させるべき動作を示す、制御対象装置ごとの補間指令をモーション制御サイクルごとに繰り返し生成し、生成した補間指令を、制御対象装置ごとに確保されるメモリ領域に格納し、通信モジュール部は、制御対象装置ごとに確保されたメモリ領域から補間指令を取得し、取得した補間指令を、制御部が認識可能な所定の信号フォーマットから、複数の制御対象装置の各々が認識可能な通信インターフェース規格の信号フォーマットに変換して送信する。

発明の効果

- [0010] 本発明によれば、非リアルタイムOS及びリアルタイムOSがインストールされたコンピュータに、通信プロトコルが異なる複数の制御対象機器を接続して制御することが可能な技術を提供することができる。

図面の簡単な説明

- [0011] [図1]実施形態に係るモーション制御システムのシステム構成例を示す図である。
- [図2]モーション制御装置の概要を説明するための図である。
- [図3]APIチャンネル及び状態チャンネルの一例を説明するための図である。
- [図4]モーション制御装置のハードウェア構成例を示す図である。
- [図5]モーション制御装置の機能構成例を示す図である。
- [図6]デバイス作成処理の一例を説明するための図である。
- [図7]デバイス作成処理が行われる際に生成される各種管理情報の一例を説明するための図である。
- [図8]デバイス作成処理の一例を説明するための図である。
- [図9]デバイス作成処理が行われる際に生成される各種管理情報の一例を説明するための図である。
- [図10]モーション制御装置が行うモーション制御処理の一例を示す図である。
- 。

[図11] モーション制御装置が行うモーション制御処理の一例を示す図である。

[図12] 制御指令モジュールが行う処理手順の一例を示すフローチャートである。

[図13] 制御指示部がクラッシュした場合の処理手順の一例を示す図である。

[図14] マッピング情報の一例を示す図である。

[図15] プロトコル変換処理の処理手順の一例を説明するための図である。

発明を実施するための形態

[0012] 添付図面を参照して、本発明の好適な実施形態について説明する。なお、各図において、同一の符号を付したものは、同一又は同様の構成を有する。

[0013] <システム構成>

図1は、実施形態に係るモーション制御システムのシステム構成例を示す図である。モーション制御システム1は、モーション制御装置10と、複数の制御対象装置20とを含む。なお、モーション制御システム1には、必ずしも複数の制御対象装置20が含まれている必要はなく、1つの制御対象装置20のみが含まれる構成としてもよい。

[0014] 制御対象装置20は、具体的には、サーボモータ（サーボドライバを含む）及びステッピングモータ等のモータである。モーション制御装置10は、複数の制御対象装置20を同期させながら制御することで、ベルトコンベアの回転制御、多軸ロボットの軸制御、回転テーブルの位置決め制御などを行う。1つの制御対象装置20は1つの「軸」を制御する。例えば、6軸ロボットとは、モータである制御対象装置20を6個備えたロボットを意味する。

[0015] 制御対象装置20に制御指令を与えるための通信インターフェース規格として、例えば、E t h e r C A T（登録商標）、R T E X（登録商標）（Realtime Express（登録商標））及びM E C H A T R O L I N K（登録商標）等が知られている。モーション制御装置10には、異なる通信インターフェース規格の制御対象装置20を混在させて接続することが可能である。

[0016] モーション制御装置 10 は、非リアルタイム OS 及びリアルタイム OS がインストールされた汎用的な情報処理装置である。非リアルタイム OS の具体例として、例えば、Windows（登録商標）、macOS（登録商標）等が挙げられる。また、リアルタイム OS は、例えば、RTX（Real Time Extension）、RTH（Real Time Hypervisor）等が挙げられる。また、汎用的な情報処理装置の具体例としては、例えば、PC（パーソナルコンピュータ）、ノート型 PC、サーバ等が挙げられる。

[0017] ここで、サーボモータやステッピングモータの制御に用いられる従来のモーションコントローラは、高精度なリアルタイム性能を確保するために、モーション制御に特化した専用のハードウェアとして提供されることが一般的である。また、モーション制御の内容をユーザがプログラミングするための GUI（Graphical User Interface）を提供するプログラムについては、モーションコントローラに接続されたパーソナルコンピュータにより提供されることが一般的である。

[0018] 一方で、本実施形態に係るモーション制御装置 10 は、モーション制御の内容をユーザがプログラミングするための GUI を非リアルタイム OS 上で提供し、プログラミングされた内容に基づいて制御対象装置 20 を実際に制御する処理をリアルタイム OS 上で行う。これにより、専用のハードウェアによるモーションコントローラを利用した場合と同様の高精度なモーション制御を汎用的な情報処理装置のみで実現する。

[0019] <モーション制御装置の概要>

図 2 は、モーション制御装置 10 の概要を説明するための図である。制御指示部 110 は、非リアルタイム OS 100 上で動作し、制御対象装置 20 が複数のモーション制御サイクルに渡って行うべき動作を示す制御指令を 1 又は複数発行することで、制御対象装置 20 を制御する。制御指示部 110 は、モーション制御システム 1 を用いるユーザが作成するプログラムを含んでおり、当該プログラムは、制御対象装置 20 に対して発行する制御指令、制御指令を発行する順序及びタイミング等が、所定のプログラミング言語を

用いて記述されている。ユーザが作成するプログラムの具体例としては、例えば、F Aを実現するためのベルトコンベア及びアーム等の制御を行うプログラム、ロボット操作を行うプログラム等が挙げられる。非リアルタイムO S 1 0 0では、複数の制御指示部1 1 0を動作させることが可能である。

[0020] 制御指令は、制御指令の識別子（I D等）と指令値とを含む。制御指令の具体例としては、例えば、P T P（Point to Point）動作（指定した地点に移動させる動作）、J O G動作（指定した速度でモータを動作）、直線補間によるP T P動作、円弧補間によるP T P動作、加速時間を指定するJ O G動作、行動時間を指定するJ O G動作等が挙げられる。制御指令の識別子は、制御指令の内容を具体的に特定する情報（例えば円弧補間によるP T P動作であることを示す識別子等）である。指令値は、動作目標を具体的に示すパラメータであり、例えば円弧補間によるP T P動作の場合、移動先の地点を示す値、移動速度、回転半径等である。制御指示部1 1 0で発行された制御指令は、後述するチャンネルを用いて連携部3 1 0に通知される。

[0021] 制御指示部1 1 0での制御指令の発行は、ユーザが作成するプログラムにおいてA P I（Application Programming Interface）関数を呼び出すことで行われる。また、指令値の指定は、A P I関数の引数（argument）に値を設定することで行われる。A P I関数は、制御指令ごとに1つ用意されており、例えばP T P動作に対しては「startpos()」というA P I関数が用意されている。本実施形態では、制御指示部1 1 0で呼び出されたA P I関数及び当該A P I関数に設定された引数を示す情報を、「A P I情報」と言う。なお、本実施形態では、A P I情報を、「制御指令情報」と呼んでもよい。

[0022] 連携部3 1 0は、制御指示部1 1 0から通知されたA P I情報を一時的に蓄積するためのF I F Oキュー（以下、「A P Iバッファ」と言う。）を備えている。連携部3 1 0は、制御指示部1 1 0から受け取った複数のA P I情報を一旦A P Iバッファ3 4 0に蓄積する。また、連携部3 1 0は、A P Iバッファ3 4 0からA P I情報を1つずつ取り出して定周期処理部3 6 0に渡す。つまり、連携部3 1 0は、キューを利用して定周期処理部3 6 0と

の間でAPI情報の受渡を行う。

[0023] 定周期処理部360は、リアルタイムOS300上で動作し、連携部310から通知された制御指令に基づいて、モーション制御サイクルごとに制御対象装置20に実行させる動作を示す補間指令を算出する。また、定周期処理部360は、算出した補間指令をモーション制御サイクルに従って制御対象装置20に送信することで、制御対象装置20をモーション制御させる。1つのモーション制御サイクルは、例えば0.5msや1msといった単位である。制御対象装置20は、モーション制御サイクルごとに通知される補間指令に従って動作することで、最終的に、制御指令により指示される動作目標を達成する。

[0024] また、定周期処理部360は、定周期処理部360が認識可能な信号フォーマット（以下、「共通信号フォーマット」と言う。）と、制御対象装置20が認識可能な通信インターフェース規格に対応する信号フォーマットとの間でプロトコル変換を行う。なお、共通信号フォーマットは、「所定の信号フォーマット」の一例である。

[0025] 制御指示部110及び連携部310は、制御指示部110及び連携部310が共通に書き込み及び読み出し可能なメモリ領域（以下、「チャンネル」と言う。）を介して各種情報の受け渡しを行う。チャンネルには、操作チャンネル、APIチャンネル及び状態チャンネルが存在する。

[0026] 「操作チャンネル」は、制御指示部110及び連携部310が制御情報を送受信するために用いるチャンネルである。操作チャンネルは、全ての制御指示部110が共通に使用するチャンネルであり、制御指示部110が1つ以上動作している間、1つの操作チャンネルがモーション制御装置10内に存在する。本実施形態では、操作チャンネルを、「操作用チャンネル」と呼んでもよい。

[0027] 「APIチャンネル」は、API情報を制御指示部110から連携部310に伝えるために使用するチャンネルである。APIチャンネルは、制御指示部110ごとに少なくとも1つ存在する。本実施形態では、APIチャンネルを、「制御指令用チャンネル」と呼んでもよい。

- [0028] 「状態チャンネル」は、制御対象装置 20 からのフィードバック情報を格納するチャンネルである。状態チャンネルは、複数の制御指示部 110 が共通に使用するチャンネルであり、モーション制御装置 10 内には 1 又は複数の状態チャンネルが存在する。フィードバック情報とは、制御対象装置 20 の状態（動作状態）を示す情報であり、制御対象装置 20 が内部に備えるセンサ又は制御対象装置 20 に接続されたセンサ等により測定される。フィードバック情報には、例えば、軸の位置、軸の回転速度、アームの位置、ベルトコンベアの位置、制御対象装置 20 の温度など、あらゆる情報が含まれる。制御指示部 110 は、状態チャンネルを参照することで、制御対象装置 20 の動作状態を把握することができる。
- [0029] 本実施形態では、1 つの制御指示部 110 に関連付けられた A P I チャンネル、状態チャンネルなどをまとめて「デバイス」と呼ぶ。後述するように、A P I チャンネルは、制御指示部 110 からデバイス作成要求があった際に、その制御指示部 110 専用のチャンネルとしてメモリ上に確保される。一方、状態チャンネルは、最初にその状態チャンネルを利用する制御指示部 110 からデバイス作成要求があった際にメモリ上に確保され、確保された後は他の制御指示部 110 から利用可能となる。
- [0030] ここで、A P I チャンネル及び状態チャンネルの具体例を説明する。図 3 は、A P I チャンネル及び状態チャンネルの一例を説明するための図である。図 3 (a) に、A P I チャンネルに格納されるデータ例を示す。「デバイス識別子」には、生成されたデバイスを一意に識別するためのデバイス識別子が格納される。「モジュール識別子」については後述する。「制御指令識別子」には、制御指令の内容（例えば、位置決め（P T P）、J O G 動作、直線補間、円弧補間等）を特定する情報が格納される。制御指令識別子は、制御指令の種別を一意に示す識別子であってもよいし、A P I 関数を一意に示す識別子であってもよい。
- [0031] 「A P I バッファ識別子」には、A P I バッファ 340 を一意に特定する A P I バッファ識別子が格納される。「記録中フラグ」については後述する

。「指令値」には、軸ごとの指令値が格納される。例えば制御指令識別子が P T P 動作の場合、指令値には、移動先の位置を示す値等が軸ごとに格納される。「生存報告」については後述する。

[0032] 図 3 (b) に、状態チャンネルに格納されるデータ例を示す。「モジュール識別子」については後述する。「エラーコード」には、何らかのエラーが生じたことでフィードバック情報を状態チャンネルに格納できない場合に、発生したエラーの内容を示すエラーコードが格納される。「フィードバック情報」には、制御対象装置 2 0 から取得されたフィードバック情報が、制御対象装置 2 0 ごと (軸ごと) に格納される。

[0033] <ハードウェア構成>

図 4 は、モーション制御装置 1 0 のハードウェア構成例を示す図である。モーション制御装置 1 0 は、C P U (Central Processing Unit) 1 1、メモリ 1 2、H D D (HardDisk) や S S D (Solid State Drive) 等の記憶装置 1 3、有線又は無線通信を行う通信 I F (Interface) 1 4、入力操作を受け付ける入力デバイス 1 5、及び情報の出力を行う出力デバイス 1 6 を有する。入力デバイス 1 5 は、例えば、キーボード、タッチパネル、マウス及び／又はマイク等である。出力デバイス 1 6 は、例えば、ディスプレイ及び／又はスピーカ等である。

[0034] <機能構成>

図 5 は、モーション制御装置 1 0 の機能構成例を示す図である。共有メモリ 2 0 0 は、非リアルタイム O S 1 0 0 側の各機能部とリアルタイム O S 3 0 0 側の各機能部とから共通に参照及び書込み可能なメモリである。共有メモリ 2 0 0 には、操作チャンネル、A P I チャンネル及び状態チャンネルに対応するメモリ領域が確保される。制御指示部 1 1 0 及び連携部 3 1 0 は、当該メモリ領域にデータを書き込んだり、当該メモリ領域からデータを読み出したることでデータの送受信を行う。

[0035] (非リアルタイム O S)

非リアルタイム O S 1 0 0 上では、1 以上の制御指示部 1 1 0、及び 1 以

上のデバイス管理部１２０が動作する。

- [0036] 制御指示部１１０は、制御指令を発行することで、制御対象装置２０をモーション制御する機能を有する。制御指示部１１０には、ユーザ作成プログラム１１１、指令受付部１１２、及びＩＦ（Ｎ側）部１１３が含まれる。
- [0037] ユーザ作成プログラム１１１は、モーション制御装置１０を利用するユーザが作成するプログラムである。ユーザは、ユーザ作成プログラム１１１において、非リアルタイムＯＳ１００側に用意されている多様なＡＰＩ関数を呼び出すことで、任意のモーション制御を実現するプログラムを作成することができる。なお、ユーザ作成プログラム１１１は予めモーション制御装置１０にプリセットされていてもよい。すなわち、本実施形態において、ユーザ作成プログラム１１１は、必ずしもユーザが作成したプログラムに限定されない。
- [0038] 指令受付部１１２は、ユーザ作成プログラム１１１にＡＰＩ関数を提供する。具体的には、指令受付部１１２は、ユーザ作成プログラム１１１にて呼び出されたＡＰＩ関数についてのＡＰＩ情報を、ＡＰＩチャンネルに格納するようにＩＦ（Ｎ側）部１１３に指示する機能を有する。
- [0039] なお、指令受付部１１２は、制御指令ライブラリ１１２１Ａ～１１２１Ｚ（以下、制御指令ライブラリＡ～Ｚを特に区別しない場合は、単に制御指令ライブラリ１１２１と言う。）及び通信ライブラリ１１２２Ａ～１１２２Ｚ（以下、通信ライブラリＡ～Ｚを特に区別しない場合は、単に通信ライブラリ１１２２と言う。）のうち、ユーザ作成プログラム１１１が実際にプログラム内で参照するライブラリに含まれるＡＰＩ関数を提供する。
- [0040] 制御指令ライブラリ１１２１Ａ～１１２１Ｚは、それぞれ、制御対象装置２０をモーション制御する処理を行うための複数のＡＰＩ関数のロードモジュール（実行ファイル）をひとまとめにしたライブラリである。また、通信ライブラリ１１２２Ａ～１１２２Ｚは、モーション制御装置１０が制御対象装置２０との間で行う通信に関する処理を行うための複数のＡＰＩ関数のロードモジュール（実行ファイル）をひとまとめにしたライブラリである。

- [0041] なお、当該通信に関する処理とは、例えば、断線などで切断された通信路が復帰した際に、通信を再開するタイミングをユーザ作成プログラム 111 から指定する処理、モーション制御装置 10 及び制御対象装置 20 の間の通信路におけるパケットロスの監視等である。通信ライブラリ 1122 が提供する API 関数が呼ばれた場合も、制御指令ライブラリ 1121 が提供する API 関数が呼ばれた場合と同様、API 情報が連携部 310 を介して定周期処理部 360 に通知される。
- [0042] IF (N側) 部 113 は、操作チャンネル及び API チャンネルにデータを格納する機能、操作チャンネル、API チャンネル及び状態チャンネルからデータを取得する機能、及び、操作チャンネル、API チャンネル及び状態チャンネルを管理する機能を有する。
- [0043] なお、指令受付部 112 及び IF (N側) 部 113 は、ユーザ作成プログラム 111 が呼び出した API 関数を受け付けて API チャンネルに格納する役割を担うことから、指令受付部 112 及び IF (N側) 部 113 を、まとめて「受付部」と呼んでもよい。
- [0044] デバイス管理部 120 は、制御指示部 110 の生死確認を行う。また、デバイス管理部 120 は、API バッファ 340 への API 情報の格納に関する各種の処理等を行う。デバイス管理部 120 は、制御指示部 110 がデバイスを作成する際に起動される。なお、デバイスには、より詳細には、API チャンネル及び状態チャンネルに加えて、デバイス管理部 120 も含まれる。
- [0045] (リアルタイム OS)
- リアルタイム OS 300 上では、連携部 310 及び定周期処理部 360 が動作する。
- [0046] 連携部 310 は、制御指示部 110 で行われる処理と、定周期処理部 360 で行われる処理とを連携するための機能部であり、制御指示部 110 から受けた API 情報を定周期処理部 360 に伝える機能を有する。連携部 310 は、FIFO キューとして機能する API バッファ 340 を備えており、制御指示部 110 から API チャンネルを介して伝えられる制御指令 (API

情報)をAPIバッファ340に蓄積する。また、連携部310は、APIバッファ340に蓄積されたAPI情報を取り出して定周期処理部360に渡す。また、連携部310は、操作チャネル、APIチャネル及び状態チャネルの状態管理(生成、維持及び破棄)を行う。

[0047] 連携部310には、上述の各機能を実行する機能部として、メイン処理部320、API実行要求処理部330、APIバッファ340及び指令実行処理部350が含まれる。メイン処理部320には、更に、IF(R側)部321が含まれる。

[0048] メイン処理部320は、制御指示部110からの要求を受けた場合に、リアルタイムOSのメモリ領域にAPIバッファ340を生成する機能を有する。本実施形態では、メイン処理部320を「キュー生成部」と呼んでもよい。

[0049] IF(R側)部321は、操作チャネル、APIチャネル及び状態チャネルの作成(共有メモリ200へのメモリ領域割り当て)、割り当てたメモリ領域の管理、及び、APIチャネル及び状態チャネルの破棄(共有メモリ200の解放)に関する各種の処理を行う。本実施形態では、IF(R側)部321を、「チャネル管理部」と呼んでもよい。或いは、IF(R側)部321は連携部310の一部であることから、連携部310を「チャネル管理部」と呼んでもよい。

[0050] 具体的には、IF(R側)部321は、リアルタイムOS300の指示によりIF(R側)部321自身が起動した場合に、操作チャネルを共有メモリ200に生成する。また、IF(R側)部321は、制御指示部110(より詳細にはIF(N側)部113)からの要求を受けてAPIチャネルを生成する。また、IF(R側)部321は、制御指示部110(より詳細にはIF(N側)部113)からの指示を受けて状態チャネルを生成する。

[0051] API実行要求処理部330は、APIチャネルからAPI情報を取得し、取得したAPI情報をAPIバッファ340に格納する。本実施形態では、API実行要求処理部330を、「格納部」と呼んでもよい。なお、連携

部 310 内では、前述した“デバイス”ごとに1つのAPI 実行要求処理部 330 が動作する。例えば、非リアルタイムOS 110 内で4つの制御指示部 110 がそれぞれデバイスを1つずつ生成した場合、連携部 310 内では、各デバイスに対応するAPI 実行要求処理部 330 が4つ動作することになる。本実施形態では、「デバイス」には、API チャンネル、状態チャンネル及びデバイス管理部 120 に加えて、API 実行要求処理部 330 も含まれることとしてもよい。

[0052] 指令実行処理部 350 は、API バッファ 340 に格納されているAPI 情報を取得し、取得したAPI 情報を制御指令モジュール 370 に渡す。本実施形態では、指令実行処理部 350 を、「指令処理部」と呼んでもよい。指令実行処理部 350 は、API 実行要求処理部 330 のようにデバイス単位で並列に動作するのではなく、予め設定ファイルで指定された数の指令実行処理部 350 が連携部 310 の中で並列に動作する。

[0053] 定周期処理部 360 は、API 情報（制御指令情報）に基づいて、制御対象装置 20 に対してモーション制御サイクルごとに実行させるべき動作を示す補間指令をモーション制御サイクルごとに送信することで、制御対象装置 20 をモーション制御する機能を有する。また、定周期処理部 360 には、更に、制御指令モジュール 370、IF 部 380 及び通信モジュール 390 が含まれる。

[0054] 制御指令モジュール 370A～370Z（以下、制御指令モジュールA～Zを特に区別しない場合は、単に制御指令モジュール 370 と言う。）は、API 情報（制御指令情報）に基づいて補間指令を算出する。また、制御指令モジュール 370 は、算出した補間指令を、IF 部 380 に渡す。なお、制御指令モジュール 370 は、API 情報と、制御対象装置 20 のフィードバック情報に含まれる制御対象装置 20 の状態とに基づいて補間指令を算出するようにしてもよい。本実施形態では、制御指令モジュール 370 を、「制御部」と呼んでもよい。

[0055] 本実施形態では、制御指令ライブラリ 1121A が提供するAPI 関数に

対応するAPI情報（制御指令情報）は、制御指令モジュール370Aで処理される。同様に、制御指令ライブラリ1121B～1121Zが提供するAPI関数に対応するAPI情報（制御指令情報）は、それぞれ、制御指令モジュール370B～制御指令モジュール370Zで処理される。つまり、制御指令ライブラリ1121A～1121Zは、それぞれ制御指令モジュール370A～370Zと1対1で対応づけられる。なお、これは実装方法の一例であり、必ずしも1対1で対応づけられていなくてもよい。例えば、複数の制御指令ライブラリ1121が1つの制御指令モジュール370に対応づけられていてもよい。本実施形態では、制御指令モジュール370を一意に特定する識別子を、「モジュール識別子」と言う。

[0056] 制御指令モジュール370の具体例としては、前述したPTP動作やJOG動作の他に、例えば、特定の動作（あるビットが1になった場合等）や、特定の軸が指定位置まで移動したことをトリガーとして、所定の動作を実行させる制御指令モジュール370、制御対象装置20との間の通信インターフェースから特定の値を取得する制御指令モジュール370が挙げられる。また、指定された制御対象装置20の動作（位置、速度、加速度等）を記録する制御指令モジュール370、制御対象装置20のゆがみ等に起因する位置の誤差等を補正する制御指令モジュール370等が挙げられる。当然ながら、これらの機能に対応するAPI関数を提供する制御指令ライブラリ1121も、非リアルタイムOS110上に用意されている。

[0057] IF部380は、制御指令モジュール370及び通信モジュール390が共通に参照可能なメモリ領域を提供する。IF部380は、制御指令モジュール370と通信モジュール390との間で補間指令値及びフィードバック情報を相互に受渡しするために用いられる。

[0058] 通信モジュール390A～390Z（以下、通信モジュールA～Zを特に区別しない場合は、単に通信モジュール390と言う。）は、それぞれ、異なる通信インターフェース規格に対応している。また、本実施形態では、通信ライブラリ1122Aが提供するAPI関数が呼び出されることで定周期

処理部 360 に通知される API 情報は、通信モジュール 390A で処理される。同様に、通信ライブラリ 1122B ~ 1122Z が提供する API 関数が呼び出されることで定周期処理部 360 に通知される API 情報は、それぞれ、通信モジュール 390B ~ 通信モジュール 390Z で処理される。つまり、通信ライブラリ 1122A ~ 1122Z は、それぞれ通信モジュール 390A ~ 390Z と 1 対 1 で対応づけられる。なお、これは実装方法の一例であり、必ずしも 1 対 1 で対応づけられていなくてもよい。例えば、複数の通信ライブラリ 114 が 1 つの通信モジュール 390 に対応づけられていてもよい。

[0059] 以上説明した、制御指示部 110、デバイス管理部 120、連携部 310、定周期処理部 360 は、モーション制御装置 10 の CPU 11 が、メモリ 12 又は記憶装置 13 に記憶されたプログラムを実行することにより実現することができる。また、当該プログラムは、記憶媒体に格納することができる。当該プログラムを格納した記憶媒体は、非一時的な記憶媒体 (Non-transitory computer readable medium) であってもよい。非一時的な記憶媒体は特に限定されないが、例えば、USB メモリ又は CD-ROM 等の記憶媒体であってもよい。なお、当該プログラムには、ユーザ作成プログラム 111 が含まれていてもよいし、含まれていなくてもよい。

[0060] <処理手順>

続いて、モーション制御装置 10 が行う具体的な処理手順について説明する。

[0061] (デバイス作成処理)

図 6 及び図 8 は、デバイス作成処理の一例を説明するための図である。また、図 7 及び図 9 は、デバイス作成処理が行われる際に生成される各種管理情報の一例を説明するための図である。図 6 ~ 図 9 を用いて、制御指示部 110 が、ユーザ操作により、制御指示部 (1) 110-1、制御指示部 (2) 110-2 の順に起動された場合に、各々が実行するデバイス作成処理について説明する。

[0062] [前提条件]

最初に、デバイス作成処理を説明するにあたり前提とする条件について説明する。図6～図9において、制御指示部(1)110-1は、制御指令ライブラリ1121Aが提供するAPI関数を呼び出すものとする。同様に、制御指示部(2)110-2は、制御指令ライブラリ1121A、制御指令ライブラリ1121B及び制御指令ライブラリ1121Cがそれぞれ提供するAPI関数を呼び出すものとする。

[0063] また、制御指示部(1)110-1に含まれるユーザ作成プログラム111、指令受付部112及びIF(N側)部113を、それぞれ、ユーザ作成プログラム111-1、指令受付部112-1、IF(N側)部113-1と呼ぶ。また、制御指示部(1)110-1に対応するデバイス管理部120を、デバイス管理部120-1と呼ぶ。

[0064] また、制御指示部(2)110-2に含まれるユーザ作成プログラム111、指令受付部112及びIF(N側)部113を、それぞれ、ユーザ作成プログラム111-2、指令受付部112-2及びIF(N側)部113-2と呼ぶ。また、制御指示部(2)110-2に対応するデバイス管理部120を、デバイス管理部120-2と呼ぶ。

[0065] また、制御指示部(1)110-1(より詳細には制御指示部(1)110-1が生成したデバイス)に対応するAPI実行要求処理部330を、API実行要求処理部(1)330-1と呼ぶ。同様に、制御指示部(2)110-2(より詳細には制御指示部(2)110-2が生成したデバイス)に対応するAPI実行要求処理部330を、API実行要求処理部(2)330-2と呼ぶ。

[0066] また、起動設定情報400には、制御指令モジュール370A～370C、及び通信モジュール390A～390Zが起動対象として指定されていると仮定する。なお、起動設定情報400とは、定周期処理部360にて動作させる制御指令モジュール370及び通信モジュール390を指定する情報である。

[0067] 〔制御指示部（１）が行うデバイス作成処理〕

図６及び図７を用いて、制御指示部（１）が行うデバイス作成処理を説明する。ユーザ操作により制御指示部（１）１１０－１が起動されると、ユーザ作成プログラム１１１－１は、制御対象装置２０へのモーション制御を開始する準備を行うことの準備指示（以下、「デバイス作成要求」と言う。）を指令受付部１１２－１に通知する。当該通知は、具体的には、ユーザ作成プログラム１１１－１がデバイスの作成を指示するＡＰＩ関数を呼び出すことで行われる。続いて、指令受付部１１２－１は、ＩＦ（Ｎ側）部１１３－１に、デバイス作成要求を通知する（Ｓ１００）。デバイス作成要求には、ユーザ作成プログラム１１１－１が発行する（後述する「モーション制御処理」にて発行予定である）制御指令を処理する制御指令モジュール３７０のモジュール識別子（制御指令モジュールＡ）が含まれる。また、指令受付部１１２－１は、ステップＳ１００の処理手順の前又は後で、デバイス管理部１２０－１を起動する。

[0068] 次に、ＩＦ（Ｎ側）部１１３－１は、リアルタイムＯＳ３００側で連携部３１０が起動しているか否かを確認する。起動していない場合、リアルタイムＯＳ３００に対して連携部３１０の起動指示を送信する（Ｓ１０１）。なお、連携部３１０が起動していない状態とは、例えば、モーション制御装置１０を起動した直後などである。起動指示を受けたリアルタイムＯＳは、連携部３１０を起動する。

[0069] 続いて、連携部３１０のメイン処理部３２０は、起動設定情報４００で指示されている制御指令モジュール３７０及び通信モジュール３９０を起動する（Ｓ１０２）。

[0070] 続いて、リアルタイムＯＳ３００の指示により起動した連携部３１０のＩＦ（Ｒ側）部３２１は、共有メモリ２００に操作チャネルに使用するメモリ領域を確保し（Ｓ１０３）、メモリ領域の確保が完了したこと（つまり、操作チャネルを作成したこと）をＩＦ（Ｎ側）部１１３－１に通知する（Ｓ１０４）。このとき、ＩＦ（Ｒ側）部３２１は、共有メモリ２００において操

作チャンネルのために確保されたアドレス範囲も I F (N側) 部 1 1 3 - 1 に通知する。I F (R側) 部 3 2 1 は、操作チャンネルに確保したアドレス範囲をアドレス管理情報 (図 7 (e)) に格納する。また、I F (N側) 部 1 1 3 - 1 は、操作チャンネルのアドレス範囲を、各 I F (N側) 部 1 1 3 から共通に参照可能なリソース (ファイル又は共通の変数等) に格納する。なお、I F (N側) 部 1 1 3 - 1 は、操作チャンネルのアドレス範囲を、更に、アドレス管理情報 (図 7 (d)) に格納することとしてもよい。図 7 (d) は、操作チャンネルのアドレス範囲が格納された状態を示している。操作チャンネルの作成が完了したことの通知を受けた I F (N側) 部 1 1 3 - 1 は、他の制御指示部 1 1 0 の I F (N側) 部 1 1 3 との間で操作チャンネルの利用について競合が生じないようにするため、操作チャンネルの利用権限を取得する (S 1 0 5)。

[0071] 利用権限が取得できた場合、I F (N側) 部 1 1 3 - 1 は、デバイス作成要求を、操作チャンネルを介して I F (R側) 部 3 2 1 に通知する (S 1 0 6)。デバイス作成要求には、ステップ S 1 0 0 の処理手順で通知されたモジュール識別子 (制御指令モジュール A) が含まれる。

[0072] デバイス作成要求を受けた I F (R側) 部 3 2 1 は、デバイス識別子を例えば「D 1」に決定する。また、I F (R側) 部 3 2 1 は、A P I チャンネルに使用するメモリ領域を確保すると共に、A P I チャンネル識別子を例えば「A P I チャンネル 1」に決定する (S 1 0 7)。また、I F (R側) 部 3 2 1 は、決定したデバイス識別子及び A P I チャンネル識別子を対応づけて、デバイス管理情報 (図 7 (b)) に格納する。

[0073] また、I F (R側) 部 3 2 1 は、共有メモリ 2 0 0 に、制御指令モジュール 3 7 0 A に対応する状態チャンネル A に使用するメモリ領域を確保する (S 1 0 8)。I F (R側) 部 3 2 1 は、作成した状態チャンネル A と制御指令モジュール 3 7 0 A との対応関係を、状態チャンネル管理情報 (図 7 (c)) として保持する。また、I F (R側) 部 3 2 1 は、A P I チャンネル 1 及び状態チャンネル A に確保したメモリ領域のアドレス範囲を、アドレス管理情報 (図

7 (e)) に格納する。

[0074] 続いて、メイン処理部320は、API実行要求処理部(1)330-1を起動する(S109)。なお、ステップS107～ステップS109の処理手順は、図6に示す順序であることに限定されず、どのような順序で行われてもよい。

[0075] 続いて、IF(R側)部321は、デバイスの作成が完了したこと(APIチャンネル1及び状態チャンネルAの作成が完了したこと)をIF(N側)部113-1に通知する(S110)。当該通知は、作成されたAPIチャンネル識別子(APIチャンネル1)、APIチャンネル1及び状態チャンネルAに確保されたメモリ領域のアドレス範囲、並びにデバイス識別子(D1)を含む。

[0076] デバイスの作成が完了したとの通知を受けたIF(N側)部113-1は、APIチャンネル1及び状態チャンネルAに確保されたメモリ領域のアドレス範囲を、アドレス管理情報(図7(d))に格納する。続いて、IF(N側)部113-1は、操作チャンネルの利用権限を解放し(S111)、デバイスの作成が完了したことを、指令受付部112-1に通知する。また、指令受付部112-1は、デバイスの作成が完了したことをユーザ作成プログラム111-1に通知する(S112)。当該通知には、デバイス識別子(D1)及びAPIチャンネル識別子(APIチャンネル1)が含まれる。当該通知は、例えば、デバイス作成を指示するAPI関数の戻り値であってもよい。

[0077] ユーザ作成プログラム111-1は、通知されたデバイス識別子(D1)及びAPIチャンネル識別子(APIチャンネル1)を対応づけて、デバイス管理情報(図7(a))として保持する。なお、デバイス管理情報として保持するとは、具体的には、デバイス作成を指示するAPI関数の戻り値として通知されたデバイス識別子及びAPIチャンネル識別子を変数に格納して保持することを意図しているが、これに限定されず、例えばファイル等で保持することも可能である。

[0078] [制御指示部(2)が行うデバイス作成処理]

図8及び図9を用いて、制御指示部(2)が行うデバイス作成処理を説明する。ユーザの操作により制御指示部(2)110-2が起動されると、ユーザ作成プログラム111-2は、デバイス作成要求を指令受付部112-2に通知する。指令受付部112-2は、IF(N側)部113-2に、デバイス作成要求を通知する(S150)。デバイス作成要求には、ユーザ作成プログラム111-2が発行する(後述する「モーション制御処理」にて発行予定である)制御指令を処理する制御指令モジュール370のモジュール識別子(制御指令モジュールA、制御指令モジュールB及び制御指令モジュールC)が含まれる。また、指令受付部112-2は、ステップS100の処理手順の前又は後で、デバイス管理部120-2を起動する。

[0079] 次に、IF(N側)部113-2は、リアルタイムOS300側で連携部310が起動しているか否かを確認する。ここでは、制御指示部(1)110-1が連携部310を既に起動していることから、IF(N側)部113-2は、連携部310の起動処理を行わない。続いて、IF(N側)部113-2は、操作チャネルの利用権限を取得する(S151)。

[0080] 利用権限が取得できた場合、IF(N側)部113-2は、デバイス作成要求を、操作チャネルを介してIF(R側)部321に通知する(S152)。デバイス作成要求には、モジュール識別子(制御指令モジュールA、制御指令モジュールB及び制御指令モジュールC)が含まれる。なお、IF(N側)部113-2は、操作チャネルのアドレス範囲を、各IF(N側)部113から共通に参照可能なリソース(ファイル又は共通の変数等)から取得する。また、IF(N側)部113-2は、操作チャネルのアドレス範囲を、アドレス管理情報(図9(d))に格納することとしてもよい。図9(d)は、操作チャネルのアドレス範囲が格納された状態を示している。

[0081] デバイス作成要求を受けたIF(R側)部321は、デバイス識別子を例えば「D2」に決定する。また、IF(R側)部321は、共有メモリ200に、APIチャネルに使用するメモリ領域を確保するとともにAPIチャネル識別子を例えば「APIチャネル2」に決定する(S153)。また、

I F (R側) 部 3 2 1 は、決定したデバイス識別子及び A P I チャンネル識別子に対応づけて、デバイス管理情報 (図 9 (b)) に追加する。

[0082] 続いて、I F (R側) 部 3 2 1 は、共有メモリ 2 0 0 に、制御指令モジュール 3 7 0 A に対応する状態チャンネル A、制御指令モジュール 3 7 0 B に対応する状態チャンネル B 及び制御指令モジュール 3 7 0 C に対応する状態チャンネル C に使用するメモリ領域を確保する。なお、制御指令モジュール 3 7 0 A に対応する状態チャンネル A は既に作成済み (図 6 のステップ S 1 0 8) であることから、I F (R側) 部 3 2 1 は、状態チャンネル B 及び状態チャンネル C に使用するメモリ領域の確保を行う (S 1 5 4、S 1 5 5)。

[0083] 続いて、I F (R側) 部 3 2 1 は、作成した状態チャンネル B 及び状態チャンネル C が、それぞれ制御指令モジュール 3 7 0 B 及び制御指令モジュール 3 7 0 C に対応づけられることを、I F (R側) 部 3 2 1 が保持する状態チャンネル管理情報 (図 9 (c)) に追加する。また、I F (R側) 部 3 2 1 は、A P I チャンネル 2、状態チャンネル B 及び状態チャンネル C に確保したメモリ領域のアドレス範囲を、I F (R側) 部 3 2 1 が保持するアドレス管理情報 (図 9 (e)) に追加する。

[0084] 続いて、I F (R側) 部 3 2 1 は、デバイスの作成が完了したこと (A P I チャンネル 2、状態チャンネル B 及び状態チャンネル C の作成が完了したこと) を I F (N側) 部 1 1 3 - 2 に通知する (S 1 5 7)。当該通知には、A P I チャンネル 2、状態チャンネル A、状態チャンネル B 及び状態チャンネル C に確保されたメモリ領域のアドレス範囲、A P I チャンネル識別子 (A P I チャンネル 2) 並びにデバイス識別子 (D 2) が含まれる。

[0085] デバイスの作成が完了したとの通知を受けた I F (N側) 部 1 1 3 - 2 は、A P I チャンネル 2、状態チャンネル A、状態チャンネル B 及び状態チャンネル C に確保されたメモリ領域のアドレス範囲を、アドレス管理情報 (図 9 (d)) に格納する。

[0086] 続いて、I F (N側) 部 1 1 3 - 2 は、操作チャンネルの利用権限を解放し (S 1 5 8)、デバイスの作成が完了したことを、指令受付部 1 1 2 - 2 に

通知する。また、指令受付部 112-2 は、デバイスの作成が完了したことをユーザ作成プログラム 111-2 に通知する (S159)。当該通知には、デバイス識別子 (D2) 及び API チャンネル識別子 (API チャンネル 2) が含まれる。ユーザ作成プログラム 111-2 は、通知されたデバイス識別子 (D2) 及び API チャンネル識別子 (API チャンネル 2) を対応づけて、デバイス管理情報 (図 9 (a)) として保持する。

[0087] なお、以上説明したデバイス作成処理では、デバイス作成要求を取得した IF (R側) 部 321 側は、API チャンネル及び状態チャンネルの両方を作成するようにした。しかしながら、本実施形態は、これに限定されない。例えば、図 6 のステップ S100～ステップ S112 及び図 7 のステップ S150～ステップ S159 の処理手順では API チャンネルを作成し、その後、同様の処理手順を繰り返すことで、API チャンネルの作成と状態チャンネルの作成を別々のタイミングで行うようにしてもよい。

[0088] 具体的には、ユーザ作成プログラム 111 は、モジュール識別子を含まないデバイス作成要求を IF (N側) 部 113 に通知し、IF (N側) 部 113 も同様に、モジュール識別子を含まないデバイス作成要求を IF (R側) 部 321 に通知する。当該デバイス作成要求を受けた IF (R側) 部 321 は、API チャンネルを作成し、デバイスの作成が完了したことを IF (N側) 部 113 に通知する。IF (N側) 部 113 は、デバイスの作成が完了したことをユーザ作成プログラム 111 に通知する。

[0089] その後、ユーザ作成プログラム 111 は、例えば制御指令を発行する前などのタイミングで、当該制御指令を処理する制御指令モジュール 370 のモジュール識別子を含む状態チャンネル作成要求を IF (N側) 部 113 に通知し、IF (N側) 部 113 も同様に、モジュール識別子を含む状態チャンネル作成要求を IF (R側) 部 321 に通知する。当該状態チャンネル作成要求を受けた IF (R側) 部 321 は、状態チャンネルを作成し、状態チャンネルの作成が完了したことを IF (N側) 部 113 に通知する。IF (N側) 部 113 は、状態チャンネルの作成が完了したことをユーザ作成プログラム 111 に

通知する。

[0090] (モーション制御処理)

続いて、モーション制御装置 10 が制御対象装置 20 をモーション制御する際の処理手順について具体的に説明する。

[0091] [前提条件]

以下の説明では、制御指示部 110 が、PTP 制御を実行可能な制御指令モジュール 370A を使用し、軸 1 に対応する制御対象装置 (軸 1) 20-1 及び軸 2 に対応する制御対象装置 (軸 2) 20-2 に対してモーション制御を行うものとする。また、制御指示部 110 が生成したデバイスのデバイス識別子は D1 であり、当該デバイスに対応する API チャンネル識別子は API チャンネル 1 であるものとする。また、API チャンネル 1 及び制御指令モジュール 370A に対応する状態チャンネル A は既に生成されているものとする。また、以下の説明において、制御指示部 110 は、PTP 制御に関する API 関数を呼び出すことで、軸 1 及び軸 2 の 2 軸を同時に制御するものとする。

[0092] 図 10 及び図 11 は、モーション制御装置 10 が行うモーション制御処理の一例を示す図である。まず、図 10 を用いて、API バッファ 340 の作成から、API バッファ 340 に API 情報がキューイングされるまでの処理手順について説明する。

[0093] [API バッファ作成、API 情報蓄積]

まず、ユーザ作成プログラム 111 は、API バッファ 340 の作成を指示する API 関数を呼び出す。当該 API 関数が呼び出されると、指令受付部 112 は、API バッファの作成を IF (N 側) 部 113 に通知する (S200)。当該通知には、作成する API バッファ 340 に対応する API バッファ識別子が含まれる。API バッファ識別子はどのように決定されてもよいが、例えば、ステップ S200 の処理手順を行う前に、例えばメイン処理部 320 から払い出されることとしてもよい。ここでは API バッファ識別子は「1」であると仮定する。

- [0094] IF (N側) 部 1 1 3 は、操作チャネルを介して、APIバッファ 3 4 0 の作成をメイン処理部 3 2 0 に要求する (S 2 0 1)。当該要求には、API バッファ識別子が含まれる。メイン処理部 3 2 0 は、API バッファ識別子「1」に対応する API バッファ 3 4 0 を作成する (メモリ領域を確保する) (S 2 0 2)。
- [0095] 続いて、ユーザ作成プログラム 1 1 1 が、API バッファ識別子「1」を指定して API 情報の記録を開始する (API 情報のキューイングを開始すること) を指示する API 関数を呼び出すと、指令受付部 1 1 2 は、当該 API 情報の記録を開始することを IF (N側) 部 1 1 3 に通知する (S 2 0 3)。IF (N側) 部 1 1 3 は、当該 API 情報の記録を開始することをデバイス管理部 1 2 0 に通知する (S 2 0 4)。デバイス管理部 1 2 0 は、API チャネル 1 の「API バッファ識別子」に「1」をセットすると共に、「記録中フラグ」をセットする (S 2 0 5)。
- [0096] 記録中フラグは、ユーザ作成プログラム 1 1 1 が呼び出した API 関数に対応する API 情報を、一旦 API バッファ 3 4 0 にキューイングしてから定周期処理部 3 6 0 に渡すのか、又は、ユーザ作成プログラム 1 1 1 が呼び出した API 関数に対応する API 情報を、API バッファ 3 4 0 にキューイングせずに直接、定周期処理部 3 6 0 に渡すのかを切り替えるために用いられる。具体的には、フラグがセットされている間に API チャネルに API 情報が格納された場合、API チャネルに格納された API 情報は、API 実行要求処理部 3 3 0 により API バッファ 3 4 0 にキューイングされる。一方、当該フラグがセットされていない状態で API チャネルに API 情報が格納された場合、API チャネルに格納された API 情報は、API 実行要求処理部 3 3 0 により取得されて定周期処理部 3 6 0 に渡され、定周期処理部 3 6 0 で実行される。
- [0097] 続いて、ユーザ作成プログラム 1 1 1 は、軸 1 に指示する指令値、軸 2 に指示する指令値及びデバイス識別子 (ここでは D 1) を引数に指定して、PTP 制御を行うための API 関数を呼び出す。当該 API 関数が呼び出され

ると、指令受付部 112 は、呼び出された API 関数に対応する API 情報を IF (N側) 部 113 に通知する (S206)。

[0098] 続いて、IF (N側) 部 113 は、API 情報を API チャンネル 1 に格納する (S207)。具体的には、API チャンネル 1 の「デバイス識別子」、「モジュール識別子」及び「制御指令識別子」に、それぞれ、「D1」、「制御指令モジュール A」及び「PTP」を格納する。また、「指令値」には軸 1 の指令値及び軸 2 の指令値を格納する。「API バッファ識別子」及び「記録中フラグ」には、ステップ S205 の処理手順で格納された情報がそのまま格納されている。

[0099] なお、API チャンネルに格納する「モジュール識別子」は、制御指令モジュール 370 を明示的に特定可能な識別子に限られず、制御指令モジュール 370 を暗示的に示す識別子であってもよい。また、API チャンネルから「モジュール識別子」を省略するようにしてもよい。本実施形態では、「制御指令識別子」で示される制御指令を実行する制御指令モジュール 370 は 1 つであることから、連携部 310 は、制御指令識別子に基づいて、制御指令モジュール 370 を一意に特定することが可能であるためである。

[0100] API 実行要求処理部 330 は、API チャンネル 1 の「記録中フラグ」がセットされている場合、API チャンネル 1 に格納されている API 情報（「デバイス識別子」、「モジュール識別子」、「制御指令識別子」、「API バッファ識別子」及び「指令値」）を取得する (S208)。また、取得した API 情報を、ステップ S208 で取得した「API バッファ識別子」に対応する API バッファ 340 にキューイングする (S209)。

[0101] 制御指示部 110 が複数の API を呼び出す場合、ステップ S206 ～ ステップ S209 の処理手順が繰り返されることで、複数の API 情報が API バッファ 340 にキューイングされる。

[0102] API バッファ 340 にキューイングさせる API 関数の呼出が完了すると、ユーザ作成プログラム 111 は、API バッファ 340 への API 情報の記録を終了する (API 情報のキューイングを終了する) ことを指示する

A P I関数を呼び出す。制御指示部110は、A P Iバッファ340へのA P I情報の記録を終了することをI F（N側）部113に通知し（S210）、I F（N側）部113は、当該通知をデバイス管理部120に通知する（S211）。デバイス管理部120は、A P Iチャンネル1にセットされている記録中フラグを消去する（S212）。

[0103] A P Iバッファ340は、ユーザ作成プログラム111（より詳細にはデバイス）の指示により生成されるが、1つのユーザ作成プログラム111が複数のA P Iバッファ340を作成することが可能である。また、複数のユーザ作成プログラム111が同一のA P Iバッファ340を共用することも可能である。本実施形態では、1つのユーザ作成プログラム111が複数のA P Iバッファ340を利用可能とすることで、複数の複雑なモーション制御を同時に動作させることを可能としている。また、複数のユーザ作成プログラム111が同一のA P Iバッファ340を共用することを可能とすることで、ユーザがユーザ作成プログラム111を作成する際の自由度を高めることができる。

[0104] [制御対象装置の制御]

次に、図11を用いて、A P Iバッファ340にキューイングされたA P I情報に従って、制御対象装置（軸1）20-1及び制御対象装置（軸2）20-2を制御する際の処理手順について説明する。

[0105] 図11に示すように、制御指令モジュール370Aは、制御指令モジュール370Aが同時に制御可能な複数の軸について、軸ごとに、指令値と、指令値を更新したことを示す実行フラグとを格納するメモリ領域371を保持している。図11の例では、制御指令モジュール370Aは最大128軸まで同時制御することが可能であり、軸1～軸128の各々について、指令値及び実行フラグを格納するためのメモリ領域371を有している。実行フラグには、「実行中」又は「実行済み」のいずれかが格納される。「実行中」は、制御指令モジュール370Aがモーション制御サイクルに従って軸を制御中であることを示し、「実行済み」は、軸の制御が完了した状態であるこ

とを示す。

[0106] ここで、ＩＦ部３８０は、定周期処理部３６０が同時に制御可能な複数の軸について、軸ごとに、補間指令値及びフィードバック情報を格納するためのメモリ領域３８１を有している。図１１の例では、ＩＦ部３８０には、軸１～軸１２８の各々について、補間指令値及びフィードバック情報を格納可能なメモリ領域３８１を有している。

[0107] まず、ユーザ作成プログラム１１１は、ＡＰＩバッファ識別子（１）に対応するＡＰＩバッファ３４０について、実行を開始することを示すＡＰＩ関数を呼び出す。具体的には、ユーザ作成プログラム１１１は、当該ＡＰＩ関数の引数に、ＡＰＩバッファ識別子（１）を設定する。続いて、指令受付部１１２は、ＡＰＩバッファ識別子（１）のＡＰＩバッファからＡＰＩ情報を取り出して制御指令モジュール３７０に渡す処理を開始すべきことをＩＦ（Ｎ側）部１１３に通知する（Ｓ２５０）。ＩＦ（Ｎ側）部１１３は、ＡＰＩバッファ識別子（１）のＡＰＩバッファからＡＰＩ情報を取り出して制御指令モジュール３７０に渡す処理を開始すべきことを、操作チャンネルを介して指令実行処理部３５０に通知する（Ｓ２５１）。

[0108] ＩＦ（Ｎ側）部１１３から通知を受けた指令実行処理部３５０は、指定されたＡＰＩバッファ３４０に格納されているＡＰＩ情報を取得して定周期処理部３６０に渡す。具体的には、指令実行処理部３５０は、ＡＰＩバッファ識別子「１」に対応するＡＰＩバッファ３４０からＡＰＩ情報を１つ取得する。取得したＡＰＩ情報の「モジュール識別子」、「制御指令識別子」及び「指令値」には、それぞれ、「制御指令モジュールＡ」、「ＰＴＰ」及び「軸１の指令値及び軸２の指令値」が格納されている。指令実行処理部３５０は、ＡＰＩ情報の「モジュール識別子」により指定された制御指令モジュール３７０Ａに対して、実行すべき制御指令識別子（ここではＰＴＰ）を通知する（Ｓ２５３）。続いて、指令実行処理部３５０は、制御指令モジュール３７０Ａのメモリ領域３７１における軸１の指令値及び軸２の指令値を、ＡＰＩ情報から取得した軸１の指令値及び軸２の指令値で更新すると共に、軸

1の実行フラグ及び軸2の実行フラグを「実行中」に変更する(S254)

。

[0109] 続いて、制御指令モジュール370Aは、メモリ領域371に格納された軸1及び軸2の指令値に基づいて、軸1及び軸2に対する補間指令値を算出し、算出した補間指令値をIF部380に格納する(S260)。補間指令値は、制御指令の内容に応じて所定のロジックで算出される。所定のロジックとしては、例えば、従来のPTP制御における制御ロジック等を利用することができる。通信モジュール390は、IF部380に格納された軸1及び軸2の補間指令値を取得し、軸1及び軸2の通信インターフェース規格に応じた信号に変換して制御対象装置(軸1)20-1及び制御対象装置(軸2)20-2に送信する(S261)。

[0110] ステップS260及びステップS261の処理手順がモーション制御サイクル(例えば1ms間隔や0.5ms間隔等)に従って繰り返されることで、軸1及び軸2が補間指令値に従って滑らかに動作する。軸1及び軸2の状態が指令値に到達すると、制御指令モジュール370Aは、軸1及び軸2の実行フラグを「実行済み」に更新する。軸1及び軸2の実行フラグが「実行済み」に更新されたことを検出した指令実行処理部350は、次にAPIバッファ340に格納されているAPI情報を読み出し、上述したステップS253及びステップS254の処理手順を繰り返し行う。これにより、制御指令モジュール370Aのメモリ領域371における指令値が更新され、再度、ステップS260及びステップS261の処理手順が実行される。以上説明したステップS252～ステップS261の処理手順の動作が、APIバッファ340が空になるまで繰り返される。

[0111] ここで、通信モジュール390は、制御対象装置(軸1)20-1及び制御対象装置(軸2)20-2からフィードバック情報を取得し、共通信号フォーマットのフィードバック情報に変換してIF部380に格納する(S262)。続いて、制御指令モジュール370Aは、IF部380に格納された軸ごとのフィードバック情報を取得する(S263)。ステップS262

及びステップS 2 6 3の処理手順は、モーション制御サイクルに従って繰り返されるようにしてもよいし、モーション制御サイクルとは異なる周期（例えば数m s周期等）で繰り返されるようにしてもよい。

[0112] 制御指令モジュール3 7 0 Aは、取得した軸1及び軸2のフィードバック情報をメイン処理部3 2 0に通知する（S 2 6 4）。メイン処理部3 2 0は、通知された軸1及び軸2のフィードバック情報を状態チャンネルAに格納する（S 2 6 5）。ユーザ作成プログラム1 1 1が、状態チャンネルAに対応するフィードバック情報を読み出すAPI関数を呼び出すと、指令受付部1 1 2は、IF（N側）部1 1 3に対して、状態チャンネルAに格納されたフィードバック情報を読み出すべきことを通知する。IF（N側）部1 1 3が読みだしたフィードバック情報は、指令受付部1 1 2を介してユーザ作成プログラム1 1 1に伝えられる。これにより、ユーザ作成プログラム1 1 1は、各軸がどのように動作しているのかを把握することができ、例えば、軸の状態に基づいて制御指令を切り替えるといった複雑なモーション制御を実現することができる。

[0113] なお、制御指令モジュール3 7 0 Aは、ステップS 2 6 0の処理手順において、例えばモーション制御サイクル毎にフィードバック情報に含まれる軸1及び軸2の状態を参照し、軸1及び軸2が意図する動作を行っているかをモーション制御サイクル毎に確認しながら次のモーション制御サイクルにおける補間指令値を算出するようにしてもよい。

[0114] 図1 2は、制御指令モジュール3 7 0が行う処理手順の一例を示すフローチャートである。まず、制御指令モジュール3 7 0は、メモリ領域3 7 1から、実行フラグが「実行中」にセットされた軸についての指令値を取得する（S 3 0 0）。続いて、制御指令モジュール3 7 0は、指令値に基づき、1モーション制御サイクル分の指令値（つまり補間指令値）を算出し（S 3 0 1）、算出した補間指令値を、IF部3 8 0のメモリ領域3 8 1に格納する（S 3 0 2）。続いて、制御指令モジュール3 7 0は、IF部3 8 0から軸ごとのフィードバック情報を取得する（S 3 0 3）。続いて、制御指令モジ

ルール 370 は、ステップ S300 で指令値を取得した軸について、フィードバック情報が指令値に達しているか否かを判定する (S304)。指令値に達している場合は、当該軸の実行フラグを「実行済み」にセットし、処理を終了する。指令値に達していない場合は、次のモーション制御サイクルまで待機し (S305) ステップ S301 の処理手順に進む。なお、ステップ S303 の処理手順を省略し、ステップ S304 の処理手順を、軸の動作が指令値に達しているか否かを、現代制御理論を利用することでフィードバック情報を利用せず判定することに置き換えるようにしてもよい。

[0115] (エラーハンドリング処理)

[制御指示部のクラッシュ]

次に、制御指示部 110 が何らかの理由でクラッシュした際に行われる処理手順について説明する。デバイス管理部 120 は、制御指示部 110 が生存 (動作) していることを定期的に監視し、制御指示部 110 が生存している間、制御指示部 110 が生存していることを示す生存報告を A P I チャンネルに書き込む。I F (R 側) 部 321 は、各制御指示部 110 が生存していることを示す生存報告の有無を、各 A P I チャンネルを参照することで周期的に取得し、生存報告が所定の期間の間なされていない場合、生存報告が報告されなかった制御指示部 110 に対応する A P I チャンネルを破棄 (消去) する (当該 A P I チャンネルのメモリ領域を解放する)。以下、図を用いて制御指示部がクラッシュした際の処理手順を具体的に説明する。

[0116] 図 13 は、制御指示部がクラッシュした場合の処理手順の一例を示す図である。図 13 の説明では、前述した「(デバイス作成処理)」で説明した前提条件と同一の前提条件が適用されるものとする。

[0117] デバイス管理部 120-1 及びデバイス管理部 120-2 は、それぞれ、A P I チャンネル 1 及び A P I チャンネル 2 に生存報告を周期的に書き込むように構成されている (S400、S401)。生存報告は、例えばタイムスタンプであってもよい。また、周期は任意であるが、例えば、1 秒間隔や 10 秒間隔といった周期であってもよい。ここで、制御指示部 (1) 110-1

がクラッシュしたことで、デバイス管理部120-1が生存報告をAPIチャンネル1に書き込まなくなると仮定する(S400)。

[0118] その後、制御指示部(1)110-1が再起動すると、制御指示部(1)110-1は、デバイス作成要求をIF(N側)部113-1に通知する(S410)。ステップS411及びステップS412の処理手順は、それぞれ、図6のステップS105及びステップS106と同一であるため説明は省略する。なお、IF(N側)部113-1は、操作チャンネルのアドレス範囲を、各IF(N側)部113から共通に参照可能なリソース(ファイル又は共通の変数等)から取得する。

[0119] 続いて、IF(R側)部321は、共有メモリ200にAPIチャンネルに使用するメモリ領域を確保する(S413)。また、IF(R側)部321は、デバイス識別子及びAPIチャンネル識別子を決定する。ここでは、デバイス識別子を例えば「D3」に決定し、APIチャンネル識別子を例えば「APIチャンネル3」に決定したとする。IF(R側)部321は、確保したAPIチャンネル1のアドレス範囲を、アドレス管理情報に格納すると共に、デバイス管理情報に、デバイス識別子(D3)及びAPIチャンネル3が対応づけられたレコードを追加する。ステップS414～ステップS416の処理手順は、それぞれ図6のステップS110～ステップS112と同一であるため説明は省略する。

[0120] 続いて、IF(R側)部321は、APIチャンネル1にて生存報告が所定の期間書き込まれていないことを検出し、APIチャンネル1を破棄する(S417)。また、IF(R側)部321は、デバイス管理情報から、APIチャンネル1を含むレコードを削除し、更に、アドレス管理情報から、APIチャンネル1のアドレス範囲が格納されたレコードを削除する。

[0121] なお、IF(R側)部321が、生存報告が書き込まれていないことを検出するタイミングによっては、ステップS417の処理手順が、ステップS414の処理手順より前になることも想定される。この場合、IF(R側)部321は、共有メモリ200に、APIチャンネルに使用するメモリ領域を

確保する際、破棄されたA P Iチャンネル1のメモリ領域（つまり、空きメモリ領域）を、新たに生成するA P Iチャンネルのメモリ領域として確保する（再利用する）ようにしてもよい。

[0122] 本実施形態では、2以上の操作チャンネルが共有メモリ200に確保されることはなく、また、状態チャンネルが、リアルタイムOS300に実装された制御指令モジュール370の数以上に確保されることはない。一方、共有メモリ200に確保されるA P Iチャンネルの数は、起動される制御指示部110の数や制御指示部110が作成するデバイスの数に応じて変化する。そのため、利用されなくなったA P Iチャンネルが共有メモリ200に残り続けると、共有メモリ200の空きリソースが不足してモーション制御装置10自体が動作不能になる可能性がある。以上説明したエラーハンドリング処理によれば、生存報告が書き込まれなくなったA P Iチャンネルは共有メモリ200から消去されることから、制御指示部110が何度もクラッシュすることにより共有メモリ200のリソースが不足する可能性を抑止することが可能になる。

[0123] また、以上説明したエラーハンドリング処理によれば、制御指示部110がクラッシュした場合、当該制御指示部110に関係のあるA P Iチャンネルの削除や作成のみが行われ、クラッシュしていない他の制御指示部110が使用しているA P Iチャンネルや状態チャンネルについては、共有メモリ200にそのまま確保され続けることになる。すなわち、制御指示部110がクラッシュしたとしても、クラッシュしていない他の制御指示部110は、何の影響を受けることなく動作を継続することが可能になる。

[0124] [連携部のクラッシュ]

連携部310がクラッシュした場合、I F（R側）部321は、自身が記憶しているデバイス管理情報（図9（b））、状態チャンネル管理情報（図9（c））及びアドレス管理情報（図9（e））を失ってしまうことになる。これらの情報を失ってしまうと、I F（R側）部321は、各チャンネルにアクセスすることが出来なくなる。

[0125] そこで、本実施形態では、共有メモリ２００における予め定められた領域に、操作チャンネル、ＡＰＩチャンネル及び状態チャンネルのために確保されているアドレス範囲を示す情報と、各アドレス範囲がどのチャンネルのために確保されているメモリ領域なのかを示す情報とを格納しておく。連携部３１０がクラッシュして再起動した場合、連携部３１０は、当該予め定められた領域にアクセスすることで、操作チャンネル、ＡＰＩチャンネル及び状態チャンネルが確保されているアドレス範囲等を認識し、認識した情報に基づいて、デバイス管理情報、状態チャンネル管理情報及びアドレス管理情報を生成する。

[0126] これにより、連携部３１０がクラッシュした場合であっても、制御指示部１１０側にて再度デバイス生成処理等やり直すことなく、モーション制御装置１０の動作を復旧させることが可能になる。

[0127] （プロトコル変換処理）

モーション制御装置１０は、通信インターフェース規格が異なる複数の制御対象装置２０を同時に制御することができる。このような制御を実現するために、ＩＦ部３８０のメモリ領域３８１では、共通信号フォーマットに従った補間指令値及びフィードバック情報を保持する。共通信号フォーマットはどのようなフォーマットであってもよいが、複数の通信インターフェース規格との間で相互に変換（プロトコル変換）可能なように、当該複数の通信インターフェース規格の必須項目を全て含有するフォーマットであることが望ましい。

[0128] 制御指令モジュール３７０は、共通信号フォーマットに従って補間指令値を生成してＩＦ部３８０に格納する。また、通信モジュール３９０は、補間指令値をＩＦ部３８０から取得し、取得した補間指令値を、共通信号フォーマットから、通信モジュール３９０が信号の送受信を行う制御対象装置２０の通信インターフェース規格に対応する信号フォーマットに変換してから制御対象装置２０に送信する。

[0129] 同様に、通信モジュール３９０は、制御対象装置２０からフィードバック情報を受信し、受信したフィードバック情報を、制御対象装置２０の通信イ

ンターフェース規格に対応する信号フォーマットから、共通信号フォーマットにプロトコル変換してからＩＦ部３８０に格納する。

- [0130] 各通信モジュール３９０が受け持つ軸（各通信モジュール３９０が受け持つ制御対象装置２０と同義）及びＩＦ部における軸ごとのアドレス範囲は、マッピング情報に定義されている。各通信モジュール３９０は、マッピング情報に基づき、各通信モジュール３９０が受け持つべき軸（制御対象装置２０）向けに確保されたメモリ領域３８１にアクセスすることで補間指令値を取得すると共に、フィードバック情報をメモリ領域３８１に格納する。同様に、各制御指令モジュール３７０は、マッピング情報に基づき、各軸向けに確保されたメモリ領域３８１にアクセスすることで補間指令値を取得すると共に、フィードバック情報をメモリ領域３８１に格納する。
- [0131] 図１４は、マッピング情報の一例を示す図である。マッピング情報４１０には、軸（制御対象装置２０）ごとに、各軸を受け持つ通信モジュール３９０と、通信インターフェース規格で使用される、制御対象装置２０を一意に特定する識別子（スレーブ番号）とが対応づけられている。
- [0132] 図１４の例では、軸１は、ＥｔｈｅｒＣＡＴ（登録商標）に対応する通信モジュール３９０が受け持ち、ＥｔｈｅｒＣＡＴ（登録商標）におけるスレーブ１に接続された制御対象装置２０に対応していることが定義されている。同様に、軸２は、ＥｔｈｅｒＣＡＴ（登録商標）に対応する通信モジュール３９０が受け持ち、ＥｔｈｅｒＣＡＴ（登録商標）におけるスレーブ２に接続された制御対象装置２０に対応していることが定義されている。同様に、軸３は、ＲＴＥＸ（登録商標）に対応する通信モジュール３９０が受け持ち、ＲＴＥＸ（登録商標）におけるスレーブ１に接続された制御対象装置２０に対応していることが定義されている。なお、マッピング情報は、例えば、複数の通信モジュール３９０がアクセス可能なリアルタイムＯＳ３００上の所定の領域に格納されていてもよい。或いは、各通信モジュール３９０が、自身に関係のある軸に関するマッピング情報を自ら保持するようにしてもよい。例えば、ＥｔｈｅｒＣＡＴ（登録商標）に対応する通信モジュール３

90は、軸1及び軸2に対応するスレーブ番号が格納されたマッピング情報を保持し、RTEX（登録商標）に対応する通信モジュール390、軸3に対応するスレーブ番号が格納されたマッピング情報を保持するようにしてもよい。

[0133] 図15は、プロトコル変換処理の処理手順の一例を説明するための図である。通信モジュール390-1及び通信モジュール390-2は、連携部310からの指示により起動したタイミング（図6のステップS102）など、少なくとも制御対象装置20との間で通信を開始する前にマッピング情報410を読み出すことで自身が制御すべき軸の情報を取得しておく。なお、以下の処理手順において、IF部380に格納されている各軸の補間指令値及びフィードバック情報を読み出す処理、及び各軸の補間指令値及びフィードバック情報をIF部380に書き込む処理は、例えば、制御指令モジュール370及び通信モジュール390が所定の関数を呼び出すことで行われる。

[0134] まず、制御指令モジュール370は、IF部380のメモリ領域381に共通信号フォーマットに従って格納されている、軸1～軸3の補間指令値を更新する（S500）。通信モジュール390-1は、軸1の補間指令値及び軸2の補間指令値を取得し（S501、S502）、取得した補間指令値の信号フォーマットを、共通信号フォーマットからEtherCAT（登録商標）の信号フォーマットに変換する。続いて、通信モジュール390-1は、信号フォーマットの変換を終えた軸1の補間指令値及び軸2の補間指令値を、それぞれ、制御対象装置20-1及び制御対象装置20-2に送信する（S504、S505）。同様に、通信モジュール390-2は、軸3の補間指令値を取得し（S503）、取得した補間指令値の信号フォーマットを、共通信号フォーマットからRTEX（登録商標）の信号フォーマットに変換する。続いて、通信モジュール390-2は、変換した軸3の補間指令値を制御対象装置20-3に送信する（S506）。

[0135] 次に、通信モジュール390-1は、制御対象装置20-1及び制御対象

装置20-2から、軸1のフィードバック情報及び軸2のフィードバック情報を取得し（S510、S511）、取得したフィードバック情報の信号フォーマットをEtherCAT（登録商標）の信号フォーマットから共通信号フォーマットに変換する。続いて、通信モジュール390-1は、変換した軸1のフィードバック情報及び軸2のフィードバック情報を、IF部380のメモリ領域381に格納する（S513、S514）。

[0136] 同様に、通信モジュール390-2は、制御対象装置20-3から軸3のフィードバック情報を取得し（S512）、取得したフィードバック情報の信号フォーマットを、RTX（登録商標）の信号フォーマットから共通信号フォーマットに変換する。続いて、通信モジュール390-2は、変換した軸2のフィードバック情報を、IF部380のメモリ領域381に格納する（S515）。制御指令モジュール370は、IF部380のメモリ領域381に格納されている軸1～軸3のフィードバック情報を取得する（S516）。

[0137] 以上説明したプロトコル変換処理では、ステップS500～ステップS516の処理手順が、モーション制御サイクルごとに繰り返し行われる。

[0138] <まとめ>

以上、実施形態に係るモーション制御装置10について説明した。制御対象装置20に対してモーション制御を行う場合、複数の制御対象装置20を同期させながら高精度に動作させるためには、モーション制御サイクル（例えば1msごとや0.5ms）ごとに補間指令を制御対象装置20に送信し続けることが必要である。Windows（登録商標）のような非リアルタイムOS100では、OS自身が行うバックグラウンド処理等の影響により処理負荷が高くなると、実行中であるプログラムの動作速度が遅くなってしまうことから、1msや0.5msといった周期で補間指令を制御対象装置20に送信し続けることは困難である。

[0139] 一方、本実施形態では、非リアルタイムOS110側で複数のモーション制御サイクルに渡る制御指令（API情報）を発行し、リアルタイムOS3

００側でＡＰＩバッファ３４０を介して当該制御指令を取得して、モーション制御サイクルごとの補間指令を発行する構成を採用した。これにより、非リアルタイムＯＳ１１０での処理速度の変化を吸収し、非リアルタイムＯＳ１１０及びリアルタイムＯＳ３００がインストールされた汎用の情報処理装置を用いてスムーズな高精度のモーション制御を実行することを可能とした。また、複数の制御指令が発行される場合、当該複数の制御指令がＡＰＩバッファ３４０に格納されることになる。これにより、各々の制御指令による連続したモーション制御（モーション制御シーケンス）をスムーズに実行することを可能とした。

[0140] また、本実施形態では、制御指示部１１０を非リアルタイムＯＳ１１０側に配置したことで、モーション制御サイクルに沿ったリアルタイム処理を実現しつつ、ユーザが使い慣れたデザインのユーザインタフェースを提供することを可能とした。

[0141] また、本実施形態では、モーション制御を行う際に必要になるＡＰＩチャネル等のリソースを制御指示部１１０ごと（より詳細にはデバイスごと）に用意するようにした。これにより、制御指示部１１０が制御指令を発行することで制御対象装置２０を制御するという一連の動作に必要なリソースを制御指示部１１０ごとに（すなわちユーザ作成プログラム１１１ごとに）分離することができ、万が一、制御指示部１１０にクラッシュ等の異常が生じたとしても、当該異常が他の制御指示部１１０の動作に影響を与えてしまう可能性を抑制することを可能とした。

[0142] また、本実施形態では、リアルタイムＯＳ３００上の制御指令モジュール３７０からＩＦ部３８０までの処理を、共通信号フォーマットを用いて共通化し、通信モジュール３９０にて、共通信号フォーマットから制御対象装置２０に対応する通信インターフェース規格の信号フォーマットにプロトコル変換するようにした。これにより、モーション制御装置１０が行う殆どの処理を共通化することができ、通信インターフェース規格が異なる複数の制御対象装置２０を混在させた制御を、容易に実現することを可能とした。

[0143] <変形例>

以上説明した実施形態は、本発明の理解を容易にするためのものであり、本発明を限定して解釈するためのものではない。実施形態で説明したフローチャート、シーケンス、実施形態が備える各要素並びにその配置、材料、条件、形状及びサイズ等は、例示したものに限定されるわけではなく適宜変更することができる。また、異なる実施形態で示した構成同士を部分的に置換し又は組み合わせることが可能である。

[0144] 例えば、APIチャネルはデバイスごとに生成されることから、デバイスが特定されればAPIチャネルも一意に特定できることになる。そのため、本実施形態において「APIチャネル識別子」を「デバイス識別子」に置き換えることとしてもよい。

請求の範囲

- [請求項1] 非リアルタイムOS及びリアルタイムOSがインストールされ、複数の制御対象装置をモーション制御するためのコンピュータで実行されるモーション制御プログラムであって、
- 前記コンピュータを、
- 前記非リアルタイムOS上で動作する受付部と、
- 前記リアルタイムOS上で動作する制御部と、
- 前記リアルタイムOS上で動作する複数の通信モジュール部と、
- して機能させ、
- 前記受付部は、前記複数の制御対象装置を制御するユーザ作成プログラムから、前記複数の制御対象装置の各々が複数のモーション制御サイクルに渡って行うべき動作を示す制御指令を受け付け、受け付けた制御指令の内容を示す制御指令情報を前記制御部に通知し、
- 前記制御部は、前記受付部から通知された前記制御指令情報に基づいて、前記複数の制御対象装置の各々に対して前記モーション制御サイクルごとに実行させるべき動作を示す、前記制御対象装置ごとの補間指令を前記モーション制御サイクルごとに繰り返し生成し、生成した前記補間指令を、前記制御対象装置ごとに確保されるメモリ領域に格納し、
- 前記通信モジュール部は、前記制御対象装置ごとに確保された前記メモリ領域から前記補間指令を取得し、取得した前記補間指令を、前記制御部が認識可能な所定の信号フォーマットから、前記複数の制御対象装置の各々が認識可能な通信インターフェース規格の信号フォーマットに変換して送信する、
- モーション制御プログラム。
- [請求項2] 前記通信モジュール部は、前記複数の制御対象装置の各々から取得される制御対象装置の状態を示すフィードバック情報を、前記複数の制御対象装置の各々が認識可能な通信インターフェース規格の信号フ

フォーマットから前記所定の信号フォーマットに変換して前記メモリ領域に格納し、

前記制御部は、前記メモリ領域から、前記所定の信号フォーマットに変換された前記フィードバック情報を取得する、

請求項 1 に記載のモーション制御プログラム。

[請求項3]

前記複数の通信モジュール部の各々は、前記複数の通信モジュール部の各々が受け持つべき前記制御対象装置と、前記受け持つべき前記制御対象装置に確保された前記メモリ領域との対応関係を定義するマッピング情報に基づいて、前記メモリ領域にアクセスすることで前記補間指令を取得すると共に、前記フィードバック情報を前記メモリ領域に格納する、

請求項 2 に記載のモーション制御プログラム。

[請求項4]

前記マッピング情報は、前記複数の通信モジュール部がアクセス可能な前記リアルタイムOS上の領域に格納される、

請求項 3 に記載のモーション制御プログラム。

[請求項5]

非リアルタイムOS及びリアルタイムOSがインストールされ、複数の制御対象装置をモーション制御するためのモーション制御装置が行うモーション制御方法であって、

前記モーション制御装置は、

前記非リアルタイムOS上で動作する受付部と、

前記リアルタイムOS上で動作する制御部と、

前記リアルタイムOS上で動作する複数の通信モジュール部と、

を有し、

前記受付部が、前記複数の制御対象装置を制御するユーザ作成プログラムから、前記複数の制御対象装置の各々が複数のモーション制御サイクルに渡って行うべき動作を示す制御指令を受け付け、受け付けた制御指令の内容を示す制御指令情報を前記制御部に通知するステップと、

前記制御部が、前記受付部から通知された前記制御指令情報に基づいて、前記複数の制御対象装置の各々に対して前記モーション制御サイクルごとに実行させるべき動作を示す、前記制御対象装置ごとの補間指令を前記モーション制御サイクルごとに繰り返し生成し、生成した前記補間指令を、前記制御対象装置ごとに確保されるメモリ領域に格納するステップと、

前記通信モジュール部が、前記制御対象装置ごとに確保された前記メモリ領域から前記補間指令を取得し、取得した前記補間指令を、前記制御部が認識可能な所定の信号フォーマットから、前記複数の制御対象装置の各々が認識可能な通信インターフェース規格の信号フォーマットに変換して送信するステップと、

を含むモーション制御方法。

[請求項6]

非リアルタイムOS及びリアルタイムOSがインストールされ、複数の制御対象装置をモーション制御するためのモーション制御装置であって、

前記非リアルタイムOS上で動作する受付部と、

前記リアルタイムOS上で動作する制御部と、

前記リアルタイムOS上で動作する複数の通信モジュール部と、

を有し、

前記受付部は、前記複数の制御対象装置を制御するユーザ作成プログラムから、前記複数の制御対象装置の各々が複数のモーション制御サイクルに渡って行うべき動作を示す制御指令を受け付け、受け付けた制御指令の内容を示す制御指令情報を前記制御部に通知し、

前記制御部は、前記受付部から通知された前記制御指令情報に基づいて、前記複数の制御対象装置の各々に対して前記モーション制御サイクルごとに実行させるべき動作を示す、前記制御対象装置ごとの補間指令を前記モーション制御サイクルごとに繰り返し生成し、生成した前記補間指令を、前記制御対象装置ごとに確保されるメモリ領域に

格納し、

前記通信モジュール部は、前記制御対象装置ごとに確保された前記メモリ領域から前記補間指令を取得し、取得した前記補間指令を、前記制御部が認識可能な所定の信号フォーマットから、前記複数の制御対象装置の各々が認識可能な通信インターフェース規格の信号フォーマットに変換して送信する、

モーション制御装置。