

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、ベルトコンベアの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、例えばユーザによって作成された、様々なモーション動作を指示するプログラムを複数実行し、複数の制御対象装置を並行に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パーソナルコンピュータを利用して、様々なモーション動作を指示するプログラムを複数搭載して複数の制御対象装置を並行に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を用いてモーション制御を実現する際に、様々なモーション動作を指示するプログラムを複数実行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、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上で動作する複数の受付部であって、複数の受付部の各々が制御対象装置を制御する複数のユーザ作成プログラムの各々に対応づけられる、複数の受付部と、リアルタイムOS上で動作する

チャンネル管理部と、リアルタイムOS上で動作する定周期処理部として機能させ、チャンネル管理部は、複数の受付部に共通の操作用チャンネルを、非リアルタイムOS及びリアルタイムOSから参照可能な共有メモリ上に作成し、受付部は、該受付部に対応づけられるユーザ作成プログラムから、制御対象装置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を受け付けた場合、操作用チャンネルを介して、制御指令用チャンネルの生成を生成部に指示し、チャンネル管理部は、受付部から指示を受けた場合に、準備指示を行ったユーザ作成プログラムに対応づけられる制御指令用チャンネルを共有メモリ上に生成し、受付部は、ユーザ作成プログラムから、制御対象装置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、制御指令用チャンネルに格納し、定周期処理部は、制御指令用チャンネルから取得した制御指令情報に基づいて、制御対象装置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、制御対象装置をモーション制御する。

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を用いてモーション制御を実現する際に、様々なモーション動作を指示するアプリケーションを複数実行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、W i n d o w s（登録商標）、m a c O S（登録商標）等が挙げられる。また、リアルタイムOSは、例えば、R T X（Real Time Extension）、R T H（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、P C（パーソナルコンピュータ）、ノート型P C、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G U I（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G U I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OS100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、FA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OS100では、複数の制御指示部110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（ID等）と指令値とを含む。制御指令の具体例としては、例えば、PTP（Point to Point）動作（指定した地点に移動させる動作）、JOG動作（指定した速度でモータを動作）、直線補間によるPTP動作、円弧補間によるPTP動作、加速時間を指定するJOG動作、行動時間を指定するJOG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PTP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PTP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110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310に通知される。

[0021] 制御指示部110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API（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、API関数の引数（argument）に値を設定することで行われる。API関数は、制御指令ごとに1つ用意されており、例えばPTP動作に対しては「startpos()」というAPI関数が用意されている。本実施形態では、制御指示部110で呼び出されたAPI関数及び当該API関数に設定された引数を示す情報を、「API情報」と言う。な

お、本実施形態では、API情報を、「制御指令情報」と呼んでもよい。

[0022] 連携部310は、制御指示部110から通知されたAPI情報を一時的に蓄積するためのFIFOキュー（以下、「APIバッファ」と言う。）を備えている。連携部310は、制御指示部110から受け取った複数のAPI情報を一旦APIバッファ340に蓄積する。また、連携部310は、APIバッファ340からAPI情報を1つずつ取り出して定周期処理部360に渡す。つまり、連携部310は、キューを利用して定周期処理部360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API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、API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からも利用可能となる。

[0030] ここで、API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、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3（a）に、API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（PTP）、JOG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、API関数を一意に示す識別子であってもよい。

[0031] 「APIバッファ識別子」には、APIバッファ340を一意に特定するAPI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PTP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20から取得されたフィードバック情報が、制御対象装置20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10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10は、CPU（Central Processing Unit）11、メモリ12、HDD（HardDisk）やSSD（Solid State Drive）等の記憶装置13、有線又は無線通信を行う通信IF（Interface）14、入力操作を受け付ける入力デバイス15、及び情報の出力を行う出力デバイス16を有する。入力デバイス15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16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10の機能構成例を示す図である。共有メモリ200は、非リアルタイムOS100側の各機能部とリアルタイムOS3

00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200には、操作チャンネル、API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110及び連携部310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りすることでデータの送受信を行う。

[0035] (非リアルタイムOS)

非リアルタイムOS100上では、1以上の制御指示部110、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、操作チャンネル、APIチャンネル及び状態チャンネルの状態管理（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、操作チャンネル、APIチャンネル及び状態チャンネルの作成（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チャンネル及び状態チャンネルの破棄（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、「チャンネル管理部」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「チャンネル管理部」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OS 300の指示によりIF（R側）部321自身が起動した場合に、操作チャンネルを共有メモ

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ＯＳ１１０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チャネル、状態チャ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、

A P I 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、I F 部 380 に渡す。なお、制御指令モジュール 370 は、A P I 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 A P I 関数に対応する A P I 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 A P I 関数に対応する A P I 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 P T P 動作や J O G 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 A P I 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 O S 110 上に用意されている。

[0057] I F 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ＣＰＵ１１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 OS 300 の指示により起動した連携部 310 の I/F (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 I/F (N側) 部 113-1 に通知する (S104)。このとき、I/F (R側) 部 321 は、共有メモリ 200 において操作チャネルのために確保されたアドレス範囲も I/F (N側) 部 113-1 に通知する。I/F (R側) 部 321 は、操作チャ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 113-1 は、操作チャネルのアドレス範囲を、各 I/F (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) に格納する。なお、I/F (N側) 部 113-1 は、操作チャ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。操作チャネルの作成が完了したことの通知を受けた I/F (N側) 部 113-1 は、他の制御指示部 110 の I/F (N側) 部 113 との間で操作チャネルの利用について競合が生じないようにするため、操作チャネルの利用権限を取得する (S105)。

[0071] 利用権限が取得できた場合、I/F (N側) 部 113-1 は、デバイス作成要求を、操作チャネルを介して I/F (R側) 部 321 に通知する (S106)。デバイス作成要求には、ステップ S100 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 321 は、デバイス識別子を例えば「D1」に決定する。また、I/F (R側) 部 321 は、APIチャネルに使用するメモリ領域を確保すると共に、APIチャネル識別子を例えば「APIチャネル1」に決定する (S107)。また、I/F (R側) 部 321 は、決定したデバイス識別子及び APIチャ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、IF（R側）部321は、共有メモリ200に、制御指令モジュール370A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（S108）。IF（R側）部321は、作成した状態チャンネルAと制御指令モジュール370A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、IF（R側）部321は、API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、操作チャン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、操作チャンネルを介して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。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、ＩＦ（Ｒ側）部３２１は、決定したデバイス識別子及びＡＰＩチャネル識別子を対応づけて、デバイス管理情報（図９（ｂ））に追加する。

[0082] 続いて、ＩＦ（Ｒ側）部３２１は、共有メモリ２００に、制御指令モジュール３７０Ａに対応する状態チャネルＡ、制御指令モジュール３７０Ｂに対応する状態チャネルＢ及び制御指令モジュール３７０Ｃに対応する状態チャネル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３７０Ａに対応する状態チャネルＡは既に作成済み（図６のステップＳ１０８）であることから、ＩＦ（Ｒ側）部３２１は、状態チャネルＢ及び状態チャネルＣに使用するメモリ領域の確保を行う（Ｓ１５４、Ｓ１５５）。

[0083] 続いて、ＩＦ（Ｒ側）部３２１は、作成した状態チャネルＢ及び状態チャネルＣが、それぞれ制御指令モジュール３７０Ｂ及び制御指令モジュール３７０Ｃに対応づけられることを、ＩＦ（Ｒ側）部３２１が保持する状態チャネル管理情報（図９（ｃ））に追加する。また、ＩＦ（Ｒ側）部３２１は、ＡＰＩチャネル２、状態チャネルＢ及び状態チャネルＣに確保したメモリ領域のアドレス範囲を、ＩＦ（Ｒ側）部３２１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、ＩＦ（Ｒ側）部３２１は、デバイスの作成が完了したこと（ＡＰＩチャネル２、状態チャネルＢ及び状態チャネルＣの作成が完了したこと）をＩＦ（Ｎ側）部１１３－２に通知する（Ｓ１５７）。当該通知には、ＡＰＩチャネル２、状態チャネルＡ、状態チャネルＢ及び状態チャネルＣに確保されたメモリ領域のアドレス範囲、ＡＰＩチャネル識別子（ＡＰＩチャネル

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 1 1 2-2 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 1 1 1-2 に通知する (S 1 5 9)。当該通知には、デバイス識別子 (D 2) 及び A P I チャンネル識別子 (A P I チャンネル 2) が含まれる。ユーザ作成プログラム 1 1 1-2 は、通知されたデバイス識別子 (D 2) 及び A P I チャンネル識別子 (A P I 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 I F (R側) 部 3 2 1 側は、A P I 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 S 1 0 0 ~ ステップ S 1 1 2 及び図 7 のステップ S 1 5 0 ~ ステップ S 1 5 9 の処理手順では A P I 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、A P I 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 1 1 1 は、モジュール識別子を含まないデバイス作成要求を I F (N側) 部 1 1 3 に通知し、I F (N側) 部 1 1 3 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 I F (R側) 部 3 2 1 に通知する。当該デバイス作成要求を受けた I F (R側) 部 3 2 1 は、A P I チャンネルを作成し、デバイスの作成が完了したことを I F (N側) 部 1 1 3 に通知する。I F (N側) 部 1 1 3 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 1 1 1 に通知する。

[0089] その後、ユーザ作成プログラム 1 1 1 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」とであると仮定する。

[0094] IF(N側)部113は、操作チャネルを介して、APIバッファ340の作成をメイン処理部320に要求する(S201)。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320は、APIバッファ識別子「1」に対応するAPIバッファ340を作成する(メモリ領域を確保する)(S202)。

[0095] 続いて、ユーザ作成プログラム111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112は、当該API情報の記録を開始することをIF(N側)部113に通知する(S203)。IF(N側)部113は、当該API情報の記録を開始することをデバイス管理部120に通知する(S204)。デバイス管理部120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(S205)。

[0096] 記録中フラグは、ユーザ作成プログラム111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340にキューイングしてから定周期処理部360に渡すのか、又は、ユーザ作成プログラム111が呼び出したAPI関数に対応するAPI情報を、APIバッファ340にキューイングせずに直接、定周期処理部360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330によりAPIバッファ340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チャンネルにAPI情報が格納された場合、APIチャンネルに格納されたAPI情報は、API実行要求処理部330により取得されて定周期処理部360に渡され、定周期処理部360で実行される。

[0097] 続いて、ユーザ作成プログラム111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D1）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API関数を呼び出す。制御指示部110は、APIバッファ340へのAPI情報の記録を終了することをIF(N側)部113に通知し(S210)、IF(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、API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] API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API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API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API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API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、APIバッファ340にキューイングされたAPI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S 2 5 2～ステップS 2 6 1の処理手順の動作が、A P Iバッファ3 4 0が空になるまで繰り返される。

[0111] ここで、通信モジュール3 9 0は、制御対象装置（軸1）2 0－1及び制御対象装置（軸2）2 0－2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I F部3 8 0に格納する（S 2 6 2）。続いて、制御指令モジュール3 7 0 Aは、I F部3 8 0に格納された軸ごとのフィードバック情報を取得する（S 2 6 3）。ステップS 2 6 2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A P I関数を呼び出すと、指令受付部1 1 2は、I F（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。I F（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370は、メモリ領域371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(S300)。続いて、制御指令モジュール370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(S301)、算出した補間指令値を、IF部380のメモリ領域381に格納する(S302)。続いて、制御指令モジュール370は、IF部380から軸ごとのフィードバック情報を取得する(S303)。続いて、制御指令モジュール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APIチャンネルに書き込む。IF(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API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APIチャンネルを破棄(消去)する(当該API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、APIチャンネル1及びAPI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(S

417)。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API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API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API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API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API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API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API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、ＩＦ（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、ＩＦ（Ｒ側）部３２１は、各チャ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、操作チャネル、ＡＰＩチャネル及び状態チャ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、操作チャネル、ＡＰＩチャネル及び状態チャ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を作り直すことなく、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 I F 部 3 8 0 に格納する。また、通信モジュール 3 9 0 は、補間指令値を I F 部 3 8 0 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 3 9 0 が信号の送受信を行う制御対象装置 2 0 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 2 0 に送信する。

[0129] 同様に、通信モジュール 3 9 0 は、制御対象装置 2 0 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 2 0 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 I F 部 3 8 0 に格納する。

[0130] 各通信モジュール 3 9 0 が受け持つ軸（各通信モジュール 3 9 0 が受け持つ制御対象装置 2 0 と同義）及び I F 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 3 9 0 は、マッピング情報に基づき、各通信モジュール 3 9 0 が受け持つべき軸（制御対象装置 2 0）向けに確保されたメモリ領域 3 8 1 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 3 8 1 に格納する。同様に、各制御指令モジュール 3 7 0 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 3 8 1 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 3 8 1 に格納する。

[0131] 図 1 4 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 4 1 0 には、軸（制御対象装置 2 0）ごとに、各軸を受け持つ通信モジュール 3 9 0 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 2 0 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 1 4 の例では、軸 1 は、E t h e r C A T（登録商標）に対応する通信モジュール 3 9 0 が受け持ち、E t h e r C A T（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 2 0 に対応していることが定義されている。同様に、軸 2 は、E t h e r C A T（登録商標）に対応する通信モジュール 3 9 0 が受け持ち、E t h e r C A T（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 2 0 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390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、RTEX (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OS 100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ふことから、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OS 110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OS 300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OS 110での処理速度の変化を吸収し、非リアルタイムOS 110及びリアルタイムOS 300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OS 110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OS 300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、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上で動作する複数の受付部であって、複数の受付部の各々が前記制御対象装置を制御する複数のユーザ作成プログラムの各々と対応づけられる、複数の受付部と、
- 前記リアルタイムOS上で動作するチャンネル管理部と、
- 前記リアルタイムOS上で動作する定周期処理部と
- して機能させ、
- 前記チャンネル管理部は、前記複数の受付部に共通の操作用チャンネルを、前記非リアルタイムOS及び前記リアルタイムOSから参照可能な共有メモリ上に作成し、
- 前記受付部は、該受付部に対応づけられる前記ユーザ作成プログラムから、前記制御対象装置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を受け付けた場合、前記操作用チャンネルを介して、制御指令用チャンネルの生成を前記チャンネル管理部に指示し、
- 前記チャンネル管理部は、前記受付部から前記指示を受けた場合に、前記準備指示を行った前記ユーザ作成プログラムに対応づけられる前記制御指令用チャンネルを前記共有メモリ上に生成し、
- 前記受付部は、前記ユーザ作成プログラムから、前記制御対象装置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、前記制御指令用チャンネルに格納し、
- 前記定周期処理部は、前記制御指令用チャンネルから取得した前記制御指令情報に基づいて、前記制御対象装置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補間指令を前記モーション

ン制御サイクルごと送信することで、前記制御対象装置をモーション制御する、

モーション制御プログラム。

[請求項2]

前記受付部は、前記制御対象装置の状態を示すフィードバック情報を格納するための状態チャネルを生成すべきとの指示を、前記操作用チャネルを介して前記チャネル管理部に通知し、

前記チャネル管理部は、前記操作用チャネルを介して、前記状態チャネルを生成すべきとの指示を受けた場合で、かつ前記共有メモリ上に前記状態チャネルが生成されていない場合、前記状態チャネルを前記共有メモリ上に生成する、

請求項1に記載のモーション制御プログラム。

[請求項3]

前記複数の受付部の各々は、前記ユーザ作成プログラムから、前記フィードバック情報を取得したいとの指示を受けた場合に、前記状態チャネルから前記フィードバック情報を取得して前記ユーザ作成プログラムに渡す、

請求項2に記載のモーション制御プログラム。

[請求項4]

前記受付部は、

前記操作用チャネルの利用権限を取得し、該利用権限を取得できた場合に、前記操作用チャネルを介して、前記制御指令用チャネルを生成すべきとの指示を前記チャネル管理部に通知し、

前記チャネル管理部から、前記制御指令用チャネルの生成が完了したとの通知を受けた場合、前記操作用チャネルの利用権限を解放する、

請求項1乃至3のいずれか一項に記載のモーション制御プログラム。

[請求項5]

前記チャネル管理部は、

前記ユーザ作成プログラムの各々が生存していることを示す生存報告の有無を周期的に取得し、

前記生存報告を所定の期間取得できない場合、前記生存報告を取得できない前記ユーザ作成プログラムに対応づけられる前記制御指令用チャンネルを前記共有メモリから削除する、

請求項 1 乃至 4 のいずれか一項に記載のモーション制御プログラム。

[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、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において、前記制御対象装置を制御する複数のユーザ作成プログラムの各々に対応づけられる、複数の受付部と、

前記リアルタイムOS上で動作するチャンネル管理部と、

前記リアルタイムOS上で動作する定周期処理部と、

を有し、

前記チャンネル管理部が、前記複数の受付部に共通の操作用チャンネルを、前記非リアルタイムOS及び前記リアルタイムOSから参照可能な共有メモリ上に作成するステップと、

前記受付部が、該受付部に対応づけられる前記ユーザ作成プログラムから、前記制御対象装置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を受け付けた場合、前記操作用チャンネルを介して、制御指令用チャンネルの生成を前記チャンネル管理部に指示するステップと、

前記チャンネル管理部が、前記受付部から前記指示を受けた場合に、前記準備指示を行った前記ユーザ作成プログラムに対応づけられる前記制御指令用チャンネルを前記共有メモリ上に生成するステップと、

前記受付部が、前記ユーザ作成プログラムから、前記制御対象装置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、前記制御指令用チャンネルに格納するステップと、

前記定周期処理部が、前記制御指令用チャンネルから取得した前記制御指令情報に基づいて、前記制御対象装置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補間指令を前記モーション制御サイクルごとに送信することで、前記制御対象装置をモーション制御するステップと、

を含むモーション制御方法。

[請求項7]

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、制御対象装置をモーション制御するモーション制御装置であって、

前記非リアルタイムOS上で動作する複数の受付部であって、複数の受付部の各々が前記制御対象装置を制御する複数のユーザ作成プログラムの各々と対応づけられる、複数の受付部と、

前記リアルタイムOS上で動作するチャンネル管理部と、

前記リアルタイムOS上で動作する定周期処理部と

を有し、

前記チャンネル管理部は、前記複数の受付部に共通の操作用チャンネルを、前記非リアルタイムOS及び前記リアルタイムOSから参照可能な共有メモリ上に作成し、

前記受付部は、該受付部に対応づけられる前記ユーザ作成プログラムから、前記制御対象装置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を受け付けた場合、前記操作用チャンネルを介して、制御指令用チャンネルの生成を前記チャンネル管理部に指示し、

前記チャンネル管理部は、前記受付部から前記指示を受けた場合に、前記準備指示を行った前記ユーザ作成プログラムに対応づけられる前記制御指令用チャンネルを前記共有メモリ上に生成し、

前記受付部は、前記ユーザ作成プログラムから、前記制御対象装置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、前記制御指令用チャンネルに格納し、

前記定周期処理部は、前記制御指令用チャンネルから取得した前記制御指令情報に基づいて、前記制御対象装置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補間指令を前記モーション制御サイクルごとに送信することで、前記制御対象装置をモーション制御する、

モーション制御装置。