(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2004-334537
(P2004-334537A)
(43)公開日　平成16年11月25日(2004.11.25)

(51)Int.Cl.[7]	F I		テーマコード（参考）
G06F 9/46	G06F	9/46　340B	5B098

審査請求 未請求　請求項の数 10　ＯＬ　（全 19 頁）

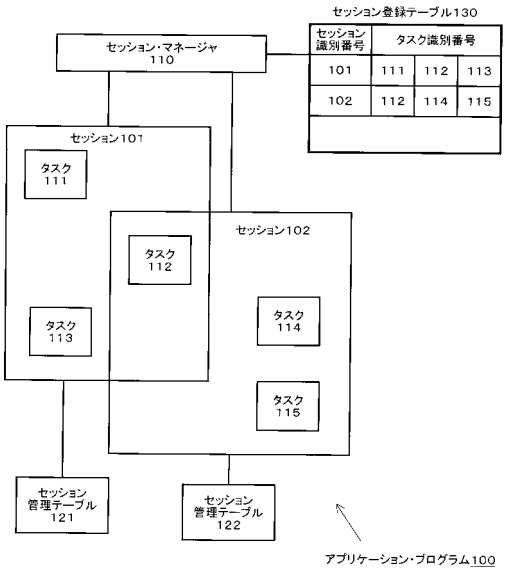
(21)出願番号　特願2003-129545(P2003-129545)
(22)出願日　平成15年5月7日(2003.5.7)

(71)出願人　000002185
ソニー株式会社
東京都品川区北品川６丁目７番３５号
(74)代理人　100093241
弁理士　宮田　正昭
(74)代理人　100101801
弁理士　山田　英治
(74)代理人　100086531
弁理士　澤田　俊夫
(72)発明者　下村　宗弘
東京都品川区北品川６丁目７番３５号　ソニー株式会社内

Ｆターム(参考)　5B098 GA04　GC01　GC16

(54)【発明の名称】プログラム処理システム及びプログラム処理方法、並びにコンピュータ・プログラム

(57)【要約】
【課題】同一のソースコードからなるタスクが複数存在する場合にメッセージ送信先を特定して好適にメッセージ通信を行なう。
【解決手段】セッションの起動時に、セッションを各機能毎にタスクを分割したときの各ブロックにブロックＩＤを割り振るとともに、ブロックＩＤとタスクＩＤの早見表であるタスク・ルックアップ・テーブルを作成する。メッセージ通信を行なう際、送信元タスクは送信先をブロックＩＤで指定し、タスク・ルックアップ・テーブルを参照してタスクＩＤを取得する。この後、送信先のタスクＩＤを指定し、ＯＳのメッセージ通信機能を用いてメッセージの送受信を行なう。
【選択図】　図２



【特許請求の範囲】
【請求項１】
複数のタスクを擬似的に並列に実行させるマルチタスク環境とタスク間通信機能を提供するマルチタスク・オペレーティング・システムが提供する実行環境下で複数のタスクからなるプログラムを実行するプログラム処理システムであって、
プログラムが提供する各機能毎に該機能を実現するために必要なタスクとその実行順序をセッションとして定義するセッション定義手段と、
セッションを実行するためのリソースと実行状態を管理するセッション管理手段とを備え、
前記セッション管理手段は、セッション起動時に当該セッションを構成する各タスクのインスタンスを生成するとともに、当該セッション及び当該セッションを構成する各タスクのインスタンスにセッションの識別情報及びインスタンスの識別情報を割り振り、セッション毎のタスク・ルックアップ・テーブル上で前記の各識別情報を管理する、
ことを特徴とするプログラム処理システム。
【請求項２】
タスク間で通信を行なう際に、タスクが含まれるセッションのタスク・ルックアップ・テーブルを参照して送信先タスクのインスタンス識別情報に基づいてタスクを特定して、メッセージ送受信を行なうメッセージ通信手段をさらに備える、
ことを特徴とする請求項１に記載のプログラム処理システム。
【請求項３】
前記メッセージ通信手段は、タスク・ルックアップ・テーブルから得られた送信先タスクのインスタンス識別情報に基づいて、前記オペレーティング・システムが提供するタスク間通信機能を用いてメッセージ送受信を行なう、
ことを特徴とする請求項２に記載のプログラム処理システム。
【請求項４】
前記メッセージ通信手段は各タスク内に組み込まれる、
ことを特徴とする請求項２に記載のプログラム処理システム。
【請求項５】
複数のタスクを擬似的に並列に実行させるマルチタスク環境とタスク間通信機能を提供するマルチタスク・オペレーティング・システムが提供する実行環境下で複数のタスクからなるプログラムを実行するプログラム処理方法であって、
プログラムが提供する各機能毎に該機能を実現するために必要なタスクとその実行順序をセッションとして定義するセッション定義ステップと、
セッションを実行するためのリソースと実行状態を管理するセッション管理ステップとを備え、
前記セッション管理ステップでは、セッション起動時に当該セッションを構成する各タスクのインスタンスを生成するとともに、当該セッション及び当該セッションを構成する各タスクのインスタンスにセッションの識別情報及びインスタンスの識別情報を割り振り、セッション毎のタスク・ルックアップ・テーブル上で前記の各識別情報を管理する、
ことを特徴とするプログラム処理方法。
【請求項６】
タスク間で通信を行なう際に、タスクが含まれるセッションのタスク・ルックアップ・テーブルを参照して送信先タスクのインスタンス識別情報に基づいてタスクを特定して、メッセージ送受信を行なうメッセージ通信ステップをさらに備える、
ことを特徴とする請求項５に記載のプログラム処理方法。
【請求項７】
前記メッセージ通信ステップでは、タスク・ルックアップ・テーブルから得られた送信先タスクのインスタンス識別情報に基づいて、前記オペレーティング・システムが提供するタスク間通信機能を用いてメッセージ送受信を行なう、
ことを特徴とする請求項６に記載のプログラム処理方法。

【請求項８】
複数のタスクを擬似的に並列に実行させるマルチタスク環境とタスク間通信機能を提供するマルチタスク・オペレーティング・システムが提供する実行環境下下で実行することにより複数の機能を実現するようにコンピュータ可読形式で記述されたコンピュータ・プログラムであって、
プログラムが提供する各機能毎に該機能を実現するために必要なタスクとその実行順序をセッションとして定義するセッション定義手段と、
セッションを実行するためのリソースと実行状態を管理するセッション管理手段と、
タスク間で通信を行なうメッセージ通信手段とを備え、
前記セッション管理手段は、セッション起動時に当該セッションを構成する各タスクのインスタンスを生成するとともに、当該セッション及び当該セッションを構成する各タスクのインスタンスにセッションの識別情報及びインスタンスの識別情報を割り振り、セッション毎のタスク・ルックアップ・テーブル上で前記の各識別情報を管理し、
前記メッセージ通信手段は、タスク・ルックアップ・テーブルから得られた送信先タスクのインスタンス識別情報に基づいて、前記オペレーティング・システムが提供するタスク間通信機能を用いてメッセージ送受信を行なう、
ことを特徴とするコンピュータ・プログラム。
【請求項９】
前記メッセージ通信手段は各タスク内に組み込まれる、
ことを特徴とする請求項８に記載のコンピュータ・プログラム。
【請求項１０】
複数のタスクを擬似的に並列に実行させるマルチタスク環境とタスク間通信機能を提供するマルチタスク・オペレーティング・システムが提供する実行環境下で複数のタスクからなるプログラムの処理をコンピュータ・システム上で実行するようにコンピュータ可読形式で記述されたコンピュータ・プログラムであって、
プログラムが提供する各機能毎に該機能を実現するために必要なタスクとその実行順序をセッションとして定義するセッション定義ステップと、
セッションを実行するためのリソースと実行状態を管理するセッション管理ステップと、
前記セッション管理ステップでは、セッション起動時に当該セッションを構成する各タスクのインスタンスを生成するとともに、当該セッション及び当該セッションを構成する各タスクのインスタンスにセッションの識別情報及びインスタンスの識別情報を割り振り、セッション毎のタスク・ルックアップ・テーブル上で前記の各識別情報を管理し、
前記メッセージ通信ステップでは、タスク・ルックアップ・テーブルから得られた送信先タスクのインスタンス識別情報に基づいて、前記オペレーティング・システムが提供するタスク間通信機能を用いてメッセージ送受信を行なう、
ことを特徴とするコンピュータ・プログラム。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】
本発明は、複数のタスクを擬似的に同時並列的に動作させるマルチタスク環境下でアプリケーションを実行するプログラム処理システム及びプログラム処理方法、並びにコンピュータ・プログラムに係り、特に、アプリケーションが複数の機能を同時並行して動作する際に複数の機能で同一のタスクを共有するプログラム処理システム及びプログラム処理方法、並びにコンピュータ・プログラムに関する。
【０００２】
さらに詳しくは、本発明は、複数の機能で共有されるタスクのリソースと実行状態を管理するプログラム処理システム及びプログラム処理方法、並びにコンピュータ・プログラムに係り、特に、同一のソースコードからなるタスクが複数存在する場合にメッセージ送信先を特定してメッセージ通信を行なうプログラム処理システム及びプログラム処理方法、並びにコンピュータ・プログラムに関する。

【０００３】
【従来の技術】
昨今のＬＳＩ（Ｌａｒｇｅ Ｓｃａｌｅ Ｉｎｔｅｇｒａｔｉｏｎ）技術における革新的な進歩とも相俟って、さまざまなタイプの情報処理機器や情報通信機器が開発・市販され、日常生活に深く浸透するに至っている。この種の機器では、オペレーティング・システムが提供する実行環境下で、ＣＰＵ（Ｃｅｎｔｒａｌ Ｐｒｏｃｅｓｓｉｎｇ Ｕｎｉｔ）やその他のプロセッサが所定のプログラム・コードを実行することによりさまざまな処理サービスを提供するようになっている。
【０００４】
一般に、オペレーティング・システムは、複数のタスクを時分割で実行することによって、プロセッサの数よりも多くのタスクをあたかも同時に実行しているかのように見せるマルチタスク機能を備えている。マルチタスク・オペレーティング・システムを使用することにより、アプリケーションを機能毎にタスクとして作成し、複数のタスクを擬似的に同時並列的に動作させることができる。
【０００５】
しかしながら、オペレーティング・システムには、タスク単位の管理機能はあるものの、複数のタスクによってある特定の機能が実現されるような場合、その機能に係る一連のタスクを一元的に管理する機構は提供していない。すなわち、マルチタスクにおいて、個々のタスクの実行を制御するのみで、タスク間の関連性までは考慮しない。このため、各タスクが自分自身でタスク間の関連性を管理しなければならず、プログラムの組み易さやメンテナンス性が低下してしまう。
【０００６】
また、同時並行して動作する複数の機能で同一のタスクを共有したい場合があるが、共有されるタスクを管理するような仕組みはオペレーティング・システムでは提供されない。
【０００７】
また、このようなマルチタスク環境下では、同一のソースコードを持つ複数のタスクが複数存在するという状況が想定されるが、このような場合にタスクを動的に特定することはできない。
【０００８】
一般的なオブジェクト間通信では、オペレーティング・システムのオブジェクト識別機能を用いてメッセージ送受信先の特定が行なわれる（例えば、特許文献１、特許文献２を参照のこと）。しかしながら、複数のセッションにおいて同じタスクが起動しているような状況においては、単にタスクを識別するタスク識別子を用いるだけでは、どのセッションのタスクなのかを特定することはできない。
【０００９】
【特許文献１】
特開平０８－５５０３７号公報
【特許文献２】
特開平０８－１０６４４１号公報
【００１０】
【発明が解決しようとする課題】
本発明の目的は、アプリケーションが複数の機能を同時並行して動作する際に複数の機能で同一のタスクを好適に共有することができる、優れたプログラム処理システム及びプログラム処理方法、並びにコンピュータ・プログラムを提供することにある。
【００１１】
本発明のさらなる目的は、複数の機能で共有されるタスクのリソースと実行状態を好適に管理することができる、優れたプログラム処理システム及びプログラム処理方法、並びにコンピュータ・プログラムを提供することにある。
【００１２】
本発明のさらなる目的は、同一のソースコードからなるタスクが複数存在する場合にメッ

セージ送信先を特定して好適にメッセージ通信を行なうことができる、優れたプログラム処理システム及びプログラム処理方法、並びにコンピュータ・プログラムを提供することにある。

【００１３】

【課題を解決するための手段及び作用】

本発明は、上記課題を参酌してなされたものであり、その第１の側面は、複数のタスクを擬似的に並列に実行させるマルチタスク環境とタスク間通信機能を提供するマルチタスク・オペレーティング・システムが提供する実行環境下で複数のタスクからなるプログラムを実行するプログラム処理システムであって、

プログラムが提供する各機能毎に該機能を実現するために必要なタスクとその実行順序をセッションとして定義するセッション定義手段と、

セッションを実行するためのリソースと実行状態を管理するセッション管理手段とを備え、

前記セッション管理手段は、セッション起動時に当該セッションを構成する各タスクのインスタンスを生成するとともに、当該セッション及び当該セッションを構成する各タスクのインスタンスにセッションの識別情報及びインスタンスの識別情報を割り振り、セッション毎のタスク・ルックアップ・テーブル上で前記の各識別情報を管理する、

ことを特徴とするプログラム処理システムである。

【００１４】

但し、ここで言う「システム」とは、複数の装置（又は特定の機能を実現する機能モジュール）が論理的に集合した物のことを言い、各装置や機能モジュールが単一の筐体内にあるか否かは特に問わない。

【００１５】

本発明に係るプログラム処理装置は、タスク間で通信を行なうメッセージ通信手段をさらに備えている。メッセージ通信手段は、タスク間で通信を行なう際に、タスクが含まれるセッションのタスク・ルックアップ・テーブルを参照して送信先タスクのインスタンス識別情報に基づいてタスクを特定して、オペレーティング・システムが提供するタスク間通信機能を用いてメッセージ送受信を行なうことができる。このようなメッセージ通信手段は各タスク内に組み込むことができる。

【００１６】

本発明に係るアプリケーション・プログラムは、複数のタスクで構成され、提供する機能毎に該機能を実現するために必要なタスクとその実行順序がセッションとして定義されている。そして、マルチタスク環境下で実行することにより、同一のソースコードを持つタスクが複数のセッションで同時に起動する。

【００１７】

一般的なオペレーティング・システムはタスク間通信機能を備えているが、同一のソースコードを持つ複数のタスクを動的に特定することはできない。そこで、本発明では、セッションを起動したときに、セッションを機能毎にタスクを分割したときの各ブロックにブロックＩＤを割り振るとともに、ブロックＩＤとタスクＩＤの早見表であるタスク・ルックアップ・テーブルを作成する。ブロックＩＤは、セッション起動時に生成されるタスクのインスタンスの識別情報に相当する。

【００１８】

そして、あるタスクが別のタスクへメッセージを送信する際には、送信元タスクは送信先をブロックＩＤで指定し、タスク・ルックアップ・テーブルを参照することによりタスクＩＤを取得することができる。この後、送信先のタスクＩＤを指定して、オペレーティング・システムのメッセージ通信機能を用いてメッセージの送受信を行なうことができる。

【００１９】

したがって、本発明によれば、同一のソースコードからなるタスクが複数存在する場合にメッセージ送信先を特定して好適にメッセージ通信を行なうことができるようになる。

【００２０】

また、本発明の第２の側面は、複数のタスクを擬似的に並列に実行させるマルチタスク環境とタスク間通信機能を提供するマルチタスク・オペレーティング・システムが提供する実行環境下下で実行することにより複数の機能を実現するようにコンピュータ可読形式で記述されたコンピュータ・プログラムであって、
プログラムが提供する各機能毎に該機能を実現するために必要なタスクとその実行順序をセッションとして定義するセッション定義手段と、
セッションを実行するためのリソースと実行状態を管理するセッション管理手段と、
タスク間で通信を行なうメッセージ通信手段とを備え、
前記セッション管理手段は、セッション起動時に当該セッションを構成する各タスクのインスタンスを生成するとともに、当該セッション及び当該セッションを構成する各タスクのインスタンスにセッションの識別情報及びインスタンスの識別情報を割り振り、セッション毎のタスク・ルックアップ・テーブル上で前記の各識別情報を管理し、
前記メッセージ通信手段は、タスク・ルックアップ・テーブルから得られた送信先タスクのインスタンス識別情報に基づいて、前記オペレーティング・システムが提供するタスク間通信機能を用いてメッセージ送受信を行なう、
ことを特徴とするコンピュータ・プログラムである。

【００２１】

本発明の第２の側面に係るコンピュータ・プログラムは、コンピュータ・システム上で所定の処理を実現するようにコンピュータ可読形式で記述されたコンピュータ・プログラムを定義したものである。換言すれば、本発明の第２の側面に係るコンピュータ・プログラムをコンピュータ・システムにインストールすることによって、コンピュータ・システム上では協働的作用が発揮され、本発明の第１の側面に係るプログラム処理システムと同様の作用効果を得ることができる。

【００２２】

本発明のさらに他の目的、特徴や利点は、後述する本発明の実施形態や添付する図面に基づくより詳細な説明によって明らかになるであろう。

【００２３】

【発明の実施の形態】

以下、図面を参照しながら本発明の実施形態について詳解する。

【００２４】

Ａ．システム構成

図１には、本発明の実施に供されるプログラム処理システム１０の構成を模式的に示している。同図に示すように、プログラム処理システム１０は、プロセッサ１１と、ＲＡＭ（Ｒａｎｄｏｍ Ａｃｃｅｓｓ Ｍｅｍｏｒｙ）１２と、ＲＯＭ（Ｒｅａｄ Ｏｎｌｙ Ｍｅｍｏｒｙ）１３と、複数の入出力装置１４－１，１４－２…と、タイマ１５とを含んでいる。

【００２５】

プロセッサ１１は、演算処理システム１０のメイン・コントローラであり、オペレーティング・システム（ＯＳ）の制御下で、アプリケーションなどの各種のプログラム・コードを実行するようになっている。オペレーティング・システムは例えば組み込み型で構成される。

【００２６】

オペレーティング・システムがプログラム実行を管理・制御する単位は、一般に「タスク」と呼ばれる。本実施形態に係るプログラム処理システム１０では、プログラム中に複数のタスクが存在することを許容し、複数のタスクを時分割で実行し、タスクを頻繁に切り替えることにより、各タスクを擬似的に並列に実行させるマルチタスク機能を備えている。したがって、実際に計算を進める実体であるプロセッサの数よりも多くのタスクが存在し、多くのタスクをあたかも同時に実行しているかのように見える。

【００２７】

オペレーティング・システムは、プロセッサ１１により処理される各タスクに、他のタス

クと識別可能なタスクＩＤを割り振り、個々のタスクの実行を管理する。但し、オペレーティング・システムは、実行されるタスク間の関連性までは考慮しない。

【００２８】

一方、プログラム処理システム１０上で実行されるアプリケーション・プログラムは、マルチタスク環境下で実行される複数のタスクで構成され、複数の機能を提供することができる。例えば、ＤＶＤ記録再生装置における組み込みアプリケーションであれば、ＤＶＤの記録面に対する記録や再生などの複数の機能が提供される。本明細書中では、提供する機能サービス毎に当該機能を実現するために必要なタスクとその実行順序を「セッション」として定義する。また、「セッション・マネージャ」と呼ばれるタスクを備え、呼び出されたセッションを実行するためのリソースと実行状態を管理する。ここで言うリソース管理には、セッションに含まれるタスクのインスタンスの生成及びタスクが行なう入出力の管理などが含まれる。セッションを操作する側は、個々のタスクの状態を意識せず、必要な機能を実現できる。

【００２９】

プロセッサ１１は、バス１６によって他の機器類（後述）と相互接続されている。システム・バス１６上の各機器にはそれぞれ固有のメモリ・アドレス又はＩ／Ｏアドレスが付与されており、プロセッサ１１はこれらアドレスを指定することによって所定の機器へのアクセスが可能となっている。システム・バス１６は、アドレス・バス、データ・バス、コントロール・バスを含む共通信号伝送路である。

【００３０】

ＲＡＭ１２は、書き込み可能なメモリであり、プロセッサ１１において実行されるプログラム・コードをロードしたり、実行プログラムの作業データを一時格納したりするために使用される。プログラム・コードには、例えば、ＢＩＯＳ（Ｂａｓｉｃ ｉｎｐｕｔ・Ｏｕｔｐｕｔ ｓｙｓｔｅｍ：基本入出力システム）、周辺機器をハードウェア操作するためのデバイス・ドライバ、オペレーティング・システム、アプリケーションなどが挙げられる。

【００３１】

ＲＯＭ１３は、所定のコードやデータを恒久的に記憶するための不揮発メモリであり、例えば、ＢＩＯＳや始動時の自己診断プログラム（Ｐｏｗｅｒ ｏｎ Ｓｅｌｆ Ｔｅｓｔ：ＰＯＳＴ）などを格納している。

【００３２】

入出力装置１４には、ディスプレイ２１を接続するためのディスプレイ・インターフェース１４－１、キーボード２２やマウス２３のようなユーザ入力装置を接続するためのユーザ入力装置インターフェース１４－２、ハード・ディスク２４やメディア・ドライブ２５などの外部記憶装置を接続するための外部記憶装置インターフェース１４－３、外部ネットワークと接続するためのネットワーク・インターフェース・カード（ＮＩＣ）１４－４などが含まれる。但し、装備すべき入出力装置１４の種類や構成などは、プログラム処理システム１０の実体が何であるかに依存する。

【００３３】

ディスプレイ・インターフェース１４－１は、プロセッサ１１が発行する描画命令を実際に処理するための専用インターフェース・コントローラである。ディスプレイ・インターフェース１４－１において処理された描画データは、例えばフレーム・バッファ（図示しない）に一旦書き込まれた後、ディスプレイ２１によって画面出力される。

【００３４】

ＨＤＤ２４は、記憶担体としての磁気ディスクを固定的に搭載した外部記憶装置であり（周知）、記憶容量やデータ転送速度などの点で他の外部記憶装置よりも優れている。通常、ＨＤＤ２４には、プロセッサ１１が実行すべきオペレーティング・システムのプログラム・コードや、アプリケーション・プログラム、デバイス・ドライバなどが不揮発的に格納されている。ソフトウェア・プログラムを実行可能な状態でＨＤＤ２４上に置くことをプログラムのシステムへの「インストール」と呼ぶ。例えば、本発明を実現するオペレー

ティング・システムや、複数のタスクが存在するように設計されたアプリケーション・プログラムをＨＤＤ２４上にインストールすることができる。

【００３５】

メディア・ドライブ２５は、ＣＤ（Ｃｏｍｐａｃｔ　Ｄｉｓｃ）やＭＯ（Ｍａｇｎｅｔｏ－Ｏｐｔｉｃａｌ　ｄｉｓｃ）、ＤＶＤ（Ｄｉｇｉｔａｌ　Ｖｅｒｓａｔｉｌｅ　Ｄｉｓｃ）などの可搬型メディアを装填して、そのデータ記録面にアクセスするための装置である。

【００３６】

可搬型メディアは、主として、ソフトウェア・プログラムやデータ・ファイルなどをコンピュータ可読形式のデータとしてバックアップすることや、これらをシステム間で移動（すなわち販売・流通・配布を含む）する目的で使用される。例えば、本発明を実現するオペレーティング・システムや、複数のタスクが存在するように設計されたアプリケーション・プログラムを、これら可搬型メディアを利用して複数の機器間で物理的に流通・配布することができる。

【００３７】

ネットワーク・インターフェース１４－１は、Ｅｔｈｅｒｎｅｔ（登録商標）などの所定の通信プロトコルに従って、システム１０をＬＡＮ（Ｌｏｃａｌ　Ａｒｅａ　Ｎｅｔｗｏｒｋ）などの局所的ネットワーク、さらにはインターネットのような広域ネットワークに接続することができる。

【００３８】

ネットワーク上では、複数のホスト端末（図示しない）がトランスペアレントな状態で接続され、分散コンピューティング環境が構築されている。ネットワーク上では、ソフトウェア・プログラムやデータ・コンテンツなどの配信が行うことができる。例えば、本発明を実現するオペレーティング・システムや、複数のタスクが存在するように設計されたアプリケーション・プログラムを、ネットワーク経由でダウンロードすることができる。

【００３９】

各入出力装置１４－１，１４－２…には、割り込みレベルが割り当てられており、所定のイベント発生（例えばキーボード入力やマウス・クリックなどのＧＵＩ処理や、ハード・ディスクにおけるデータ転送の完了など）に応答して、割り込み要求信号線１９を介してプロセッサ１１に通知することができる。プロセッサ１１は、このような割り込み要求に応答して、対応する割り込みハンドラを実行する。

【００４０】

タイマ１５は、タイマ信号を所定周期で発生させる装置である。タイマ１５にも割り込みレベルが割り当てられており、割り込み要求信号線１９を介してプロセッサ１１に対して周期的な割り込みを発生する。

【００４１】

Ｂ．アプリケーション・プログラムの構成

本実施形態では、アプリケーション・プログラムは、マルチタスク環境下で実行される複数のタスクで構成され、複数の機能を提供することができる。図２には、アプリケーション・プログラム１００の構成を模式的に示している。

【００４２】

図示のアプリケーション・プログラム１００によれば、並行動作する複数のタスクの連携によって、ある特定の機能が実現される。この一連の動作のことをセッションと呼び、セッションの動作に係るタスクをセッション構成タスクと呼ぶ。図示の例では、タスク１１１、タスク１１２、タスク１１３の実行順序で構成されるセッション１０１と、タスク１１２、タスク１１４、タスク１１５の実行順序で構成されるセッション１０２が定義されている。

【００４３】

また、アプリケーション・プログラム１００を構成するタスクの１つはセッション・マネージャ１１０であり、セッション毎の動作の実行を管理する。セッション・マネージャ１

１０は、アプリケーション・プログラム１００内のすべてのセッションを一元的に管理するために、あらかじめ静的に定義されているセッション登録テーブル１３０を備えている。セッション登録テーブル１３０には各セッションの識別番号と、それぞれのセッションを構成するタスクの識別番号並びに実行順序、各セッションにおいて使用されるハードウェア資源が格納されている。

【００４４】

セッション登録テーブル１３０には、セッションを構成するタスクの数だけタスク識別番号が登録される。このとき、異なるセッションで同一のタスクを重複して登録することが可能である。図２に示す例では、タスク１１２は、セッション１０１及びセッション１０２の双方に登録されている。

【００４５】

セッション・マネージャ１１０は、セッション登録テーブル１３０を用いて、セッションを実行するためのリソースと実行状態を管理する。ここで言うリソース管理には、セッションに含まれるタスクのインスタンスの生成及びタスクが行なう入出力の管理などが含まれる。

【００４６】

また、セッションは、タスクと同様に、起動、中断、再開、終了など、動作の実行状態を遷移させることが可能である。これらセッション単位の実行状態の操作及び管理は、セッション・マネージャ１１０が外部からの要求（コマンド）に応じて行なう。

【００４７】

セッションの操作の際には、セッション・マネージャ１１０がセッションを構成するすべてのタスクの操作を一括して行なうので、セッション操作を要求する側では個々のセッション構成タスクを意識する必要はない。

【００４８】

セッションの起動は、セッション・マネージャ１１０が、セッション登録テーブル１３０から当該セッションを構成する各タスクの識別番号を読み込み、該当するタスクを一斉に起動することにより行なわれる。この際、セッション・マネージャ１１０は、該当するセッションに対してセッション管理テーブルを作成し、そこに当該セッション及び各セッション構成タスクの識別番号及び動作状態を記録し監視する。

【００４９】

また、セッションの中断及び再開の操作は、セッション・マネージャ１１０が、指定された識別番号を持つセッションに対し、セッション管理テーブルに記録されているセッション構成タスクの実行を一斉に中断又は再開することにより行なわれる。

【００５０】

また、セッションの終了では、セッション・マネージャ１１０は、セッション構成タスクを一斉に強制終了し、当該セッションのセッション管理テーブルを削除する。

【００５１】

上述したセッション操作の際におけるセッションを構成する各タスクの動作は、セッション・マネージャ１１０がオペレーティング・システムのシステム・コールを発行することにより制御する。１つのアプリケーション・プログラムにおいて複数のセッションを登録することが可能であり、複数のセッションを同時に並行動作させることが可能である。また、同種のセッションを複数同時に動作させることも可能である。また、異種のセッション間で同一のタスクを共有することも可能である。

【００５２】

ここで、ＤＶＤ記録再生装置における組み込みアプリケーションを例にとって、本発明に係るアプリケーション・プログラムの機能構成並びのその作用について説明する。

【００５３】

図３には、ＤＶＤ記録再生装置における組み込みアプリケーションの構成を模式的に示している。

【００５４】

アプリケーション・プログラムは、複数のタスクで構成される。図示の例では、アプリケーション・プログラムを構成するタスクは、ソフトウェア・ブロックに格納されており（但し、セッション・マネージャを除く）、再生制御タスク、記録制御タスク、入出力制御タスク、多重化タスク、逆多重化タスク、映像符号化タスク、映像逆符号化タスク、音声符号化タスク、音声逆符号化タスク、メモリ制御タスクなどのタスクからなる。また、アプリケーション・プログラムは、入出力制御装置、多重化装置、逆多重化装置、映像信号制御装置、音声信号制御装置、メモリなどのハードウェア資源を使用する。

【００５５】

また、アプリケーションは、並行動作する複数のタスクの連携によって、ＤＶＤディスクの記録面に対する再生や記録特定の機能が実現される。ここでは、これら再生や記録に関する一連の動作のことをそれぞれ「再生セッション」、「記録セッション」と呼ぶ。

【００５６】

図示の例では、再生セッションは、再生制御タスク、入出力制御タスク、逆多重化タスク、映像逆符号化タスク、音声逆符号化タスク、メモリ制御タスクというセッション構成タスクで定義される。また、記録セッションは、記録制御タスク、入出力制御タスク、多重化タスク、映像符号化タスク、音声符号化タスク、メモリ制御タスクというセッション構成タスクで定義される。

【００５７】

また、このアプリケーション・プログラムを構成するタスクの１つは、セッション・マネージャであり、セッション毎の動作の実行を管理する。セッション・マネージャは、このアプリケーション・プログラム内のすべてのセッションを一元的に管理するために、あらかじめ静的に定義されているセッション登録テーブルを備えている。セッション登録テーブルには、再生セッション、記録セッションなど各セッションの識別番号と、それぞれのセッションを構成するタスクの識別番号並びに実行順序、各セッションにおいて使用されるハードウェア資源が格納されている。

【００５８】

セッション登録テーブルには、セッションを構成するタスクの数だけタスク識別番号が登録される。このとき、異なるセッションで同一のタスクを重複して登録することが可能である。図３に示す例では、入出力制御タスクやメモリ制御タスクは、再生セッション及び記録セッションの双方に登録されている。

【００５９】

セッション・マネージャは、外部からセッションのオープンが指示されると、セッション登録テーブルを参照して、当該セッションを構成するタスクや、各タスクが使用するハードウェア資源を獲得する。そして、セッション構成タスクのインスタンスを生成するとともに、インスタンスによるハードウェア資源に対する入出力を管理する。

【００６０】

また、これらのセッションは、タスクと同様に、起動、中断、再開、終了など、動作の実行状態を遷移させることが可能である。これらセッション単位の実行状態の操作及び管理は、セッション・マネージャが外部からの要求に応じて行なう。セッションの操作の際には、セッション・マネージャがセッションを構成するすべてのタスクの操作を一括して行なうので、セッション操作を要求する側では個々のセッション構成タスクを意識する必要はない。

【００６１】

図３に示す例では、再生セッションや記録セッションを始め複数のセッションが登録されている。図示の例では、再生セッションや記録セッションなど複数のセッションが同時に並行動作している。また、２つの再生セッションが同時に動作している。また、再生セッションと記録セッションという異種のセッション間で、入出力制御タスクとメモリ制御タスクが共有されている。再生セッション１、記録セッション２、再生セッション１は、ソフトウェア・ブロックにあるセッション構成タスクのインスタンスを生成したイメージを示している。

【００６２】
Ｃ．タスクの特定及びタスク間のメッセージ通信
前項Ｂでは、複数のタスクで構成され、マルチタスク環境下で実行されるアプリケーション・プログラムが、提供する機能毎に該機能を実現するために必要なタスクとその実行順序をセッションとして定義し、タスクの１つとして構成されるセッション・マネージャが呼び出されたセッションを実行するためのリソースと実行状態を管理する、という点を説明した。これによって、複数のタスクをセッションという形で一元的に管理することができ、なお且つ実行効率、生産性、移植性、保守性といったタスク単位での開発のメリットを生かすことができる。また、セッションを操作する側では、個々のタスクの状態を意識することなく、必要な機能を実現することが可能である。また、セッション間でタスクを共有することができるので、従来のタスク単体よりも効率的なアプリケーション開発が可能となる。
【００６３】
一方、このようなセッション定義並びにセッション管理機能を備えたアプリケーション・プログラムは、マルチタスク・オペレーティング・システム上で動作させた場合に、同一のソースコードを持つタスクが複数のセッションで同時に起動する可能性が高い（例えば、図３を参照のこと）。
【００６４】
このような場合、オペレーティング・システムにより提供されているメッセージ機能やフラグ機能などをソースコード上で使用する場合、プログラム開発者はあらかじめセッション数を把握してコーディングをしなければならず、セッション数およびタスク数が静的に決まっていることが必要になる。ところが、セッション数およびタスク数が固定であることを前提でコーディングしてしまうと、機能拡張や再利用を行なうときにソースコードの変更部分が多大になり、プログラムの拡張性や再利用性が低下してしまう。
【００６５】
本発明は、このような問題を解決するために、セッション数及びタスク数が変更されても対応できる仕組みを実現するものである。
【００６６】
通常の組み込み機器（例えば、ＤＶＤ記録再生装置）では、主に機能ブロック毎にタスクを分割し実装する。まず、その機能毎にブロックＩＤを定義し、付加しておく。また、１つのタスクにはメッセージやメール・ボックスなどの受信部は１つのみとする。
【００６７】
図４には、本実施形態に係るプログラム処理システム１０のタスク間通信機能の仕組みを図解している。同図に示すように、プログラム処理システム１０のタスク間通信機能は、マルチタスク・オペレーティング・システム１４０と、セッション・マネージャ１１０と、セッション内の各機能ブロック毎のタスクを参照するためのタスク・ルックアップ・テーブル１５０と、このタスク・ルックアップ・テーブル１５０を管理するテーブル・コントロール部とで構成される。
【００６８】
セッション・マネージャ１１０によってセッションが構成され、そのセッションを構成するタスクがマルチタスク・オペレーティング・システム１４０によって起動される。
【００６９】
タスク・ルックアップ・テーブル１５０は、セッション内の各機能ブロック毎のタスクを参照するための早見表である。図５には、タスク・ルックアップ・テーブル１５０の構成例を模式的に示している。同図に示すように、タスク・ルックアップ・テーブル１５０は、セッション毎にテーブル化されており、各セッションについてのセッションＩＤ、セッションを構成する各ブロックのブロックＩＤ、並びにブロックに対応するタスクのタスクＩＤをそれぞれ登録している。
【００７０】
図６には、タスク・ルックアップ・テーブルを生成するための手順をフローチャートの形

式で示している。また、図７には、マルチタスク環境下でセッションＢが起動したときのタスク・ルックアップ・テーブルを生成するときの処理の流れを図解している。

【００７１】

コマンド・インターフェースなどを介してセッション・オープンの指示があると、セッションを起動し（ステップＳ１）、セッション・マネージャ１１０が空きセッション番号をテーブルに問い合わせる（ステップＳ２、Ｐ１）。

【００７２】

テーブル・コントロール部１６０内のテーブル管理部１６１は、タスク・ルックアップ・テーブル１５０を参照し、空きセッションＩＤを検索し（ステップＳ２、Ｐ３）、空きセッションＩＤをセッション管理部１６２に通知する（Ｐ５）。

【００７３】

また、セッション管理部１６２は、その空きセッションＩＤを使用してセッションを起動するため、テーブル管理部１６１にセッションＩＤを使用する旨を通知する（ステップＳ３、Ｐ１）。テーブル管理部１６１は、この通知に応答して、セッション・ルックアップ・テーブル１５０上に使用するセッションＩＤを確保する（Ｐ２）。セッション管理部１６２は、そのセッションＩＤでセッションを起動し、セッションに属する各タスクの起動をオペレーティング・システム１４０に指示する（ステップＳ４）。

【００７４】

セッション管理部１６２は、タスクの起動をオペレーティング・システム１４０に指示すると同時に、テーブル管理部１６３にブロックＩＤとタスクＩＤを通知する（ステップＳ５、Ｐ１）。そして、テーブル管理部１６１は、そのタスクＩＤとブロックＩＤをタスク・ルックアップ・テーブルに記録する（Ｐ２）。セッションＩＤとブロックＩＤの組み合わせにより、タスクのインスタンスを識別することができる。

【００７５】

ステップＳ４～Ｓ５の処理を起動させるタスクが無くなるまで行ない（ステップＳ６）、起動するすべてのタスク（すなわち、セッションを構成するすべてのタスク）を、当該セッションのタスク・ルックアップ・テーブルに登録する。そして、起動するすべてのタスクについての登録処理が完了すると、これに伴い、セッションの起動も完了する（ステップＳ７）。

【００７６】

なお、タスク・ルックアップ・テーブルの登録はセッションを構成する各タスクを起動する度に行なうのではなくて、タスクの起動を開始する前、あるいはすべてのタスクを起動した後に一括して登録するようにしても良い。

【００７７】

このようにして、起動した各セッション毎にタスク・ルックアップ・テーブルが作成され、セッション内での機能ブロックとタスクとの対応関係が明瞭となる。そして、起動中のあるタスクが他のタスクに対してメッセージ送信を行なおうとする際には、このタスク・ルックアップ・テーブルを参照することにより送信先となるタスクのＩＤを取得することができる。

【００７８】

図８には、タスク・ルックアップ・テーブルを参照することによってタスクＩＤを取得してメッセージを送受信するための処理手順をフローチャートの形式で示している。また、図９には、タスク・ルックアップ・テーブルから取得されたタスクＩＤを用いてオペレーティング・システムのタスク間通信機能を用いてメッセージ送受信を行なう流れを示している。

【００７９】

あるタスクが同一セッション内へメッセージの送信を行なうとき（ステップＳ８）、送信元のタスクは、送信先となるタスクのメール・ボックスＩＤを知る必要がある。本実施形態に係るプログラム処理システム１００では、メール・ボックスＩＤをタスクＩＤとしているので、送信先のタスクＩＤを求めることになる。

【００８０】
送信先のタスクＩＤを求めるためには、まず、当該セッションのタスク・ルックアップ・テーブルを参照して、ブロックＩＤを特定する（ステップＳ９）。他のセッションに向けた送信を行う場合はセッションＩＤをともに指定することで可能だが、ここでは同一セッション内のタスクに送ることを想定する。
【００８１】
ブロックＩＤを特定した後、送信元のタスクは、テーブル管理部１６１にブロックＩＤを渡す（Ｐ６）。タスクＩＤ取得部１６３は、テーブル管理部１６１に対し、ブロックＩＤからタスクＩＤを求めるよう指示する（Ｐ７）。テーブル管理部１６１は、この指示に応答して、送信元のタスクが属しているセッションのセッションＩＤを取得し（ステップＳ１０）、このセッションに該当するタスク・ルックアップ・テーブルよりタスクＩＤを取得し（Ｐ３）、これをタスクＩＤ取得部１６３に通知する（Ｐ８）。タスクＩＤ取得ブロックは、受け取ったタスクＩＤを、要求元のタスクに対して送信先タスクＩＤとして渡す（Ｐ９）。
【００８２】
これらの動作によって、送信元タスクは、自セッション内における送信先タスクのタスクＩＤを取得する（ステップＳ１１）。メッセージ送信先のメッセージ・ボックスＩＤは送信タスクのタスクＩＤと同じであるから、そのタスクＩＤをそのままメッセージ送信ＩＤとして使用する。この結果、メッセージ・ボックスのＩＤ（ｍｉｄ）は必然的に決定される（ｔｉｄ＝ｍｉｄ）。
【００８３】
次いで、送信元のタスクは、メッセージを送信するため、オペレーティング・システム１４０が持つメッセージ送信ブロック１４１に対して、メッセージ・ボックスＩＤ（ここでは送信先タスクのタスクＩＤ）とメッセージ本体を送信する（ステップＳ１２、Ｐ１０）。オペレーティング・システム１４０は、メッセージ・ボックスＩＤを特定し、メッセージ受信ブロック１４２を通して、送信先タスクにメッセージを渡す（ステップＳ１３、Ｐ１２）。
【００８４】
送信元タスクから送信先タスクへ送るメッセージの中に送信元のタスクＩＤを入れておくことで、送信先タスクから送信元タスクへのメッセージ送信は、テーブルからタスクＩＤを引くことなく送信元にメッセージを送り返すことができる（Ｐ１３、Ｐ１４、Ｐ１５）。
【００８５】
なお、上述の説明ではメッセージの送信方法を例にしたが、メール・ボックス、フラグ、データ・キューなどにおいても同様の手順で実施が可能であるということを充分理解されたい。
【００８６】
Ｄ．ソースコードの記述
本実施形態に係るアプリケーション・プログラムは、複数のタスクで構成され、提供する機能毎に該機能を実現するために必要なタスクとその実行順序がセッションとして定義されている。そして、マルチタスク環境下で実行することにより、同一のソースコードを持つタスクが複数のセッションで同時に起動する。
【００８７】
一般的なオペレーティング・システムはタスク間通信機能を備えているが、同一のソースコードを持つ複数のタスクを動的に特定することはできない。そこで、本実施形態では、セッションを起動したときに、セッションを各機能毎にタスクを分割したときの各ブロックにブロックＩＤを割り振るとともに、ブロックＩＤとタスクＩＤの早見表であるタスク・ルックアップ・テーブルを作成する。そして、あるタスクが別のタスクへメッセージを送信する際には、送信元タスクは送信先をブロックＩＤで指定し、タスク・ルックアップ・テーブルを参照することによりタスクＩＤを取得することができる。この後、送信先の

タスクＩＤを指定して、オペレーティング・システムのメッセージ通信機能を用いてメッセージの送受信を行なうことができる。

【００８８】

したがって、同一のソースコードからなるタスクが複数存在する場合にメッセージ送信先を特定して好適にメッセージ通信を行なうことができるようになる。以下では、タスクを構成するソースコードを参照しながら、本実施形態に係るメッセージ通信の仕組みについて説明する。

【００８９】

図１０には、同一のソースコードからなる複数のタスクの実体（インスタンス）が起動される様子を図解している。

【００９０】

図示の通り、タスクＡのソースコードには、コード“ｒｃｖ＿ｍｓｇ（＊ｍｓｇ）”によって構成されるメッセージの受信部と、コード“ｓｎｄ＿ｍｓｇ（＊ｍｓｇ）”によって構成されるメッセージの送信部が含まれている。タスクのソースコードには、受信部が１つでよいという点に十分留意されたい。

【００９１】

コマンド・インターフェースからのセッション・オープンの指示に応答して、オペレーティング・システムがセッションを構成するすべてのタスクを起動することによって、その実体すなわちインスタンスが生成される。そして、オープンが指示される複数のセッションに同じタスクが含まれている場合、図示のように、同じソースコードのタスクが複数起動される。既に説明したように、タスク起動時には、タスクＩＤとブロックＩＤの対応関係がセッションについてのタスク・ルックアップ・テーブルに登録される。

【００９２】

図１１には、本発明に係るメッセージ通信方式を取り入れた場合のタスクのソースコード内での送信部及び受信部の記述方法を示している。

【００９３】

タスクＡのソースコードには、コード“ｒｃｖ＿ｍｓｇ（ｔｉｄ，＊ｍｓｇ）”によって構成されるメッセージの受信部と、コード“ｓｎｄ＿ｍｓｇ（ｓｔｉｄ，＊ｍｓｇ）”によって構成されるメッセージの送信部が含まれている。

【００９４】

上述したように、本実施形態では、セッション内の機能ブロック毎にブロックＩＤを割り振り、セッション・ルックアップ・テーブルを作成している。したがって、送信元のタスクは、送信先のタスクのブロックＩＤ（ｂｉｄ）からそのタスクＩＤ（ｔｉｄ）を検索することができ、そのｔｉｄを用いてメッセージを送信することができる。

【００９５】

図１１に示す例では、送信元となるタスク１では、送信先のタスクをブロックＩＤで指定しておく（ｂｉｄ＝２）とともに、テーブル・ルックアップ・テーブルから当該ブロックＩＤを引くことで取得されるタスクＩＤを変数ｓｔｉｄに代入し（ｓｔｉｄ＝ｇｅｔ＿ｔｉｄ＿ｆｒｏｍ＿ｂｉｄ（ｂｉｄ））、ｓｔｉｄを送信先に指定して（ｓｎｄ＿ｍｓｇ（ｓｔｉｄ，＊ｍｓｇ））、メッセージ送信する。一方、送信先となるタスクのソースコードには、受信部が１つでよい（同上）。すなわち、コード“ｇｅｔ＿ｔｉｄ（ｔｉｄ）”により送信元タスクのｔｉｄを取得し、コード“ｒｃｖ＿ｍｓｇ（ｔｉｄ，＋ｍｓｇ）”により送信元タスクからのメッセージを受信することができる。

【００９６】

したがって、アプリケーション・プログラムに含まれるセッション数が増えたり、タスクの構成が変更したりしたとしても、ソースコードを変更する必要はない。

【００９７】

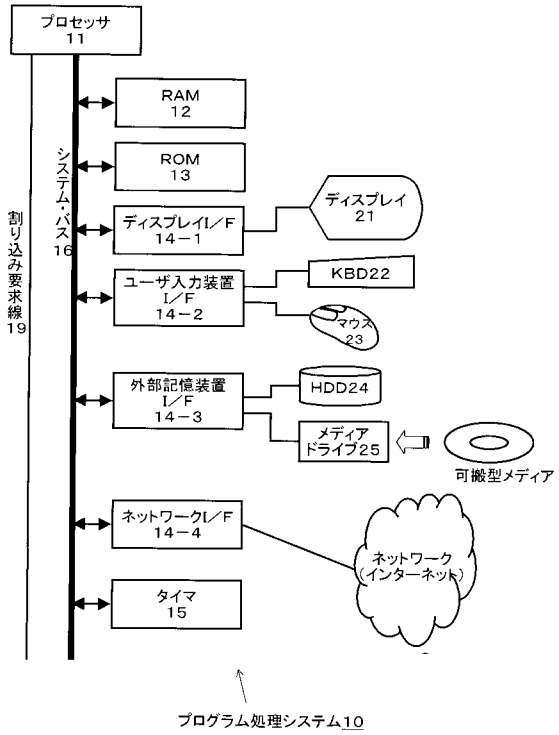
なお、あるタスクから別のタスクへメッセージを送信する場合、ソースコード上にＩＤを直接記述することによっても、同一のソースコードを持つタスクが複数のセッションで同時に起動する環境下でメッセージ送受信を実現することもできる。図１２には、この場合

の例を示している。
【００９８】
同図に示す例では、送信元のタスクのソースコードには、起動したときのセッションＩＤに応じた送信部を備えている。すなわち、セッション１でこのタスクが起動したときにはメッセージの送信先のタスクＩＤは２であり、それ以外のセッションで起動したときにはメッセージの送信先のタスクＩＤは３である。
【００９９】
［追補］
以上、特定の実施形態を参照しながら、本発明について詳解してきた。しかしながら、本発明の要旨を逸脱しない範囲で当業者が該実施形態の修正や代用を成し得ることは自明である。すなわち、例示という形態で本発明を開示してきたのであり、本明細書の記載内容を限定的に解釈するべきではない。本発明の要旨を判断するためには、冒頭に記載した特許請求の範囲の欄を参酌すべきである。
【０１００】
【発明の効果】
以上詳記したように、本発明によれば、アプリケーションが複数の機能を同時並行して動作する際に複数の機能で同一のタスクを好適に共有することができる、優れたプログラム処理システム及びプログラム処理方法、並びにコンピュータ・プログラムを提供することができる。
【０１０１】
また、本発明によれば、複数の機能で共有されるタスクのリソースと実行状態を好適に管理することができる、優れたプログラム処理システム及びプログラム処理方法、並びにコンピュータ・プログラムを提供することができる。
【０１０２】
また、本発明によれば、同一のソースコードからなるタスクが複数存在する場合にメッセージ送信先を特定して好適にメッセージ通信を行なうことができる、優れたプログラム処理システム及びプログラム処理方法、並びにコンピュータ・プログラムを提供することができる。
【０１０３】
本発明によれば、目的のタスク及びメッセージ送受信先を動的に特定する機能がないオペレーティング・システムによって提供される実行環境に、当該機能をオペレーティング・システム外で提供することができる。また、このようなメッセージ送受信機能を搭載したことで、仕様変更に伴うソースコードの変更量を軽減することができ、生産性の向上につながる。
【図面の簡単な説明】
【図１】本発明の実施に供されるプログラム処理システムの構成を模式的に示した図である。
【図２】アプリケーション・プログラムの構成を模式的に示した図である。
【図３】ＤＶＤ記録再生装置における組み込みアプリケーションの構成を模式的に示した図である。
【図４】プログラム処理装置１０のタスク間通信機能の仕組みを説明するための図である。
【図５】タスク・ルックアップ・テーブル１５０の構成例を模式的に示した図である。
【図６】タスク・ルックアップ・テーブルを生成するための手順を示したフローチャートである。
【図７】マルチタスク環境下でセッションＢが起動したときのタスク・ルックアップ・テーブルを生成するときの処理の流れを示した図である。
【図８】タスク・ルックアップ・テーブルを参照することによってタスクＩＤを取得してメッセージを送受信するための処理手順を示したフローチャートである。
【図９】タスク・ルックアップ・テーブルから取得されたタスクＩＤを用いてオペレーテ

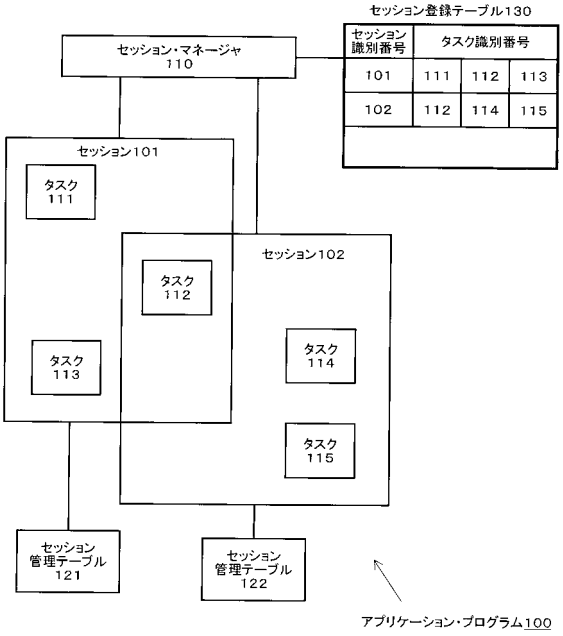
ィング・システムのタスク間通信機能を用いてメッセージ送受信を行なう流れを示した図である。
【図１０】同一のソースコードからなる複数のタスクの実体（インスタンス）が起動される様子を示した図である。
【図１１】本発明に係るメッセージ通信方式を取り入れた場合のタスクのソースコード内での送信部及び受信部の記述方法を説明するための図である。
【図１２】あるタスクから別のタスクへメッセージを送信する場合、ソースコード上にＩＤを直接記述することによってメッセージ送受信を実現する例を説明するための図である。

【符号の説明】
１０…プログラム処理システム
１１…プロセッサ
１２…ＲＡＭ
１３…ＲＯＭ
１４…入出力装置
１５…タイマ
１６…システム・バス
１９…割り込み要求線
２１…ディスプレイ
２２…キーボード
２３…マウス
２４…ＨＤＤ
２５…メディア・ドライブ
１００…アプリケーション・プログラム
１０１，１０２…セッション
１１０…セッション・マネージャ
１１１，１１２，１１３，１１４，１１５…タスク
１２１，１２２…セッション管理テーブル
１３０…セッション登録テーブル
１４０…マルチタスク・オペレーティング・システム
１５０…タスク・ルックアップ・テーブル
１６０…テーブル・コントロール部

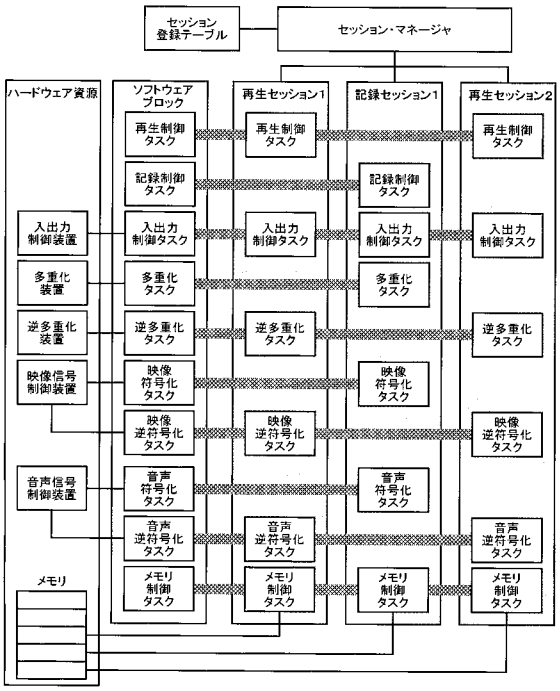
【図１】


プロセッサ
11
RAM
12
ROM
13
システム・バス
16
割り込み要求線
19
ディスプレイI／F
14－1
ディスプレイ
21
ユーザ入力装置
I／F
14－2
KBD22
マウス
23
外部記憶装置
I／F
14－3
HDD24
メディア
ドライブ25
可搬型メディア
ネットワークI／F
14－4
ネットワーク
（インターネット）
タイマ
15
プログラム処理システム10

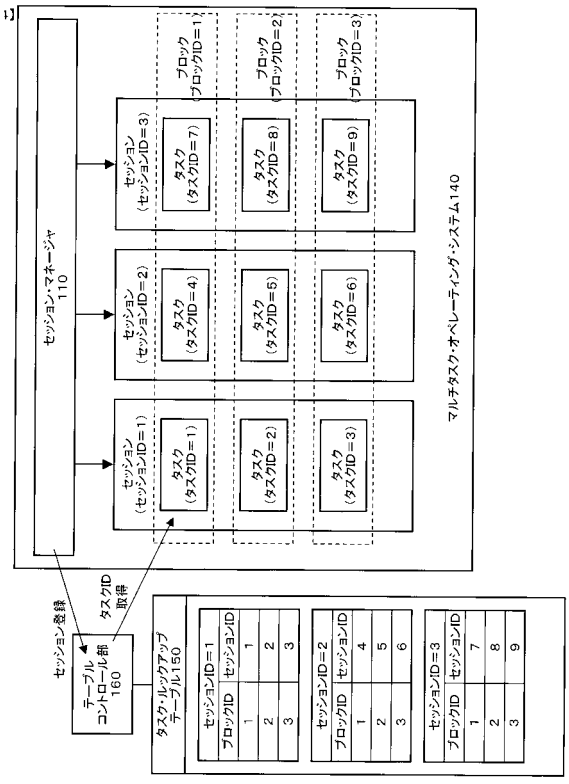
【図２】


セッション登録テーブル130
セッション識別番号
タスク識別番号
101
111
112
113
102
112
114
115
セッション・マネージャ
110
セッション101
タスク
111
セッション102
タスク
112
タスク
113
タスク
114
タスク
115
セッション
管理テーブル
121
セッション
管理テーブル
122
アプリケーション・プログラム100

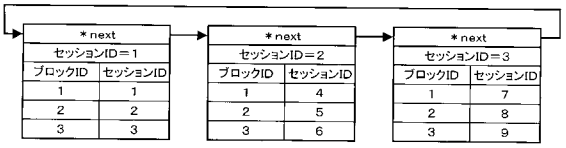
【図３】


セッション
登録テーブル
セッション・マネージャ
ハードウェア資源
ソフトウェア
ブロック
再生セッション1
記録セッション1
再生セッション2
再生制御
タスク
記録制御
タスク
入出力
制御タスク
多重化
タスク
逆多重化
タスク
映像
符号化
タスク
映像
逆符号化
タスク
音声
符号化
タスク
音声
逆符号化
タスク
メモリ
制御
タスク
入出力
制御装置
多重化
装置
逆多重化
装置
映像信号
制御装置
音声信号
制御装置
メモリ

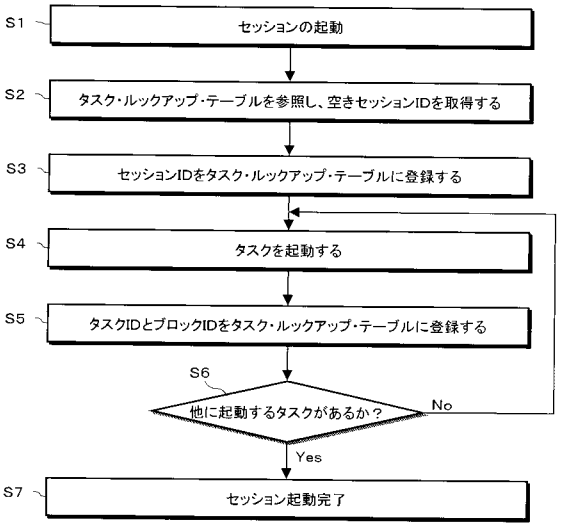
【図４】


セッション・マネージャ
110
ブロック
ブロックID＝1
ブロック
ブロックID＝2
ブロック
ブロックID＝3
セッション
（セッションID＝3）
セッション
（セッションID＝2）
セッション
（セッションID＝1）
タスク
（タスクID＝1）
タスク
（タスクID＝2）
タスク
（タスクID＝3）
タスク
（タスクID＝4）
タスク
（タスクID＝5）
タスク
（タスクID＝6）
タスク
（タスクID＝7）
タスク
（タスクID＝8）
タスク
（タスクID＝9）
マルチタスク・オペレーティング・システム140
セッション登録
タスクID
取得
テーブル部
コントロール部
160
タスク・リックアップ
テーブル150
セッションID＝1
セッションID
ブロックID
1
2
3
セッションID＝2
セッションID
ブロックID
4
5
6
セッションID＝3
セッションID
ブロックID
7
8
9

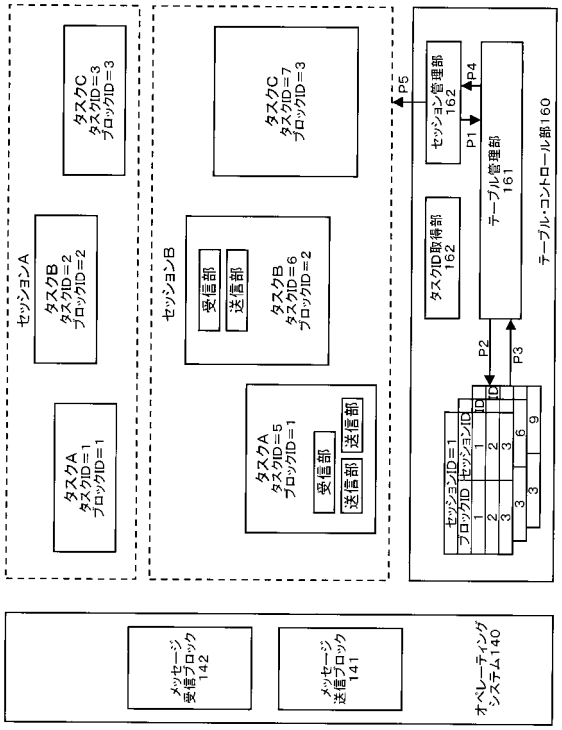
【図５】


＊next
セッションID＝1
ブロックID
セッションID
1
1
2
2
3
3
＊next
セッションID＝2
ブロックID
セッションID
1
4
2
5
3
6
＊next
セッションID＝3
ブロックID
セッションID
1
7
2
8
3
9

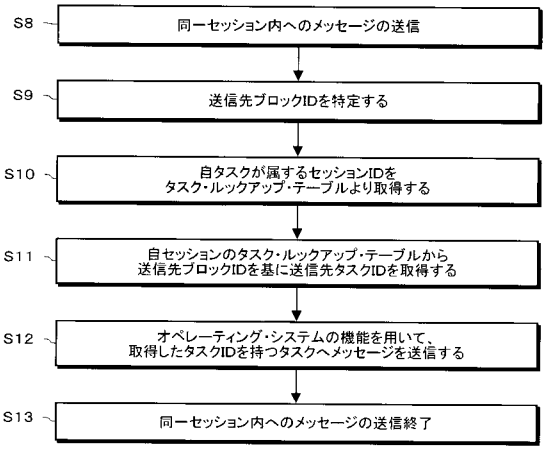
【図６】


S1
セッションの起動
S2
タスク・ルックアップ・テーブルを参照し、空きセッションIDを取得する
S3
セッションIDをタスク・ルックアップ・テーブルに登録する
S4
タスクを起動する
S5
タスクIDとブロックIDをタスク・ルックアップ・テーブルに登録する
S6
他に起動するタスクがあるか？
No
Yes
S7
セッション起動完了

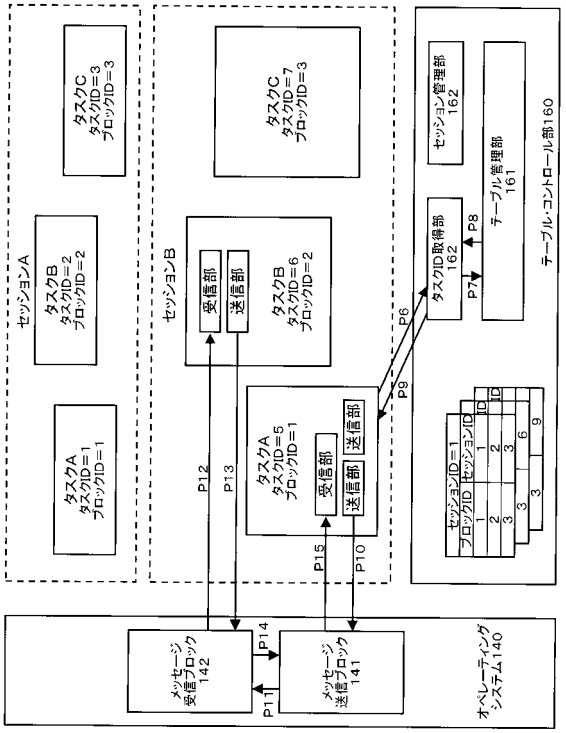
【図７】


セッションA
タスクA
タスクID=1
ブロックID=1
タスクB
タスクID=2
ブロックID=2
タスクC
タスクID=3
ブロックID=3
セッションB
タスクA
タスクID=5
ブロックID=1
受信部
送信部
タスクB
タスクID=6
ブロックID=2
タスクC
タスクID=7
ブロックID=3
テーブル・コントロール部160
セッション管理部
162
タスクID取得部
162
テーブル管理部
161
P1
P2
P3
P4
P5
セッションID=1
ブロックID
セッションID
メッセージ
送信ブロック
141
メッセージ
受信ブロック
142
オペレーティング
システム140

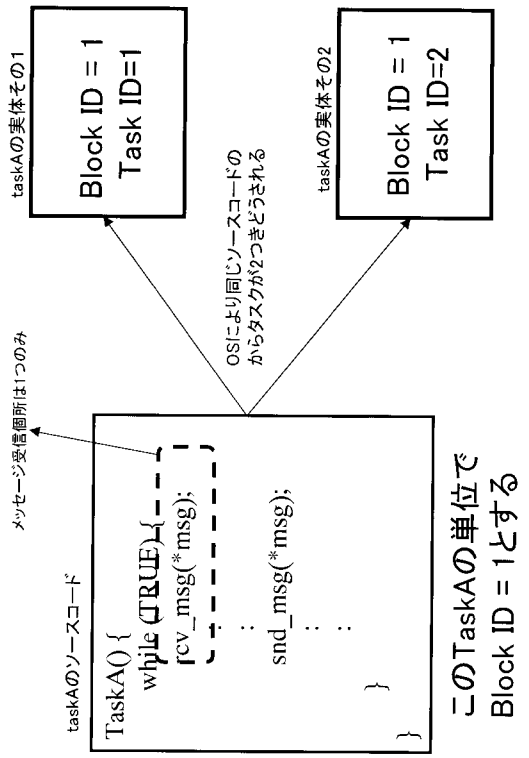
【図８】


S8
同一セッション内へのメッセージの送信
S9
送信先ブロックIDを特定する
S10
自タスクが属するセッションIDを
タスク・ルックアップ・テーブルより取得する
S11
自セッションのタスク・ルックアップ・テーブルから
送信先ブロックIDを基に送信先タスクIDを取得する
S12
オペレーティング・システムの機能を用いて、
取得したタスクIDを持つタスクへメッセージを送信する
S13
同一セッション内へのメッセージの送信終了

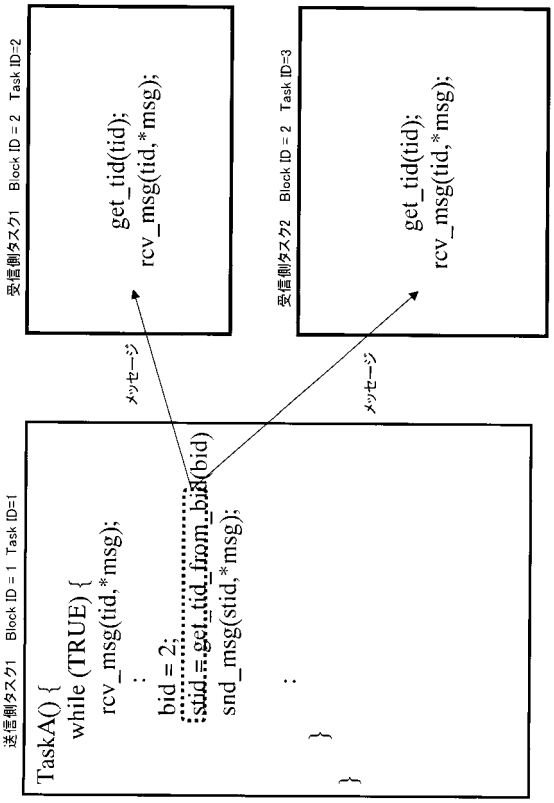
【図９】


セッションA
タスクA
タスクID=1
ブロックID=1
タスクB
タスクID=2
ブロックID=2
タスクC
タスクID=3
ブロックID=3
セッションB
タスクA
タスクID=5
ブロックID=1
受信部
送信部
タスクB
タスクID=6
ブロックID=2
タスクC
タスクID=7
ブロックID=3
テーブル・コントロール部160
セッション管理部
162
タスクID取得部
162
テーブル管理部
161
P6
P7
P8
P9
P10
P11
P12
P13
P14
P15
セッションID=1
ブロックID
セッションID
メッセージ
送信ブロック
141
メッセージ
受信ブロック
142
オペレーティング
システム140

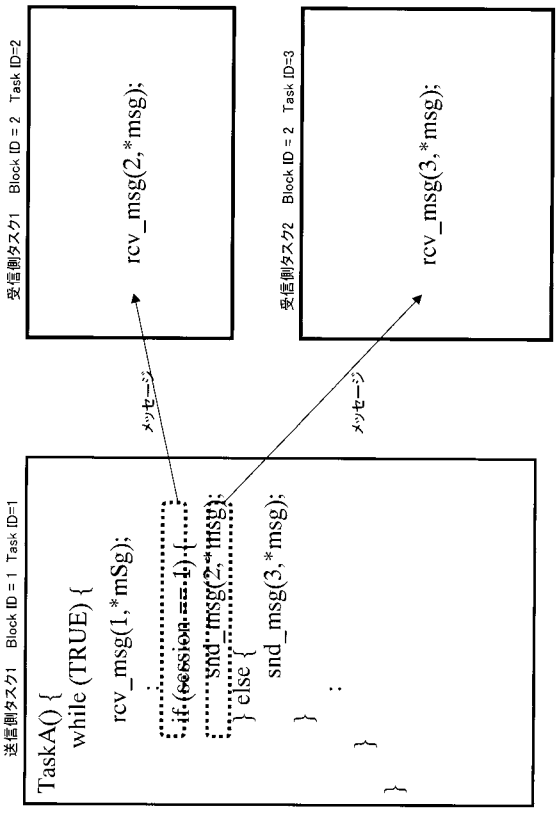
【図１０】


taskAの実体その1
Block ID = 1
Task ID=1
taskAの実体その2
Block ID = 1
Task ID=2
OSにより同じソースコードの
からタスクが2つきどうされる
メッセージ受信個所は1つのみ
taskAのソースコード
TaskA() {
while (TRUE){
rcv_msg(*msg);
:
snd_msg(*msg);
:
}
}
このTaskAの単位で
Block ID = 1とする

【図１１】


受信側タスク1 Block ID = 2 Task ID=2
get_tid(tid);
rcv_msg(tid,*msg);
受信側タスク2 Block ID = 2 Task ID=3
get_tid(tid);
rcv_msg(tid,*msg);
メッセージ
メッセージ
送信側タスク1 Block ID = 1 Task ID=1
TaskA() {
while (TRUE) {
rcv_msg(tid,*msg);
:
bid = 2;
stid = get_tid_from_bid(bid)
snd_msg(stid,*msg);
:
}
}

【図１２】


受信側タスク1 Block ID = 2 Task ID=2
rcv_msg(2, *msg);
受信側タスク2 Block ID = 2 Task ID=3
rcv_msg(3, *msg);
メッセージ
メッセージ
送信側タスク1 Block ID = 1 Task ID=1
TaskA() {
while (TRUE) {
rcv_msg(1, *mSg);
:
if (session == 1){
snd_msg(2,*msg);
} else {
snd_msg(3, *msg);
}
:
}
}