(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第3599738号
(P3599738)
(45)発行日　平成16年12月8日(2004.12.8)　(24)登録日　平成16年9月24日(2004.9.24)

(51)Int.Cl.[7]		F I		
G06F	9/46	G06F	9/46	340A
G04G	7/00	G04G	7/00	
G04G	9/00	G04G	9/00	303Z
G06F	3/14	G06F	3/14	340A
H04N	5/265	H04N	5/265	

請求項の数 12　(全 27 頁)

(21)出願番号	特願平6-525382
(86)(22)出願日	平成6年1月6日(1994.1.6)
(65)公表番号	特表平8-510344
(43)公表日	平成8年10月29日(1996.10.29)
(86)国際出願番号	PCT/US1994/000141
(87)国際公開番号	WO1994/027234
(87)国際公開日	平成6年11月24日(1994.11.24)
審査請求日	平成13年1月9日(2001.1.9)
(31)優先権主張番号	08/060,771
(32)優先日	平成5年5月10日(1993.5.10)
(33)優先権主張国	米国(US)

(73)特許権者
オブジェクト　テクノロジー　ライセンシング　コーポレイション
アメリカ合衆国　９５０１４―２０８３　カリフォルニア州　クパチーノ　メールステーション　３８―オーティーエル　インフィニト　ループ　１
(74)代理人
弁理士　谷　義一
(74)代理人
弁理士　阿部　和夫
(72)発明者　ミルネ，スティーブ，エイチ．
アメリカ合衆国　９４３０１　カリフォルニア州　パロ　アルト　ウェブスター　ストリート　１３７

最終頁に続く

(54)【発明の名称】マルチメディア同期システム

(57)【特許請求の範囲】

【請求項１】

タイムベースメディアシーケンスの提示の新しいクロックレートを特定する同期要求に応答して、タイムベースメディアシーケンスの提示を同期させるコンピュータシステムにおいて、

（ａ）ストレージと、

（ｂ）マスタータイム信号を供給する手段と、

（ｃ）現在時間値を供給するクロックオブジェクトであって該クロックオブジェクトはストレージ内に内在し、およびマスタータイム信号、当初のクロックレート値および当初のオフセット値のクロック関数にしたがって現在時間値を計算するための現在時間算出手段を有するクロックオブジェクトと、

（ｄ）クロックオブジェクトと協働し、現在時間値と同期してタイムベースメディアシーケンスを提示するメディアシーケンス提示手段と、

（ｅ）同期要求に応答してクロックオブジェクトを同期させるクロック同期手段であって、前記提示手段が新しいクロックレートでメディアシーケンスを提示し、その一方で、同期時に現在時間値を変えないようにするクロック同期手段であり、マスタータイム信号、新しいクロックレート、当初のクロックレート値および当初のオフセット値の新しいクロック関数にしたがって新しいオフセット値を計算する手段およびクロックオブジェクトの当初のクロックレート値と当初のクロックオフセット値とを新しいクロックレートと新しいオフセット値とにそれぞれ置き換える手段を有するクロック同期手段と

を備えたことを特徴とするコンピュータシステム。
【請求項２】
請求項１に記載のコンピュータシステムにおいて、ディスプレイと、ディスプレイ上にアイコングラフィックを表示する手段と、アイコングラフィックのユーザによる操作を検出する手段と、アイコンのユーザーによる操作に応答して同期要求を生成する手段とを有することを特徴とするコンピュータシステム。
【請求項３】
請求項２に記載のコンピュータシステムにおいて、クロックオブジェクトはクロックオブジェクトを表わす第１のクロックアイコンをディスプレイ上に生成する手段を有し、前記コンピュータシステムがさらに、ディスプレイ上に第２のクロックオブジェクトを表わす第２のクロックアイコンを表示する手段を含む第２のクロックオブジェクトと、ディスプレイ上で第１のクロックアイコンを第２のアイコンと結び付ける幾何学的形状を通じて、クロックオブジェクトを第２のクロックオブジェクトに連結する手段であって、第２のクロックオブジェクトをクロックオブジェクトに同期させるためにクロックオブジェクトの現在時間属性がマスタータイム信号として第２のクロックオブジェクトに供給される手段とを備えたことを特徴とするコンピュータシステム。
【請求項４】
請求項１に記載のコンピュータシステムにおいて、マスタータイム信号を供給する第２のクロックオブジェクトを有し、該第２のクロックオブジェクトはストレージに内在する第２の現在時間値を計算する手段を有し、第２の現在時間値はマスタータイム信号として供給されることを特徴とするコンピュータシステム。
【請求項５】
請求項１に記載のコンピュータシステムにおいて、新しいオフセット信号を受信し、当初のオフセット値を新しいオフセット信号に置き換え、クロック関数にしたがって新しい現在時間値を前記現在時間算出手段に計算させる手段をさらに備えたことを特徴とするコンピュータシステム。
【請求項６】
請求項１に記載のコンピュータシステムにおいて、クロック関数は、オフセット値とレート値とマスタータイム信号値の積、との和として現在時間値を計算することを特徴とするコンピュータシステム。
【請求項７】
ストレージ、ディスプレイおよび、マスタータイム信号を供給するメカニズムを有するコンピュータ上で、タイムベースメディアシーケンスの提示の新しいクロックレートを特定する同期要求に応答して、タイムベースメディアシーケンスの提示を同期する方法において、
（ａ）ストレージ内に、マスタータイム信号、当初のクロックレート値および当初のオフセット値のクロック関数にしたがって現在時間値を計算するメカニズムを有するクロックオブジェクトを作成するステップと、
（ｂ）現在時間値と同期してタイムベースメディアシーケンスを提示するステップと、
（ｃ）同期要求に応答し、マスタータイム信号、新しいクロックレート、当初のクロックレート値および当初のオフセット値の新しいクロック関数にしたがって新しいオフセット値を計算するステップと、
（ｄ）メディアシーケンスが新しいクロックレートで提示され、その一方で、同期時に現在時間値を変えないようにするために、クロックオブジェクトの当初のクロックレート値と当初のクロックオフセット値とを新しいクロックレートと新しいオフセット値とにそれぞれ置き換えるステップと
を備えたことを特徴とする方法。
【請求項８】
請求項７に記載の方法において、
（ｅ）ディスプレイ上にアイコングラフィックを表示するステップと、

（ｆ）アイコングラフィックのユーザによる操作を検出するステップと、
（ｇ）アイコングラフィックのユーザによる操作に応答して同期要求を生成するステップと
をさらに備えたことを特徴とする方法。
【請求項９】
請求項７に記載の方法において、ステップ（ａ）において作成されたクロックオブジェクトは、クロックオブジェクトを表わす第１のクロックアイコンをディスプレイ上に生成する手段を有し、前記方法はさらに、
（ｈ）第２のクロックオブジェクトを作成するステップであって、当該第２のクロックオブジェクトは第２のクロックオブジェクトを表わす第２のクロックアイコンをディスプレイ上に表示する手段を有するステップと、
（ｉ）第２のクロックオブジェクトをクロックオブジェクトの同期させるために該クロックオブジェクトの現在時間属性がマスタータイム信号として第２のクロックオブジェクトに供給されるように、ディスプレイ上で第１のクロックアイコンを第２のアイコンと結び付ける幾何学的形状を通じて、クロックオブジェクトを第２のクロックオブジェクトにリンクするステップと
を備えたことを特徴とする方法。
【請求項１０】
請求項７に記載の方法において、
（ｊ）ストレージ内に内在する第２の現在時間値を計算するメカニズムを有する第２のクロックオブジェクトを作成するステップと、
（ｋ）第２の現在時間値をマスタータイム信号としてクロックオブジェクトに供給するステップと
をさらに備えることを特徴とする方法。
【請求項１１】
請求項７に記載の方法において、
（ｌ）新しいオフセット信号を受信するステップと、
（ｍ）クロックオブジェクト内の当初のオフセット値を新しいオフセット信号と置き換えて、現在時間値を計算するメカニズムが新しい現在時間値を計算するステップと
をさらに備えたことを特徴とする方法。
【請求項１２】
請求項７に記載の方法において、ステップ（ａ）は、
（a1）オフセット値とレート値とマスタータイム信号値の積、との和として現在時間値を計算するためのクロック関数を有するクロックオブジェクトを作成するステップ
を有することを特徴とする方法。
【発明の詳細な説明】
著作権所有表記
本特許出願の一部には著作権保護の対象となる内容が含まれている。著作権者は米国特許商標局の特許ファイルまたは記録のままでの何人の特許文書または特許開示事項のファクシミリ複製を妨げるものではないが、そうでない場合は、一切の著作権を留保する。
関連特許出願の相互参照
本特許出願は特許出願、すなわち、発明の名称が“Object Oriented Framework System"、Debra L.Orton,David B.Goldsmith,Christopher P.Moeller、およびAndrew G.Heningerによるものであり、1992年12月23日に出願され、被譲渡人がTaligentである特許出願の関連出願であり、この出願の開示事項を本明細書の一部とする。
発明の分野
本発明、一般にコンピュータシステムの改善に関わり、さらに詳しくは種々のマルチメディアのイベントのタイミングを同期させるためのシステムに関する。
発明の背景
マルチメディアは恐らくコンピュータシステムに対して最も急速に成長しているアプリケ

ーションである。ユーザはますますコンピュータを使用してグラフィック、サウンドおよびイメージ情報をエンドユーザに提供しようとしている。ユーザはマルチメディアのプレゼンテーションを取り扱うための人間工学的なインターフェースをますます要求している。過去においては、システムクロックはある時刻でサウンドの再生を開始するために、あるいは、特定の時刻にコンピュータのディスプレイ上に情報を提供するためにしばしば使用される。けれどもミュージックやサウンドのプレゼンテーションを、説明用のマルチメディアプレゼンテーションとしての情報の表示と同期させるツールは可能とはなっていなかった。

本発明の同期能力を有していない現在のマルチメディアシステムとしてはアップル（Apple）社のQuicktimeおよびマイクロソフト（Microsoft）社のVideo for Windows（“NEWMEDIA"3月号の“It's Showtime"pp.36－42、1993に記載）を例として挙げることができる。従来技術が直面する同期の問題を解決するという重要性については“IEEE Spectrum"3月号，“Interactive Multimedia",pp.22－31（1993）および“The Technology Framework",IEEE Spectrum,pp.32－39（1993）において議論されている。論文は、本発明の同期能力でのみ効果的に提供されるマルチメディアの実施可能性についての重要性を指摘している。

発明の概要

したがって、本発明の第１の目的は、記憶装置（Storage）およびディスプレイを有するコンピュータを使用してマルチメディアのプレゼンテーションを行う過程において種々のマルチメディアのイベントを同期させるシステムおよび方法を提供することにある。クロックオブジェクトは記憶装置上で定義づけられ、現時点の内／外部のソース（source）と関連づけられる。クロックオブジェクトはディスプレイ上で表示可能であり、一度それらのリンクが定義されると隠匿することも可能となる。オーディオ、ビジュアルさもなくば他のマルチメディアイベントに代表される１以上のマルチメディアオブジェクトは定義され、特別なクロックオブジェクトまたは複数のクロックオブジェクトにリンクされる。それ故、プロセッサはマルチメディアオブジェクトを関連のクロックオブジェクトに同期させる。

【図面の簡単な説明】

第１図は好適実施例に従ったパーソナルコンピュータシステムのブロック図である。

第２図は好適実施例に従ったクロックオブジェクトを示す図である。

第３図は好適実施例に従ったシステムタイマーを示す図である。

第４図は好適実施例に従ってクロックオブジェクトへ時間を入力するための外部ソースを示す図である。

第５図は好適実施例に従ったマスタークロックオブジェクトおよびスレーブクロックオブジェクトを示す図である。

第６図は好適実施例に従ってクロックオブジェクトの階層を駆動するタイムソースを示す図である。

第７図は好適実施例に従ったタイムベースシーケンスを示す図である。

第８図は好適実施例に従ったタイムベースメディアプレイヤーを示す図である。

第９図は好適実施例に従ってマスタークロックに外部的に同期するマルチメディアプレイヤーを示す図である。

第10図は好適実施例に従って別のクロックオブジェクトに対してマスターとして交互に動作する他のクロックオブジェクトおよびこの他のクロックオブジェクトに対してマスターとして動作するクロックオブジェクトを示す図である。

第11図は好適実施例に従ったオーディオおよびビデオシーケンスの同期を示す図である。

第12図は好適実施例に従ってビデオプレーヤー（スレーブ）に同期するオーディオプレーヤー（マスター）を示す図である。

第13図は好適実施例に従ってオーディオシーケンス、すなわち、ビデオシーケンスを２倍の早さで動作するように同期させたオーディオシーケンスを示す図である。

第14図はコンピュータディスプレイ、すなわち、ビデオシーケンスを好適実施例に従って

オーディオシーケンスの２倍の早さに同期させたコンピュータディスプレイを示す図である。
第15図は好適実施例に従ってマスターおよびスレーブとして動作するプレーヤーオブジェクトを示す図である。
第16図は好適実施例に従ったジョグ／シャトルのノブを示す図である。
第17A図は好適実施例に従って複数のマルチメディア上でマスターとして動作するジョグ／シャトル　ノブオブジェクトを示す図である。
第17B図および第17C図は好適実施例に従ったシャトルおよびジョグのノブのフローチャートである。
第18図は好適実施例においてコンピュータディスプレイ上に現れる時のクロック階層を示す図である。
第19図は好適実施例に従って別のクロックを通じてタイムソースに間接的に連結されたクロックを示す図である。
第20図は好適実施例に従ったSetTime関数のロジックを示す詳細フローチャートである。
第21図は好適実施例に従ったTickleWakeups関数のロジックを示す詳細フローチャートである。
第22図は好適実施例に従った現タイム取得オペレーションのロジックを示す詳細フローチャートである。
第23図は好適実施例に従ったディレイおよびアラーム処理のロジックを示す詳細フローチャートである。
第24図は好適実施例に従ったSync TO My Time処理を示す詳細フローチャートである。
第25図は好適実施例に従ったオーディオプレイヤーを示す図である。
第26図は好適実施例に従ったマスターとしてのオーディオプレイヤーを示す図である。
第27図は好適実施例に従ってクロックに従属するオーディオプレイヤーを示すフローチャートである。
第28図は好適実施例に従った日の出のイメージのシーケンスを示す図である。
第29図は好適実施例に従ったグラフィックプレイヤーに関連するロジックを示すフローチャートである。
第30図は好適実施例に従ったMIDIシーケンスおよびMIDIトラックフォーマットを示す図である。
第31図は好適実施例に従ってドライバーを通じてシーケンスを実行するMIDIプレイヤーを示す図である。
第32図は好適実施例に従ったMIDIプレイヤーの詳細ロジックを示すフローチャートである。
第33図は好適実施例に従った空のデスクトップを示す図である。
第34図は好適実施例に従って選択されたクロックオブジェクトを定義する表示画面を示す図である。
第35図は好適実施例に従った種々のクロックおよびマルチメディアオブジェクトを示す図である。
第36図は好適実施例に従って互いにリンクされたクロックオブジェクトおよびマルチメディアオブジェクトを示す図である。
第37図は好適実施例に従ってオーディオオブジェクトに同期させたビジュアルオブジェクトを示す図である。
発明の詳細な説明
本発明はIBM社製PS/2やAPPLE社製Macintoshコンピュータのようなパーソナルコンピュータに内在するオペレーティングシステムに関連して好適に実施される。代表的なハードウェア環境を第１図に示す。それは本発明に従ったワークステーション、すなわち、通常のマイクロプロセッサーのような中央演算処理ユニット10およびシステムバス12を介して相互接続された多数の他のユニットを有するワークステーションの典型的なハードウェアの配置を示す。第１図に示すワークステーションはランダムアクセスメモリ（RAM）14、リ

ードオンリメモリ（ROM）16、ディスクユニット20のような周辺機器をバスに接続するためのI/Oアダプター、キーボード24とマウス26とスピーカー26とマイクロフォン28および／またはタッチスクリーン装置（不図示）のような他のユーザインターフェース機器とをバスに接続するためのユーザインターフェースアダプター22、ワークステーションをデータ処理ネットワークに接続するための通信アダプター34およびバスを表示器38へ接続するためのディスプレイアダプター36を含んでいる。ワークステーションはAPPLE System/t operating systemのようなオペレーティングシステムをその中に内蔵する。

好適実施例において本発明はプログラミング技術を支援するオブジェクトを使用してＣ＋＋プログラミング言語で実現される。当業者であれば理解しようがObject－OrientedProgramming（OOP）は実在するソフトウェアであり、データ構造およびデータ上のオペレーション（data structures and operations on the data）から成る。併せ、これらの要素はその特徴、すなわち、そのデータ要素およびその働きを示し、操作機能を示す特徴によってオブジェクトが実世界のものいずれかを仮想的にモデル化することができるようにする。この方法でオブジェクトは人およびコンピュータのように具体的な物のモデルを作成することができ、また、数のような抽象的な概念または幾何学的な概念のモデルを作成することができる。オブジェクトテクノロジーの恩恵で３つの基本的な原理：カプセル化、多形性（polymorphism）、継承性（inheritance）が生じる。

オブジェクトはそのデータ構造およびその機能を働かせるアルゴリズムを隠匿あるいはカプセル化する。これらの実現の細部を表面化する代わりにオブジェクトは外部情報なしでそれらの抽象概念をきれいに表現するインタフェースを提供する。多形性はカプセル化をさらに１ステップ進める。その考えは多くの形を１つのインタフェースというものである。ソフトウェアの要素は別の構成要素（compronent）がなにかを正確に知らないでもその構成要素を要求することができる。その要求を受け取る構成要素はそれを判断し、どのようにして要求を実行するかをその変数およびデータに従って計算を行う。第３の原理は継承性であり、それによりデベロッパー（developers）は既存のデザインおよびコードを再使用することができる。この能力によりデベロッパーはデータ消去状態からソフトウェアを作成することが避けられる。むしろ継承を通じてデベロッパーは動作を引く継ぐサブクラスを受け継ぎ、それを特別のニーズに合致するようにカストマイズすることができる。

従来技術のアプローチではオブジェクトを階層化し、手続き環境下ではライブラリをクラス化する。マーケット上の多数のアプリケーションのフレームワークはこのデザインアプローチを採用している。このデザインにおいてはモノリシックのオペレーティングシステムのトップに１以上のオブジェクト層がある。このアプローチはカプセル化、多形性、継承の３つ全ての原理を使用し、手続きプロブラミング技法について大幅に改善されているがこのアプローチには制限がある。これらの困難性はデベロッパーにとっては彼ら自身のオブジェクトを再使用することは容易であるが他のシステムからのオブジェクトを使用することは難しくデベロッパーは手続きのオペレーテーティングシステムコール（procedural Operating System calls）を以て下位の非オブジェクト層にさらに到達する必要がある。

オブジェクト指向言語の別の解釈はアプリケーション開発に対するフレームワークアプローチである。フレームワークの最も理にかなった定義の１つがイリノイ大学のRalph E.JohnsonおよびPurdueのVincent F.Russoから発せられている。彼らの論文、Reusing Object－Oriented Designs,University of Illinois tech report UIUCDCS91－1696において、彼らは次のような定義、すなわち、“抽象的なクラスは共同で１組の責務を実行する１組のオブジェクトのデザインである。したがって、フレームワークは定義されたコンピュータ処理の責務を共同で実行する１組のオブジェクトクラスである。”を提唱している。プログラミングの見地からすると、フレームワークはワーキングアップリケーションの予め組み立てられた構造を提供するオブジェクトクラスを連結した欠くことの出来ないグループ群である。たとえば、ユーザインターフェースフレームワークはウィンドウ、スクロールバー、メニュー等の描画をサポートし、デフォルトを提供する。フレームワークはオブジェクト技術を基礎にしているので、この動作は継承され、デベロッパーがフレームワー

クを拡張できないようにし、専門知識の特別な領域での問題をカストマイズできないようにする。プログラマーは元のコードを変更せず、むしろソフトウウェアを拡張するので、これは、伝統的なプログラミングを超える大きな利点である。加えて、フレームワークはアーキテクチャをガイドし、モデル化を行うので、デベロッパーはコードの階層を通してやみくもに作業せず、同時に所定の動作を提供してそれらを問題領域（problem domain）に対して無縁なもとする。

ビジネスの見地からすると、フレームワークは専門知識を特別な知識領域においてカプセル化または具体化する方法と見られる。会社の開発組織、独立のソフトウェアベンダー（ISV）およびシステムインテグレーターは特別なエリア、すなわち、製造、会計あるいは通貨取引のような特別なエリア内で専門知識を獲得する。この専門知識は彼らのコード内で具体化される。フレームワークによって組織は組織のコード内でそれを具体化することによりその専門知識の共通特徴を掴まえパッケージ化する。第１に、これによってデベロッパーは専門知識を使用するアプリケーションを作成または拡張することができ、１度解決された問題、ビジネスルールおよびデザインを一貫して実施、使用することができる。また、フレームワークおよびフレームワークを背景として具体化された専門知識は、縦断的なマーケット、すなわち、製造、会計あるいは、バイオ技術のような専門知識をパッケージ化、再販売、展開し、技術を促進化するための流通メカニズムを有するマーケットでは含み資産を持つ。

歴史的には、フレームワークはコンピュータ処理のプラトホーム上のメインストリーム概念として最近出現しただけのものである。この転移はＣ＋＋のようなオブジェクト指向言語の利用に助けられている。伝統的にはＣ＋＋は商的に設置されたパーソナルコンピュータよりもむしろUNIXシステムおよび研究者のワークステーション上で見受けられた。数多くの大学および研究プロジェクトを今日の商業的なフレームワークおよびクラスライブラリに対して先駆者足らしめたのがＣ＋＋およびSmalltalk他、のような他のオブジェクト指向言語である。これらのいくつかの例はスタンフォード大学からのInterViews、カーネルギーメロン大学のthe Andrew toolkitおよびチューリッヒ大学のET＋＋frameworkに存在する。

あなたがたが関連するシステムがどのレベルかおよびあなた方が解決しようとする問題がどのような種類かに応じて多種のフレームワークがある。フレームワークの種類の範囲はユーザインターフェースの開発を支援するアプリケーションフレームワークから通信、印刷、ファイルシステムサポート、グラフィックス等のような基本的なシステムソフトウェアサービスを提供する低レベルのフレームワークにわたっている。アプリケーションフレームワークの商的な例では名前をいくつか挙げるとMacApp（Apple）、Bedrock（Symantec）、OWL（Borland）、NeXTStep App Kit（NeXT）およびSmalltalk－80 MVC（ParcPlace）がある。

フレームワークでのプログラミングは他種のシステムに慣れたデベロッパーにとっては新しい思考法を要する。事実、これまでの伝来的な感覚ではまったく“プログラミング”のようではない。DOSまたはUNIXのような古いスタイルのオペレーティングシステムではデベロッパー自身のプログラムが全ての構造を与える。オペレーティングシステムはシステムコールすなわち、デベロッパーのプログラムはサービスを必要とする時に呼出し（call）、サービスが供給された時に制御をリターンするようなシステムコールを通じてサービスを供給する。

フレームワークを使用するとこれが逆転する。デベロッパーはフロー制御に対してもはや責を負わない。デベロッパーは実行フローの見地からプログラミングタスクを前以って理解しなくてはならない。むしろ、思考はオブジェクトの責務という見地にならなければならない。それらオブジェクトはタスクが実行する時にフレームワークによって決定処理を実行しなければならない。デベロッパーにより書かれるルーチンはデベロッパーが書かなかった、デベロッパーがこれまでに見たことがないようなコードにより活性化される。制御フローにおけるこのフリップフロップは手順的なプログラミングだけしか経験していないデベロッパーにとっては意味ありげな心理学的なバリヤーに成り得る。けれども一度で

もこれを理解すると、フレームワークプログラミングは他のタイプのプログラミングよりも作業が少なくなる。
アプリケーションプログラムがデベロッパーに組立機能を与えるのと同じ方法で好適実施例に含まれるもののようなシステムフレームワークはシステムレベルサービスを与えることにより同じ概念を導入している。それらをサブクラス／オーバーライドに用いてシステムプログラマーのようなデベロッパーはカストマイズされた解決（solutions）を作成する。たとえば、オーディオ、MIDI、アニメーション等のような新しく多様な機器をサポートするための土台を提供するマルチンメディアフレームワークを考える。新しいある種の機器をサポートする必要があるデベロッパーはデバイスドライバー（device driver）を書かなければならない。フレームワークでこれを行うためにはデベロッパーはその新しい機器に対して規定する動作および特徴を提供する必要があるのみである。
この場合におけるデベロッパーはマルチメディアフレームワークにより呼び出されるあるメンバー関数のためにあることを実行する。デベロッパーにとって速効的な利益は機器の各カテゴリーに必要な総称コードはマルチメディアフレームワークにより既に与えられている点にある。これはデバイスドライバーの開発者にとって書き込み、テストおよびデバッグするコードが少ないことを意味する。システムフレームワークを使用する別の例ではSCSI機器、NuBusカードおよびグラフィック機器のための分離I/Oフレームワークを持つべきであろう。継承する機能があるので、各フレームワークはその機器のカテゴリーに見られる共通の機能をサポートする。他のデベロッパーはそれ故全種類の機器へのこれら結合インターフェースに依存できた。
好適実施例ではフレームワークの概念を採り入れそれを全体システムに適用している。商業的なまたは会社のデベロッパー、システムインテグレーターあるいはOEMにとってこれは、MacAppのようなフレームワークに関して明らかにされた全利点を、テキストおよびユーザインタフェースのような物のためのアプリケーションレベルで取り込むことが可能となるのみならず、グラフィックス、マルチメディア、ファイルシステム、I/O、テスト等のようなサービスのためのシステムレベルでも取り込むことが可能となる。
好適実施例のアーキテクチャーにおけるアプリケーションの作成は本質的にはフレームワークのプロトコルに固着するドメインの特定のパズル（domain－specific puzzle）を書くようなものである。このようにして、プログラミングの全体概念が変化している。多数のAPI階層を呼び出すためのコードを何行も書く代わりに、この環境内の前から存在するフレームワークからクラスを受け継ぐことによりおよび新しい動作を加えることによりおよび／または所望の受け継いだ動作を無効化することによりソフトウェアは開発される。したがって、デベロッパーのアプリケーションは他のフレームワークアプリケーションで書かれたおよび分割されたコードの集積となる。デベロッパーは他人の各仕事に基づいて構築することができるので、これはパワフルな概念である。これによりデベロッパーは柔軟性を以て必要の多少に応じてカストマイズすることができる。いくつかのフレームワークはまさにそのまま使用するであろう。いくつかのケースではカストマイズの量は最小となるであろうし、デベロッパーが挿入するパズルピース（puzzle piece）は小さくなる。他のケースでは、デベロッパーは大きい修正を行い、完全に新しい何かを作成するかもしれない。
好適実施例では第１図に示すようにソフトウェアクロックはタイムベース（time base）に対して応答する。一方、RAM14に内蔵され、CPU10の制御の下にあるかまたはバス12または通信アダプター34を介して外部的に装着されるマルチメディアプレイヤーは以下のクロックに対して応答する。セントラルプレイヤーがシステムの全ての処理を整理し管理する必要はない。このアーキテクチャーは柔軟性を与え、付加される新しいメディアとして増大の拡張性に対処する。本発明の好適実施例の革新的なシステムおよび方法により以下の特徴が生じる。
・プレイヤーはアプリケーションが要求するどのような正確さのレベルに対しても同期を維持する必要があるごとにクロックでチェックを行う。
・多数のクロックを互いに同期させることができ、クロックの間のリニアーな時間関係を

用いてシーケンスのオフセットおよびスピードを互いに関連づける。
・多数のタイムソース
どのクロック（故にどのプレイヤー）もシステムクロックだけでなく、外部のタイムソース（例としてビデオテープからのタイムコード）あるいはユーザ動作（ソフトウェアシャトルノブ、ジョグノブ、他のコントローラのような）に対しても同期させることができる。
・クロックは時間の後戻りをすることができる。
。クロックは多数のアドレススペースを拡大することができる。
用語の定義
ソフトウェアクロックとは？
第２図に示すようにソフトウェアクロックは以下の機能を実行するオブジェクトである。
・クロックは、タイムオブジェクト（time object）200により表される現在時間を持つ。タイムオブジェクトは、時間のあるポイントをピコ秒で測定する浮動小数点数である。メンバー関数は現在時間をセッティングし、現在時間を得る。メンバーファンクションはあるオブジェクトを動作させる関数である。現在時間もまた、値を減少させることができその場合、クロックは上述したように後戻りする。
・遅れ：クロックはある時間、いわゆるディレイ時間までスレッド（thread）が到達することをブロックすることができる。もしもクロックが前進する場合はクロックの現在時間がディレイ時間以上になった時はスレッドはブロックされない。もしクロックが後進する場合、クロックの現在時間がディレイ時間以下の時にスレッドはブロックされない。
・アラーム：クロックは、ある時間、いわゆるアラーム時間に到達すると、IPC（Interprocess Communication）メッセージをポートに送信することができる。もし、クロックが前進する場合、クロックの現在時間がアラーム時間以上になるとIPCメッセージが送信される。もし、クロックが後進する場合、クロックの現在時間がアラーム時間以下となった時にIPCメッセージが送信される。
・マルチタイムソース（multiple time sources）：通常、クロックの時間はシステムタイマーに基づいてリアルタイムで進む。システムタイマーは第３図においてタイムソースオブジェクト300により表されている。しかしながら、クロックもまた、第４図に示すようにビデオテープレコーダ（VTR）からコンピュータへ入力するSociety of Motion Picture and Engineers（SMPTE）タイムコードのような別のタイムソースに同期するとができる。クロックの現在時間はVTRから到来するタイムコードに従う。VTRがスピードアップするとクロックも追従してスピードアップする。VTRがスローダウンすると、クロックも追従してスローダウンする。VTRが後進すると、クロックも追従して後進する。クロックはVTRに追従したままである。
タイムソースは、まさに、クロックのように現在時間を持つ。リニアな関数がタイムソースの現在時間とクロックの現在時間との間の相関関係を定めるのに使用される。

$$t_{clock} = a \cdot t_{timesource} + b$$

ここで、t_{clock}はクロックの現在時間である。
$t_{timesource}$はタイムソースの現在時間である。
ａはタイムソースの現在時間に対するクロックの現在時間の比率を決定する浮動小数点数である。
ｂはクロックの現在時間のオフセットを決定するタイムオブジェクトである。
この関数をクロック関数と呼ぶことにする。
・同期：複数のクロックもまた互いに同期させることができる。マスタークロックおよびスレーブクロックの２つのクロックが与えられると、ちょうどタイムソースのようにクロック関数が定められる。

$$t_{slave} = a \cdot t_{master} + b$$

ここで、t_{slave}はスレーブクロックの現在時間である。
t_{master}はマスタークロックの現在時間である。
ａはマスタークロックの現在時間に対するスレーブクロックの現在時間の比率を定める浮

動小数点数である。
ｂはマスタークロックの現在時間に対するスレーブクロックのオフセットを定めるタイムオブジェクトである。
たとえば、第５図ではクロック510（スレーブ）はクロック500（マスター）に同期している。スレーブクロックはまた、別のクロックに対してはマスターに成りうる。これらの２つのルール故にクロックの任意のツリー構造を作成することができる。第６図はソフトウェアクロックの階層を駆動するタイムソースを示す。
２つのクロックメンバー関数が、クロックを互いに同期させるために与えられる。それらはスレーブクロック上で呼び出される。１つをSyncToKeepMyTime（）であり、クライアントがマスタークロックおよび速度を規定できるようにし、オフセット、すなわち、同期の瞬間に、フレーブクロックの現在時間の値を変化させないようにするオフセットを計算できるようにする。
加えて、メンバー関数SetRate（）を使用してマスターに対するスレーブクロックの相対速度を直接的に設定することができる。
異なるアドレススペースに存在するクロックオブジェクトおよびタイムソースオブジェクトの事例も互いに同期させることができる。
スタートおよびストップ：クロックは停止することができる。その場合、その現在時間はそのマスターの変化の有無にかかわらず変化しない。停止したクロックは再スタートすることができクロックは引き続きそのマスターに追従する。速度は変化しない。クロック停止以前と同様の状態となる。クロックがスタートした時点ではその現在時間はクロックが停止したときと同じなので、オフセットは変化する。
設定速度：クロック関数の速度をいずれかの浮動小数点値に設定することができる。速度が変化するとき、クロックの現在時間は変化しないのでオフセットは変化する。
タイムベース・メディアシーケンス（time－based media sequence）とは？
タイムベース・メディアシーケンスは、抽象的なベースクラス、すなわち、オーディオ、ビデオ、アニメーションあるいはミュージカル・インスツルメント・ディジタル・インターフェース（MIDI）データさもなくば時間的に変化するいずれかの他のデータのクリップを表わすために使用することができる抽象的なベースクラスである。それは時間０で始まり、タイムオブジェクトにより表わされる継続時間である。第７図は３秒の継続時間のタイムベース・メディアシーケンスの例である。タイムベース・メディアシーケンスのサブクラスは、オーディオ、ビデオ、アニメーションおよびMIDIシーケンスを実行するために用いられる。
タイムベース・メディアプレイヤーとは？
タイムベース・メディアプレイヤー（以後、プレイヤーと称す）はタイムメディアシーケンスを再生または記録にするために用いられる抽象的なベースクラスである。このクラスはオーディオ、ビデオおよびMIDI用のプレイヤーを作成するためにサブクラス化される。
第８図はタイムベース・メディアプレイヤーの一例を示す。プレイヤーは関連ソフトウェアクロックオブジェクトを有する。ソフトウェアクロックの現在時間800はプレイヤーの再生位置を表わす。再生位置810はテープレコーダ上の再生ヘッドに類似しており、プレイヤーのクロックはデフォルトクロックかまたはシステムタイマーのようなタイムソースの同期する。このようなプレイヤーは内部的に同期すると言われる。けれどもプレイヤーのクロックは別のタイムソースまたは別のプレイヤーのクロックを含むクロックに同期させることができる。このようなプレイヤーは外部的に同期すると言われる。
第９図において、プレイヤー900は外部的にマスタークロック900に同期している。マスタークロック910はスレーブクロックとの同期に応じてプレイヤー900の再生位置を決定する。
たとえば、プレイヤーのクロックは、
$t_{player} = 1 \cdot t_{master} + 0$
のようにマスタークロックに同期する。もし、マスタークロックがスローダウンすると、プレイヤーの再生も応じてスローダウンする。もし、マスタークロックがスピードアップ

すると、プレイヤーもスピードアップする。あらゆる場合において、プレイヤーはマスターに同期したままである。
外部的に同期する場合、再生の間、再生位置がいつもそのクロックの現在時間に反映するようにすることはプレイヤーの責任である。同様のことが記録に関しても言える。
プレーヤーのクロックは、また、外部クロック、すなわち、革って、別のクロックに対してマスターになり得る外部クロックに対するマスタークロックとしても用いられる。
第10図において、プレイヤーＡのクロック1000は、クロックX1010、すなわち、代わってプレイヤーＢのクロック1020に対してマスターとして動作するクロックX1010に対してマスターとして動作する。プレイヤーA1000が再生を行っているとき、クロックX1010およびプレイヤーB1020の双方はプレイヤーＡのクロックに追従する。プレイヤーＡが停止すると、そのクロックは動かない。また、クロックＸおよびプレイヤーＢのクロックも動かない。内部に同期するとき、クロックの現在位置が再生位置に常に反映するようにすることはプレイヤーの責任である。
タイムベース・メディアシーケンスの同期
マスターおよびスレーブプレイヤー
第11図に示すようにオーディオおよびビデオシーケンスを互いに同期させるため２つのプレイヤーのクロックは第12図に示すように互いに同期させられる。ビデオプレイヤーは常にオーディオプレイヤーに追従する。もし、オーディオプレイヤーがスピードアップすると、ビデオプレイヤーもスピードアップする。もしオーディオプレイヤーがスローダウンすると、ビデオもスローダウンする。もし、オーディオが停止すると、ビデオもストップする。オーディオが後進し始めるとビデオも応じて後進する。このようにしてオーディオおよびビデオだけでなく、２つのどのようなタイムベース・メディアプレイヤーでも同期させられる。
共通クロックへのプレイヤーの同期
第13図に示すようにオーディオシーケンスの２倍の速さでビデオシーケンスを再生するようなオーディオシーケンスの１秒後でビデオシーケンスをスタートするためには第14図で示すようにそれらのクロックを単一のマスタークロックに同期するように２つのプレイヤーは作成されなければならない。任意の個数のタイムベース・メディアシーケンスを単一のマスタークロックに同期させることができる。
ディジィーチェーン結合のプレイヤー
第15図に示すようにプレイヤーはマスターおよびスレーブの双方となることが可能である。このようなプレイヤーは外部的に同期させられ、そしてそれ故にスレーブのように動作しなければならない。マスターのように動作することは余分な努力なしに同時に実現される。スレーブ1520はマスター／スレーブプレイヤー1510のクロックに単純に結合可能である。複数のプレイヤーを互いにディジィーチェーン結合することができる。
ソフトウェア　ジョグ／シャトル　ノブ速度コントローラ
ジョグ／シャトル　ノブは、収集のタイムベース・メディアシーケンスの再生の速度を制御するために用いられる。シャトルモードのときは、ノブの位置が再生の速度を制御する。ジョグモードのときは、ノブが再生位置を制御する。ソフトウェア　ジョブ／シャトル　ノブはソフトウェアクロックの使用を実現できる。ジョブ／シャトル　ノブの絵画表現は第16図に示すようにコンピュータスクリーン上に描かれ、マウスで制御される。ジョグモードにあるときは、スクリーン上のコントローラノブ1610の位置はシーケンス中の時間位置に変換することができる。タイムソースは時間位置を表わすのに用いられる。ノブが新しい位置にセットされたときはいつでもタイムソースの現在時間は、シーケンス中の時間位置に対応してセットされる。
クロックオブジェクトはタイムソースに従属する。そのため、１以上のプレイヤーをクロックオブジェクトに同期させることにより第17A図に示すようなノブにより決定される位置に追従する。シャトルモードであるとき、クロックオブジェクトはシステムタイムソースに従属する。スクリーン上のノブの位置は再生速度に変換することができる。ユーザがスクリーン上のノブの位置を変更する毎に対応速度がクロック上でセットされる。クロッ

クの速度のセッティングにより全ての同期しているプレイヤーは対応してその速度を変更する。

第17B図は好適実施例におけるシャトル制御処理に関連の詳細なロジックについてのフローチャートである。マウスボタンの押下が機能ブロック1700において検出されると、判断ブロック1702でテストが実行されて、マウスカーソルが第16図の1610で示すように、シャトルコントローラ上のノブ上に位置しているかを判定する。もし、カーソルがそのように位置している場合、機能ブロック1704でシャトルクロック速度が０にセットされ、シャトルクロックは現在時間０にセットされる。機能ブロック1706でプレイヤーはシャトルクロックに従属し、プレイヤーは機能ブロック1708でスタートし、処理はターミナル1710で完了する。

機能ブロック1720でマウス位置の変化が検出されると、正午12時と、ノブ位置との間の角度が“度”の単位で機能ブロック1722で計算され、テストが判定ブロック1724でテストが実行されて角度が負であるか判定される。もし、そうであれば機能ブロック1728でSIGN＝１、角度∂＝－∂がセットされ、制御は判定ブロック1730に進む。もし、機能ブロックでテストが偽となったときはSIGNは１にセットされる。次に、判定ブロック1730でテストが実行され∂≧０度および＜80度であるかの判定がなされる。RATEがSIGN＊≧/80度に設定され、制御は機能ブロック1742に進み、シャトルクロック速度がRATEと同じ値に設定される。もし、そうでないときは、別のテストが判定ブロック1736で実行され、∂が≧80度および＜100のときはRATEがSIGN＊1.0にセットされ、制御機能ブロック1742に進み、シャトルクロック速度がRATEと同じ値に設定される。もし、そうでないときはRATEはSIGN＊１＋（∂－100度）/80°と同じ値に設定され、制御がファンクションブロック1742に進み、RATEと同じ値にシャトルクロック速度が設定される。

マウスボタンが機能ブロック1744でアップしていると、次に、プレイヤーは機能ブロック1746で停止し、プレイヤーは機能ブロック1748でプレイヤーの内部クロックに縦続する。

第17C図は本発明に従ったジョグモード処理のロジックを詳細化したフローチャートである。マウスボタンの押下が機能ブロック1750で検出されると、テストが判定ブロック1752で実行され、マウスカーソルがノブ上にあるかが判定される。もし、そうでないならば、処理はターミナル1762で放棄される。マウスがノブ上にあるときは、次に、ファンクションブロック1754でジョブタイムソースの現在時間が０にセットされる。最後のノブ位置が正午に等しくセットされる。度の単位の時間が２秒/360度の値に等しく機能ブロック1756でセットされ、プレイヤーは機能ブロック1758でジョグタイムソースに縦続し、プレイヤーは機能ブロック1760でスタートする。

機能1764でマウスボタンの位置が変化すると、現在のノブの位置が機能ブロック1766で計算される。最後のノブ位置および現在のノブ位置の間Δが機能ブロック1768で計算され、機能ブロック1770でTIME DELTAが計算され、ジョグタイムソースの現在時間がTIME DELTAの値に等しく設定される。

機能ブロック1774でマウスボタンがアップしていると、次に、機能ブロック1776でプレイヤーはストップし、プレイヤー機能ブロック1780でプレイヤーの内部ブロックに縦続するよう設定される。

タイムソースおよびクロックオブジェクトの実施

タイムソースの実行

タイムソースおよびそれに全てのクロックが同期しているセットは、クロック階層と呼ばれる。第18図はクロック階層の図であり、好適実施例ではコンピュータディスプレイ上に現われる。タイムソースはその現在時間を供給し、アラームおよびディレイを実施しなければならない。アラームおよびディレイは、ウェイクアップと呼ばれる。IPCメッセージ（アラーム）またはスレッドのブロック解除（ディレイ）はウェイクアップ起動（firing wake up）と呼ばれる。タイムソースはクロック階層中のいずれかのクロック上にセットされたいずれかおよび全てのウェイクアップの起動に応答する。

クロック階層中の各クロックのために、タイムソースは直接関数（direct function）、すなわち、タイムソースの現在時間からクロックの現在時間への変換に用いられる直接関

数を保持する。直接関数の形態は、
$t_{clock}＝a・t_{timesource}＋b$
ここでt_{clock}はクロックの現在時間である。
$t_{timesource}$はタイムソースの現在時間である。
ａはタイムソースの現在時間に対するクロックの現在時間の相対速度を決定する浮遊小数点数である。
ｂはタイムソースの現在時間に対する相対的なクロックの現在時間のオフセットを決定するタイムオブジェクトである。
逆直接関数は、他方向、すなわち、クロックの現在時間からタイムソースの現在時間：
$t_{timesource}＝（t_{clock}－b）/a$
への変換に用いられる。
クロックが間接的に１以上のクロックを通じてタイムソースに結合されると、直接関数を代数学的は数値代入によりこれまでどおりに計算することができる。たとえば、第19図のクロックY1900を考える。クロックX1910は式：
$t_{clockx}＝a_x・t_{timesource}＋b_x$
を有するタイムソース1920と比較した関数である。
クロックY1900のクロックX1910に対する関数は、
$t_{clocky}＝a_y・t_{clockx}＋b_y$
である。
クロックY1900の直接関数を代入により計算することができる。
$t_{clocky}＝a_y・（a_x・t_{timesource}＋b_x）＋b_y$
$＝（a_y・a_x）・T_{timesource}＋（a_y・b_x＋b_y）$
$＝a・t_{timesource}＋b$
ここで、
$a＝a_y・a_x$
$b＝a_y・b_x＋b_y$
タイムソースには駆動、非駆動の２種類のタイプがある。駆動タイムソースは、クライアントの現在時間のセッティングにより絶えず更新される。これは、駆動タイムソースのSetTime（）メンバー関数を呼び出すことにより実現される、SetTime（）の実行処理は第20図に示すフローチャート中に示されている。ターミナル2000で処理が始まり、ここで、新しい現在時間を記憶する機能ブロック2010でパラメータ“new time"がSetTime（）に渡される。
タイムソースの方向は、判定ブロック2020で“new time"をタイムソースの古い現在時間と比較することで決定される。もし、大であればタイムソースは前進し、制御は判定ブロック2040に渡される。もし、小であれば、タイムソースは後進し、制御は判定ブロック2030に渡される。もし、等しい場合、タイムソースの方向は変化しない。“Next time"は次回のウェイクアップを起動する必要がある時間である。もし、タイムソースが前進している場合、“new time"が現在時間以下のときは、ウェイクアップは遅れる。もし、タイムソースが後進している場合、もし、タイムソースが後進している場合、“next time"が現在時間以上のときにウェイクアップは遅れる。“ticklewakeups"と記載された機能ブロック2050は全てのアラームを起動し、ディレイが遅れる。ウェイクアップの起動は以下のタイムソースウェイクアップリストのセクションにおいてより詳細に説明する。処理はターミナルブロック2060で完了する。
非駆動のタイムソースは現在時間がどうなっているかを常時、述べる必要は無い。それは、基本オペレーティングシステムがたえず現在時間の情報を得るシステムタイマーのようなタイムソースのために用いられる。非駆動のタイムソースは現在時間を検出する方法を知って承知しており、それはメンバー関数GetNextTime（）、すなわち、アラームまたはディレイを起動すべき“NEXT TIME"を返す関数を有している。GetNextTime（）は新しいアラームまたはディレイがタイムソースに負荷されたときはいつでも呼び出される。新しいアラームまたはディレイを起動する必要があるときは、タイムソースのTickleWakeUps

（）メンンバー関数、それは遅れているウェイクアップのいずれかを起動するが、そのメンバー関数を呼び出すことができる。これにより駆動タイムソースに必要とされたような、Settime（）の常時呼び出しを必要としないタイムソースをインプリメンター（implementor）が作成することができる。

タイムソースの現在時間の取得

タイムソースはその現在時間を非常に迅速に入手することができる。なぜならば、駆動タイムソースの場合、それは、内部データ構造内に保存された値を有するからであり、さもなくば、非駆動のタイムソースの場合、それは基本オペレーティングシステムを照会することができるからである。

ウェイクアップの取扱い

２種のウェイクアップ、すなわち、ディレイおよびアラームはタイムソースにより同様に取扱われる。タイムソースはクロックからウェイクアップのためにウェイクアップ時間と呼ばれるある時間で起動するように要請を受ける。ウェイクアップは要求のクロックのタイムソースからタイムソースの逆直接関数を使用するローカルタイムベースに変換される。タイムソースはウェイクアップが遅れており、起動する必要があるかをチェックする。ウェイクアップが遅れていれば、タイムソースがそれを起動し、さもなくばタイムソースはウェイクアップリストと呼ばれる時間順のウェイクアップのリストにウェイクアップを加える。

タイムソースのウェイクアップリスト

ウェイクアップは除去されたり付加されたりするので、ウェイクアップリストが変化するときはいつでもタイムソースのGetNextTime（）方法が呼び出され、それはウェイクアップを起動する必要がある次回の時間（NEXT TIME）を計算する。タイムソースが駆動タイムソースであるならば、そのSetTime（）メンバー関数は、１以上のウェイクアップが遅れる場合はいつでもTiclkleWakeups（）を呼び出す。TiclkeWakeups（）はウェイクアップリストを通り抜けて、遅れている全てのウェイクアップを起動する。ウェイクアップが起動すると、それはリストからも除去される、。TickleWakeUps（）を次に呼び出す時間がいつかを検出するために、その次にその時間でタイマーをオフにセットするために、非駆動のタイムソースのGetNextTime（）メンバー関数を無効化することができる。タイマーがオフになると、タイムソースはTickleWakeups（）を呼び出す。

第21図はTickleWakeups（）関数の詳細論理を示す。処理はターミナル2102で始まり、直ちに、機能ブロック2104に進みここで変数CHANGEDが偽にセットされる。次に、機能ブロック2120で最初のウェイクアップが得られ、判定ブロック2160でテストが実行されてウェイクアップの要否が判定される。ウェイクアップが要であればウェイクアップが機能ブロック2150で起動され、次回のウェイクアップを機能ブロック2140で取得し、変数CHANGEDは真にセットされ、および制御は判定ブロック2160に戻って次のウェイクアップを待つ。判定ブロック2160でウェイクアップが必要でない場合は、変数CHANGEDが判定ブロック2170でテストされる。変数CHANGEDが真であれば機能ブロック2180でNEXT TIMEが取得される。もし偽であれば制御は終了のためにターミナル2190に渡される。

ソフトウェアクロックの実現

ソフトウェアクロックの現在時間の取得

クロック用の現在時間を取得するためにクロック手順は最初にそのクロック階層用のタイムソースから現在時間を取得する。次にそれは直接関数を取得し、第22図のフローチャートに示すようにソースの現在時間からクロックの現在時間に変換する。処理は2200で始まり、直ちに機能ブロック2210に進んでクロック階層用のタイムソースを取得する。次に、機能ブロック2220でタイムソースの時間が取得され、機能ブロック2230で直接関数が取得される。最後に機能ブロック2240で現在時間を結果として得るために直接関数がタイムソースの時間に対して適用され、ターミナル2250に戻る。

クロックおよびタイムソースは異なるアドレススペースに在るので、タイムソースの混在時間およびクロックの直接関数の双方は２つのアドレススペースに分割されてメモリ内に格納される。セマフォがダイレクト関数およびタイムソースの時間を分割メモリから読出

すときにそれらが変化しないよう用いられる。
ソフトクロック上のディレイのセッティング
ディレイには２つのタイプがある。１つはクロックのメンバー関数がDelayUntil（）と呼ばれ、クロックの現在時間が所定の値に到達するまでクライアントのタスクを阻止する。第２は、クロックのメンバー関数がDelayFor（）と呼ばれ、所定長さの時間が経過するまでクライアントのタスクを阻止する。第２の形態のディレイは実質的にはクロックの現在時間＋所定長さの経過時間で規定される第１の形態と同じである。
クロック上でディレイさせるために、クロックはディレイさせたい（またはディレイさせるための）時間をクロック階層用タイムソースに送る。タイムソースはクロック用逆直接関数を使用してそれをタイムソースに対してはローカルな時間に変換する（およびもし、現在時間がディレイのため（delay for）、およびディレイまででではない場合は現在時間を付加する）。その要求は（すでにウェイクアップ処理の項で述べたように）ウェイクアップリストに加えられ、ウェイクアップが起動するまで呼び出し側が阻止される。
ソフトウェアクロック上のアラームのセッティング
クロック上のアラームのセッティングは呼び出し側が阻止されない点を除いてクロック上のディレイのセッティングと同様である。
第23図はディレイおよびアラームの双方を実施する方法を示すフローチャートである。処理は2300で始まり、ここで入力時間およびレシーバー（receiver）が入力される。機能ブロック2310で直接関数が取得され、直接関数の逆を機能ブロック2320で適用し、判定ブロック2340でテストが実行されてディレイが“delay for"のタイプであるかの判定が行われる。もしそうであれば機能ブロック2342で現在時間が取得され、現在時間に時間が加えられて、制御は機能ブロック2346に移行する。
もし“delay for"でないときは次に、機能ブロック2346で時間とレシーバーとを用いてウェイクアップが作成される。次に、判定ブロック2350で別のテストが実行されて、ウェイクアップ用か判定がなされる。もしそうでないときは、機能ブロック2352でウェイクアップはウェイクアップリストに加えられ、GetNextTimeが2372に実行され、制御はターミナル2370に戻りキャンセルされる。処理はターミナル2392で完了する。もし、判定ブロック2350でウェイクアップが要の場合は、次に、ウェイクアップがディレイのときに機能ブロック2354でタスクの阻止を解除し、ウェイクアップは機能ブロック2360で取り除かれる。もしウェイクアップがアラームの場合は、次に、メッセージを機能ブロック2356で送り、機能ブロック2360でウェイクアップを除去し、処理をターミナル2392で完了する。
クロックの同期
クロックの同地はソフトウェアクロックおよびタイムソースの双方において実施される。２つのソフトクロックメンバー関数がクロックを同期させるために、クライアントにより使用される。両者はマスターにスレーブクロックを同期させるために、スレーブクロック上で呼び出される。１つのメンバー関数は、SyncTo（）と呼ばれ、クライアントがマスタークロックまたはタイムソース、速度およびオフセットを規定できるようにする。第２のSyncToKeepMyTime（）はクライアントがマスタークロックおよび速度を規定できるようにするが、オフセットは計算するので、スレーブクロックの現在時間は同期時点では値が変化しない。
SyncToKeepMyTime（）
SyncToKeepMyTime（）によりクライアントは、マスタークロックおよび速度を規定できるがオフセットは計算するので、スレーブクロックの現在時間は同期時点では変化しない。
第24図はSyncToKeepMyTime（）の詳細論理を示す。処理は2400で始まり、直ちに、判定ブロック2420に進み、マスタークロックが同じクロック階層に在るかを判定する。もしそうでなければ、機能2410でタイムソース間の以後の同期は保持状態になり、判定ブロックでテストが実行され、マスターがスレーブクロックに対して、現在、スレーブとなっているかを判定する。もし、そうであれば機能ブロック2454を経由してエラーが返され、制御はターミナル2499に進む。
もし、そうでなければ、機能ブロック2432で現在時間をスレーブクロック用に取得し、ク

ロックについての全ての情報を機能ブロック処理がターミナル2499で終了する前に機能ブロック2452で、タイムソースの間で同期が行われる。
もし、判定ブロック2420において、マスタークロックが同じタイムソースに在るときは次に判定ブロック2460で別のテストが実行され、マスターが現在スレーブクロックに対して、現在、スレーブであるかが判定される。もし、そうでなければ、機能ブロック2462でスレーブクロックについての現在時間を取得する。スレーブクロックについての関数を現在時間に基づいて計算し、機能ブロック2466で速度を供給する。機能ブロック2468で直接関数を計算し、機能ブロック2470でクロックウェイクアップを再計算する。テストを判定ブロック2472で実行し、方向が変化したかを判定する。もし、そうであれば、機能ブロック2474で全ウェイクアップを起動する。もし、そうでなければ、処理がターミナル2499で終了する前に機能ブロック2476で全スレーブクロックについての処理が繰り返される。
新しいクロック関数は以下のうように計算する。マスターに関する同期の前のスレーブクロックの関数：

$t_{slave} = a_{original} \cdot t_{master} + b_{original}$

新しい速度a_{new}でマスターに同期した後のスレーブクロックの関数

$t_{slave} = a_{new} \cdot t_{master} + b_{new}$

ここでb_{new}は$t_{newslave} = t_{slave}$等であり、

$b_{new} = t_{master} \cdot (a_s - a_{new}) + b_{original}$

SYNCTO（）
SyncTo（）によりクライアントはマスタークロックまたはタイムソース、速度、およびオフセットを規定できる。処理は第24図に示された論理にとても似ている。クロックの新しい関数はSyncTo（）の場合、次のように計算される。マスター同期する前のスレーブクロックの関数：

$t_{slave} = a_{original} \cdot t_{master} + b_{original}$

新しい速度a_{new}およびオフセットb_{new}でマスターに同期した後のスレーブクロックの関数：

$t_{slave} = a_{new} \cdot t_{master} + b_{new}$

クロックは他のどのクロックにも異なるクロック階層のクロックでさえも同期させることができる。
オーディオプレイヤーの実施
オーディオシーケンスオブジェクトはディジタル化したサウンドのサンプルセグメントを有する。それは、オーディオサンプルを検索し、格納するメンバー関数を有する。
オーディオプレイヤーはオーディオストリームを再生するので、それをスピーカー上で聞くことができる。これは、一定周期でオーディオストリームからデータを読み込み、それをオーディオ出力ポートに書き込むことによりなされる。
読取り周期はデジタルアナログ変換器（DAC）により決定される。DACはディジタルオーディオサンプルを、スピーカーに接続したオーディオ増幅器により音に変換することができるアナログの電気信号に変換する。それは１秒あたりＮ個のディジタルオーディオサンプルを変換する。NはDACのサンプル速度と呼ばれる。DACは変換を待つオーディオサンプル用の内部バッファを有する。オーディオプレイヤーは、ホストプロセッサ内で動作し、オーディオシーケンスから読出されたサンプルをDACのバッファに充填する。このバッファの容量が半分になると、DACはホストプロセッサを中断させる。オーディオプレイヤーは次にオーディオシーケンスから読取ったオーディオサンプルのバッチを再び充填する。この処理を第25図に示す。
第26図に、マスターとしてのオーディオプレイヤーを示す。これはオーディオプレイヤーのクロックを、スピーカーにより現在、再生されているサンプルの時間に常時反映させることにより、実現される。たとえば、Ｔ時点でオーディオシーケンスの開始から88200番目のサンプルがスピーカー上で再生されている場合、DACのサンプル速度およびオーディオシーケンスは共に44,100サンプル／秒であり、それ故、オーディオプレイヤーのクロックは時間で２秒になるにちがいない。これはDACにより駆動されるタイムソースにより達

成される。DACがバッファ　フルのサンプルを出力するときはいつでもホストプロセッサは中断する。ホストプロセッサは最後の中断以後、変換されたサンプル量に相当する時間までタイムソースの現在時間をインクリメントする。これは、
〈インクリメント時間〉
＝〈最後の中断以後の時間〉／〈DACサンプル速度〉により決定される。
オーディオプレイヤーはそのソフトウェアクロックをDACタイムソースに同期させる。線形関数は次に式に従う。

$$t_{audioplayerclock} = 1 \cdot t_{DAC\ time\ source} + b$$

ここでbは、シーケンスの再生が開始したとき、$t_{audioplayerclock}$となるようにセットされる。これは、クロックのSetTime関数を用いて達成される。

スレーブとしてのオーディオプレイヤー
オーディオプレイヤーは外部クロックに同期させることができる。この場合、オーディオプレイヤーは、スピーカーにより再生するサンプルが外部クロックの現在時間に対応するようにしなければならない。たとえば、所定の時点で外部クロックオブジェクトの現在時間が２秒の場合、それ故、オーディオシーケンスについて、44,100サンプル速度と想定してオーディオシーケンス中の88,200番目のサンプルが、スピーカーにより再生する時点とならなければならない。この処理は以下のように達成される。オーディオプレイヤーは、DACから、オーディオデータがさらに必要であると指示するDACからの割込み要求（interruput）を待つ。このプレイヤーはそのクロックの現在時間を調べ再生速度のスピードアップまたはスローダウン、さもなくば、正しいサンプルをスピーカーから再生できるようにオーディオシーケンス中の新しい位置へジャンプする必要があるかを決定する。オーディオシーケンスサンプルの速度をサンプル出力前のDACのサンプル速度に変換するためにはデジタル処理のアルゴリズムを用いる。
好適実施例に従って、オーディオプレイヤーの従属に関連する詳細論理を第27図に示す。処理は2700で始まり、ここでDACの割込み要求が発生するまで待機状態が入力される。DACの割込み要求が発生すると、制御は機能ブロック2710に進み、ここでクロック時間が得られる。次に、機能ブロック2720でDACに対するクロック速度が計算され、テストが2730で実行されてクロック速度が高すぎるか判定される。もし、であれば、次に、クロック速度が最高速度にセットされる。もし、そうでないときは、機能ブロック2750で所望の位置が計算される。機能ブロック2750は、その処理は問題を含んでいるので、より一層の説明を要する。それはオーディオストリームの所望の位置の計算である。この位置はプレイヤーが出力する次のサンプルを定める。この値の計算は、DACによりもたらされるディレイ故に重要である。プレイヤーはDACにサンプルを出力するとそれはDAC内部のバッファに入れられる。サンプルはDACのバッファのこれまでの全サンプルが空になるまで、スピーカーで再生されない。幸いにもこのディレイは計算することができる。それは、

DACのディレイ

＝＜DAC 内部バッファ内のサンプル数＞／DACのサンプル速度

オーディオシーケンスサンプルはクロック速度により定まる速度で出力される。これはDACのクロック位置に比例した外部クロックから導き出される。所望の位置を決定するために、DACのディレイは減じられるが、そうする前に、オーディオストリームのタイムベースへそれ変換しなければならない。これは次のようになされる。

オーディオシーケンスのディレイ(delayaudio sequence)

＝(DACのディレイ(delayDAC)・ クロック速度(clockRate)

オーディオストリームの所望の位置は次のように計算される。
所望の位置（desiredPosition）
＝〈シーケンス内の次のサンプルの位置〉
－オーディオのディレイ（$delay_{audio\ sequence}$）
次に、判定ブロック2760で所望の位置が実際の位置と比較され、それが実際の位置から離

れすぎているかが判定される。もし、そうであれば、機能ブロック2770でオーディオシーケンスは所望の位置へシークされる。いずれのケースにおいても、次に、機能ブロック2780でサンプルはオーディオストリームから読出され、クロック速度に変換され、出力バッファに書込まれる。

グラフィックプレイヤーの実施

グラフィックシーケンスオブジェクトはグラフィックオブジェクトの集積である。グラフィックシーケンス内のブラフィックオブジェクトはデジタル化されたビデオフレーム、レンダリングされたアニメーションフレーム、さもなくば時間関連の他のグラフィックオブジェクトであろう。グラフィックシーケンスは格納および検索のための個々のグラフィックをアクセスさせるメンバー関数を持つ。各グラフィックは、持続時間を持ち、それ故、シーケンスはグラフィックオブジェクト構成要素の持続時間の合計となる持続時間を持つ。グラフィックシーケンスも、また、個々のグラフィックオブジェクトに対してそのシーケンスの中で時間をマッピングするメンバー関数を与える。各グラフィックオブジェクトの持続時間を下のフレームで表した日の出のイメージのシーケンスの例を第28図に示す。

グラフィックプレイヤーはグラフィックシーケンスをそのグラフィック出力ポート、それは、グラフィックインプットポートと接続されているが、そのグラフィックポートを通して、再生する。プレイヤーは他のクロックを（マスターとして）駆動するか、または（スレーブとして）駆動される。プレイヤーはそのクロック上の時間に応じてグラフィックシーケンスの中のグラフィックオブジェクトを順序付ける。クロックは時間０からグラフィックが再生される持続時間に相当する時間までの範囲に制限される。プレイヤーは現在のフレームのクロック上の持続時間が経過するまでディレイする。プレイヤーは次に、クロック上の現在時間をチェックし、時間をマッピングし、グラフィックオブジェクトを出力ポートに書込むシーケンスの中のグラフィックオブジェクトを取得する。

プレイヤーは常時クロック上の現在時間を、出力ポートに書込みすべきグラフィックオブジェクトが選択されたときに、使用する。これによりもし、動作がスローであるときに、自動的に低下を実現できる。もし、書込みのグラフィックオブジェクトがその持続時間よりも長い時間を要する場合、１以上のフレームが効果的にスキップされる。クロックの方の変化も同様である。グラフィックプレイヤーに関連の詳細論理を第29図のフローチャートに示す。処理はターミナル2900で始まり、直ちに、機能ブロック2910に進み、次のグラフィックオブジェクトのための時間までディレイ処理を入れる。ディレイの後、クロック時間が機能ブロック2920で取得され、グラフィックオブジェクトが機能ブロック2930で出力ポートに書込まれる。テストが機能ブロック2940で実行されてストップメンバー関数が呼び出されたかが判定される。もしそうであれば、処理はターミナル2960で終了する。そうでなければテストが機能ブロック2950で実行されて、最後のフレームがおよびクロックがスレーブとなっていないかが判定される。そうでなければ制御は機能ブロック2910に進み、そうであれば処理はターミナル2960で終了する。

なお、プレイヤーのクロックが別のクロックに同期している場合、そのため、プレイヤーが明らかに停止するまで表示のループを実行し続けることに注意されたい。けれども、もし、クロックが同期していない場合、シーケンスの中の最後のグラフィックが書込まれると、ループは終了する。

MIDIプレイヤーの実施

ミディ（midi）シーケンスオブジェクトはMIDIイベントの集積である。そのメンバー関数は再生を制御し、MIDIイベントの時間上のトラックにアクセスする。MIDIシーケンスおよびMIDIトラックのフォーマットを第30図に示す。

MIDIプレイヤーは出力ポートすなわち、MIDIシンセサイザーの最終的な出力のためのMIDIドライバーのそれを含むいずれの入力ポートと接続できる出力ポートを通じてミディ（midi）シーケンスを再生する。プレイヤーはソフトウェアクロックに従ってイベントを出力するとそれらが電流になる。ホストプロセッサが遅くなるならば処理は時間の流れの前進および後進の双方で適切に実行し、減少方針を考慮に入れる。第31図はドライバーからシンセサイザーまでを通じてシーケンスを再生するMIDIプレイヤーを示す。

MIDIプレイヤーは同期の関係においてスレーブまたはマスターあるいは双方としてベースクラスのタイムベースメディアプレイヤーから継承したSyncToおよびSyncToSelfのメンバー関数を使用することによりその役目を果たすことができる。実施の詳細は変化がなく、２つの特別ケースを持つ時間上のイベントループに従う。第１の特別ケースは時間の方向切替えを取扱う。ディレイの終了の後、プレイヤーは時間をチェックして特別ケースが発生したかを判定する第１の特別ケースはディレイが早いときに生じる。これは、以下のフローチャートのメインループ中の最初の判定箇所に示されている。検出されたときには、プレイヤーは方向を切替えるので、“次”は現在の反対となる。第２の特別ケースはプロセッサのオーバロードに関する。ホストプロセッサがもはや続行できなくなると、減少方針が適用される。シーケンスの最後が到着したときか、あるいは前者のケースのイベントの最後でまたは後者のケースのイベントの最初でプレーヤーは停止する。

MIDIプレイヤーの詳細化論理のフローチャートを第32図に示す。処理はターミナル3200で始まり、直ちに機能ブロック3210に進んで次のMIDIイベントまでディレイに入る。次のMIDIイベントの実行用意ができると、機能ブロック3220でクロック時間が取得される。テストが判定ブロック3230で実行され、イベントが早いかどうかが判定される。もし、そうであれば方向が切替えられ、制御は機能ブロック3210に進む。もし、そうでなければ次に、別のテストが判定ブロック3242で実行され、イベントが遅いかどうかが判定される。もし、イベントが遅い場合、次に、減少処理がキャッチアップに適用され、制御は機能ブロック3210に戻る。もし、イベントが遅くはない場合、イベントは機能ブロック3260で出力され、テストが判定ブロック3270で実行されイベントの最後が完了し、およびクロックオブジェクトがスレーブとなっていないかが判定される。もし、イベントの最後が完了し、クロックはスレーブとなっていない場合、次に処理はターミナル3280で終了する。多くのイベントがある場合、制御は機能ブロック3210に進み、次のイベントを処理する。

表示画面

第33図から第37図までは好適実施例に従った表示画面の図である。第33図は何もないデスクトップ画面、すなわち、ユーザーがソフトウェアクロックの定義およびそれとマルチメディアオブジェクトとの同期を開始するデスクトップ画面である。第34図は好適実施例における選択クロックオブジェクト定義画面の図である。ユーザーはクロックオブジェクト定義を符号3410のメニューバーから選択する。特に、クロックメニューアイテム3420が第１図に示すマウスおよび符号3420の図形に示すようなプルダウンメニューを使用して選択される。第35図は、好適実施例において、第34図に示されるメニュー選択を使用する種々のクロックオブジェクトおよび定義されたマルチメディアオブジェクトの図である。

実際のマルチメディアプレゼンテーションにおいて現れるような、スライドバーオブジェクト3510、ジョブ／シャトルコントローラ3520、クロックオブジェクト3530およびアニメーションマルチメディアオブジェクト3540が全て示されている。

第36図は、好適実施例における互いにリンクした種々のクロックオブジェクトおよびマルチメディアオブジェクトを示す。カーソルを線分のようなゴムバンドの幾何学図形にカーソルを合わせ、クロックオブジェクト3610を別のクロック3620に結合するかまたはマルチメディアオブジェクト3630および3640に結合してリンクが作成される。

第37図は好適実施例に従ってオーディオオブジェクトと同期するビジュアルオブジェクトを示す図である。ビジュアルクロックオブジェクト3710がオーディオクロックオブジェクト3720に同期してアニメーションマルチメディアオブジェクト3730により示されるミュージックおよび表示の関連マルチメディアプレゼンテーションを制御する。

【図２】

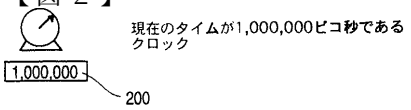


FIG. 2

【図３】

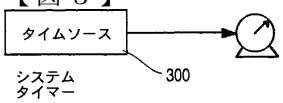


FIG. 3

【図４】

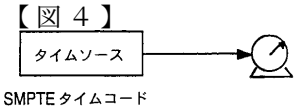


FIG. 4

【図５】

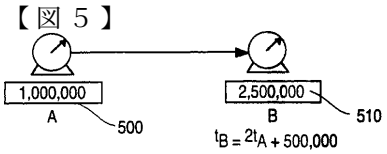


FIG. 5

【図１】

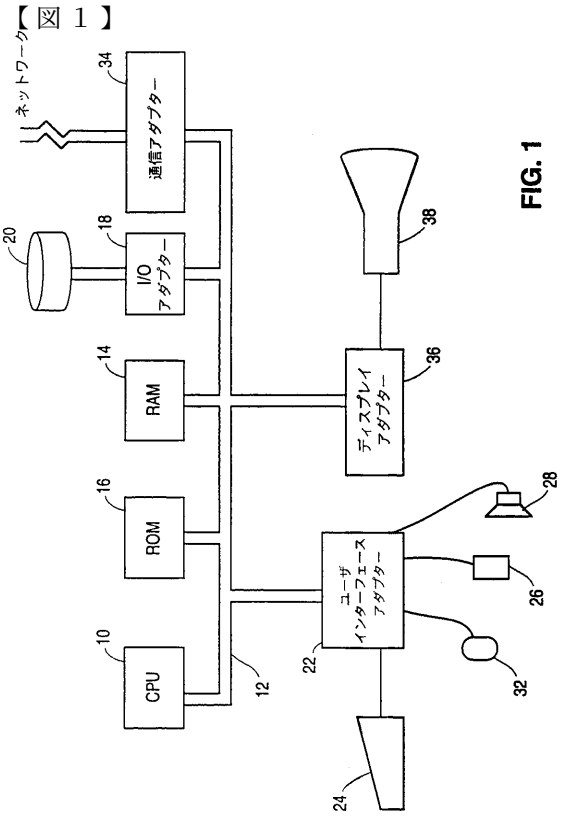


FIG. 1

【図６】

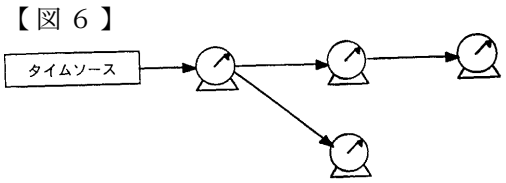


FIG. 6

【図７】

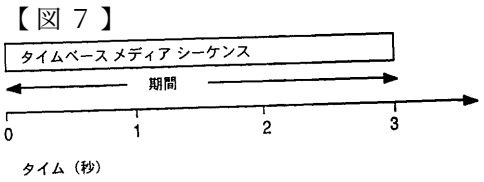


FIG. 7

【図８】

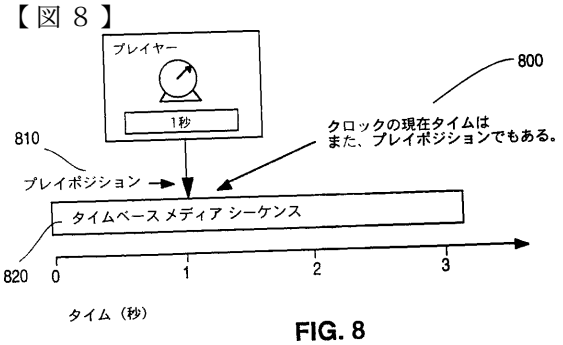


FIG. 8

【図９】

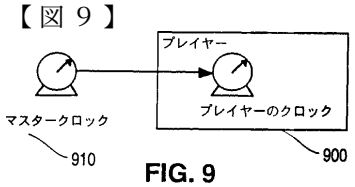


FIG. 9

【図１０】

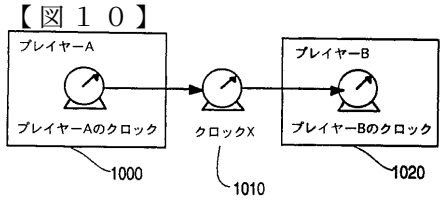
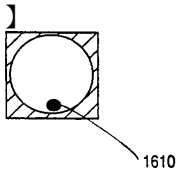


FIG. 10

【図１６】

FIG. 16



【図１８】

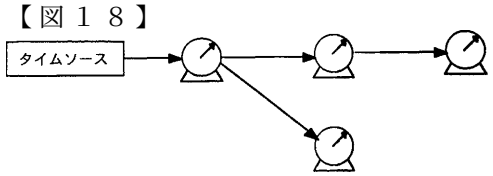


FIG. 18

【図１１】

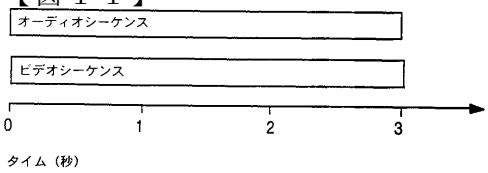


FIG. 11

【図１２】

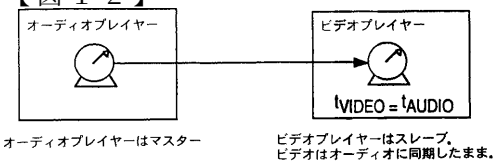


FIG. 12

【図１７Ａ】

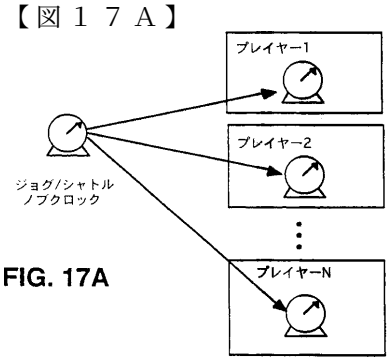


FIG. 17A

【図１９】

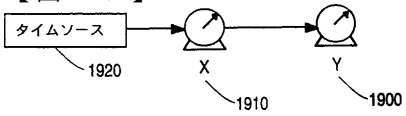


FIG. 19

【図２５】

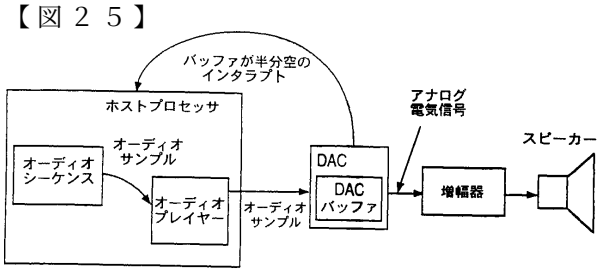


FIG. 25

【図２８】

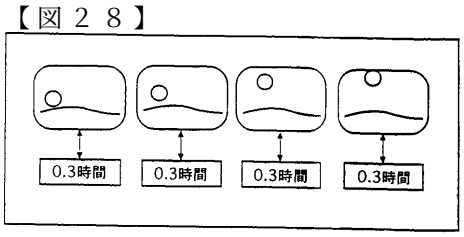


FIG. 28

【図１３】

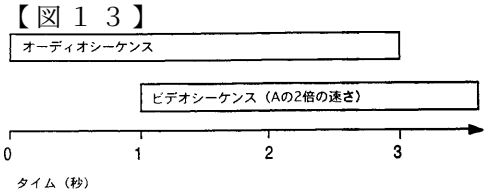


FIG. 13

【図１４】

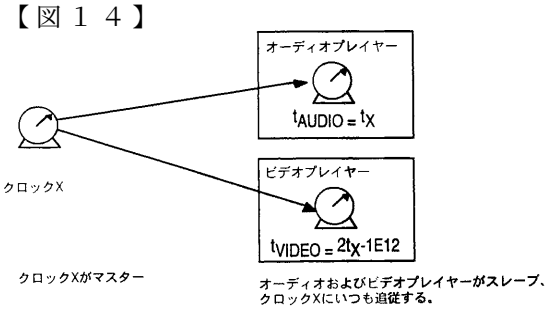


FIG. 14

【図１５】

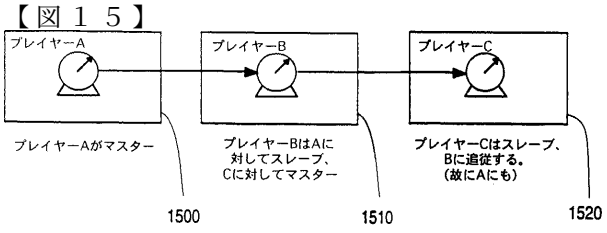


FIG. 15

【図１７Ｂ】

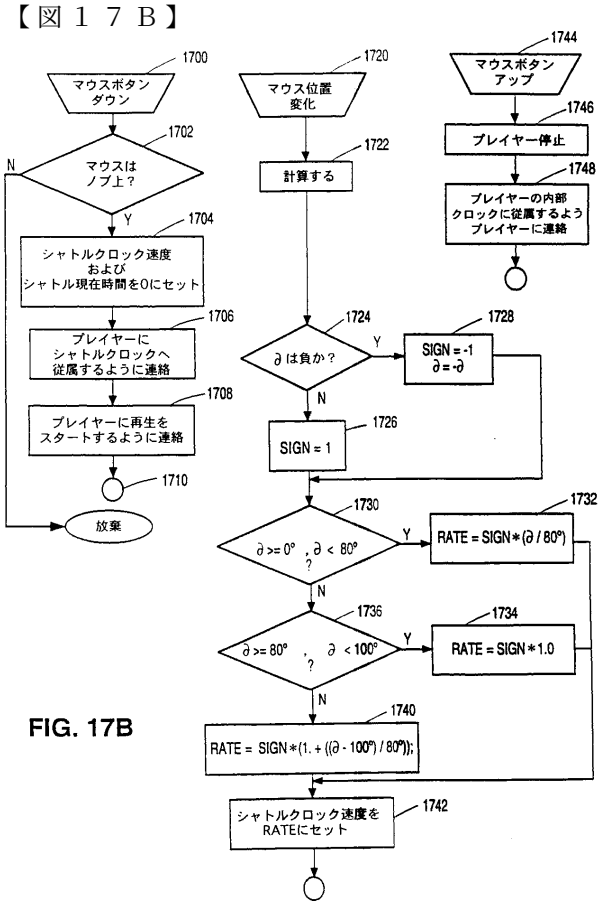


FIG. 17B

【図１７Ｃ】

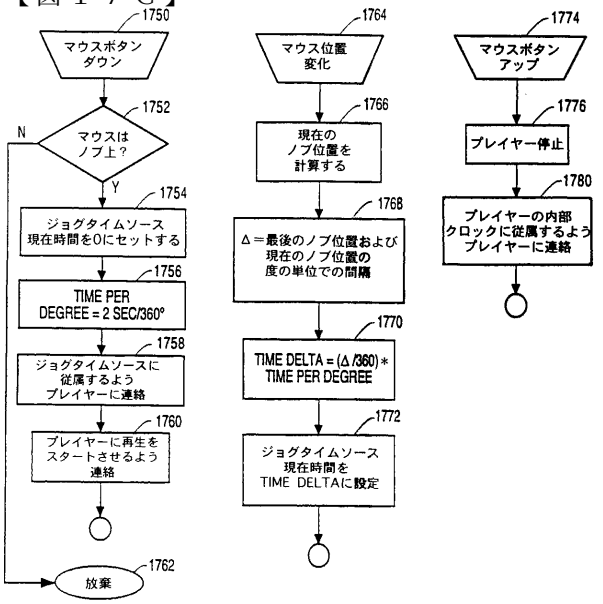


FIG. 17C

【図２０】

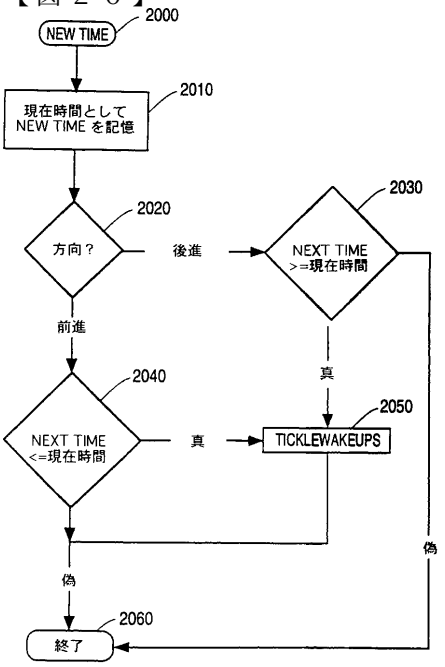


FIG. 20

【図２１】

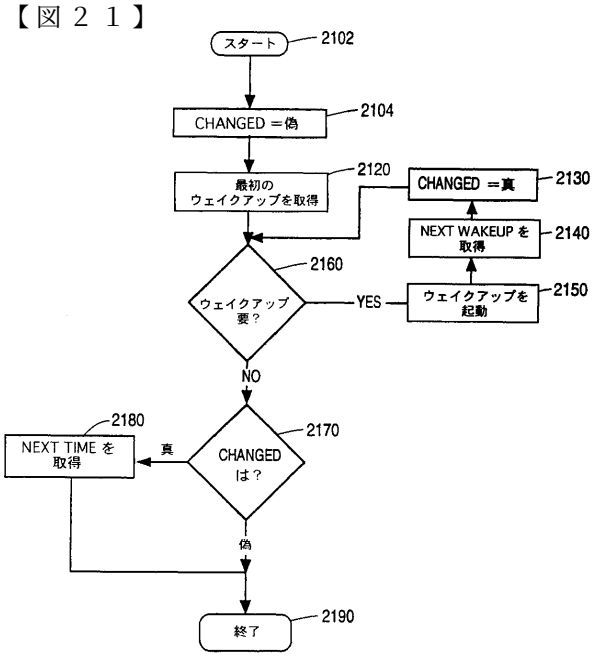


FIG. 21

【図２２】

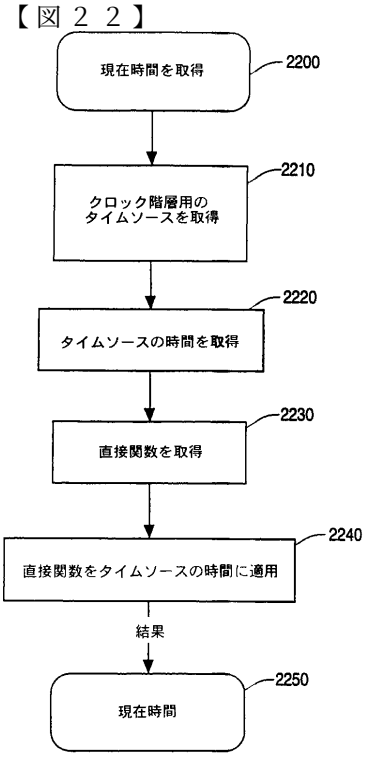


FIG. 22

【図３１】

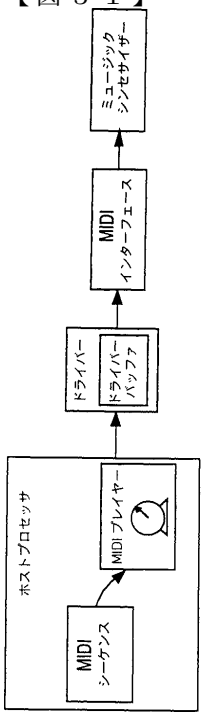


FIG. 31

【図２３】

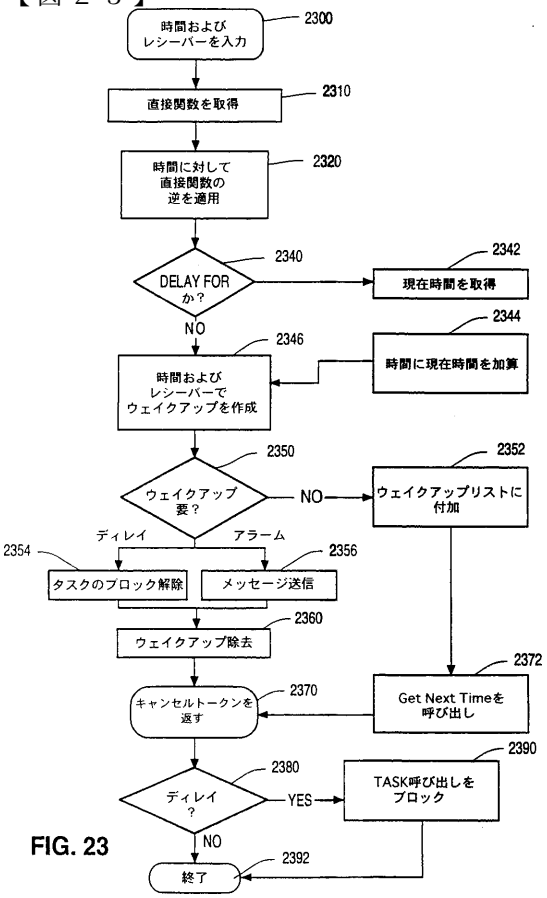


FIG. 23

【図２４Ａ】

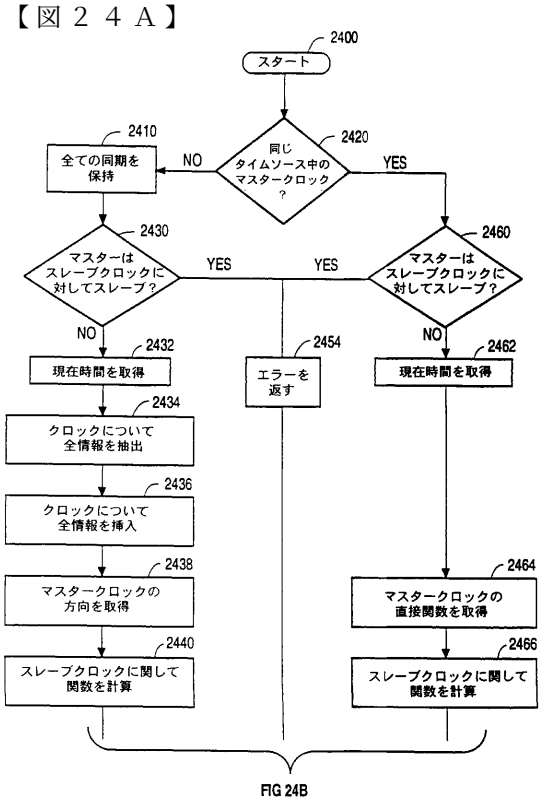


FIG. 24A

【図２４Ｂ】

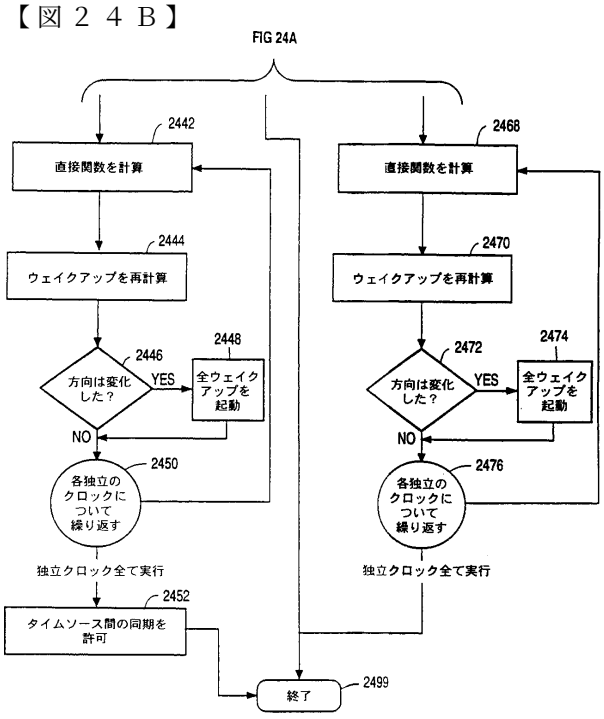


FIG. 24B

【図２６】

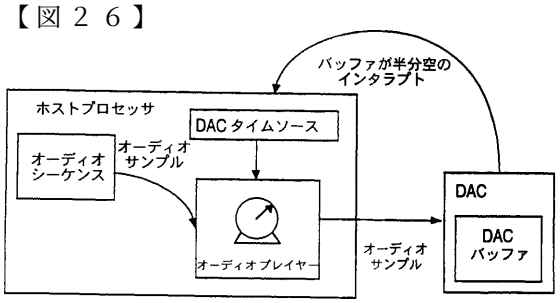


FIG. 26

【図３０】

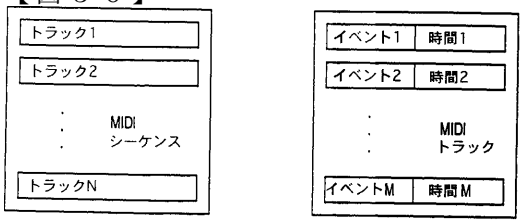


FIG. 30

【図２７】

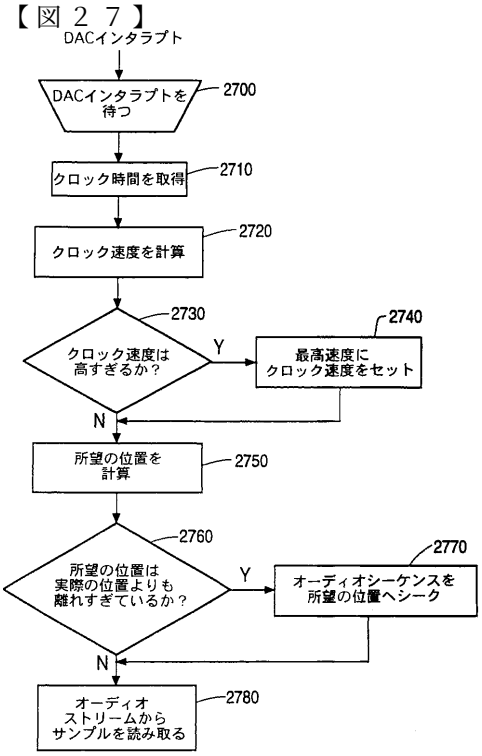


FIG. 27

【図２９】

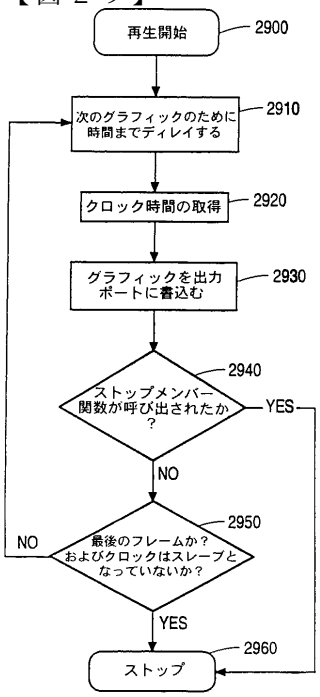


FIG. 29

【図３２】

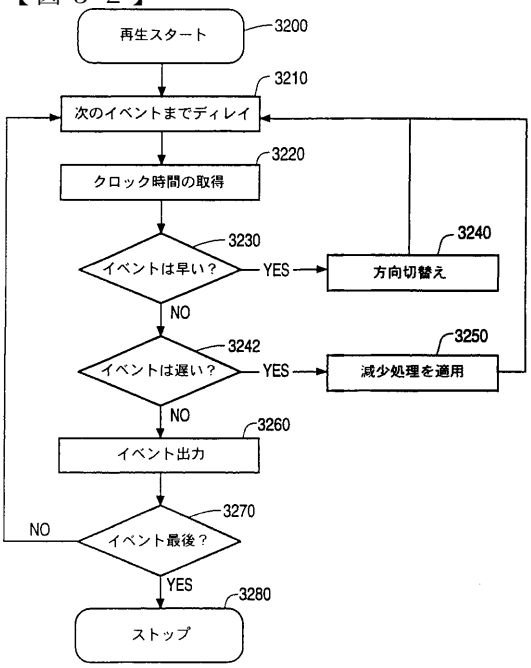


FIG. 32

【図３３】

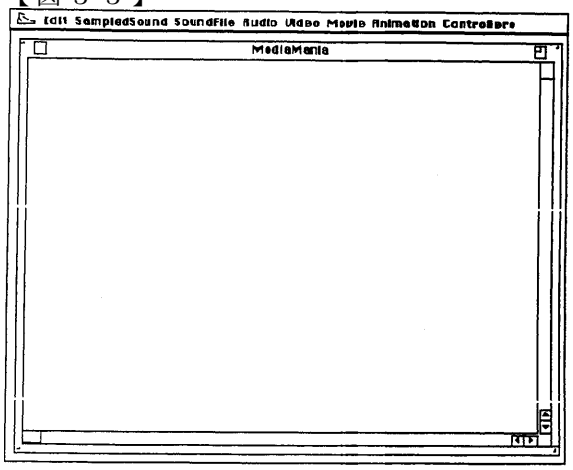


FIG. 33

【図３４】

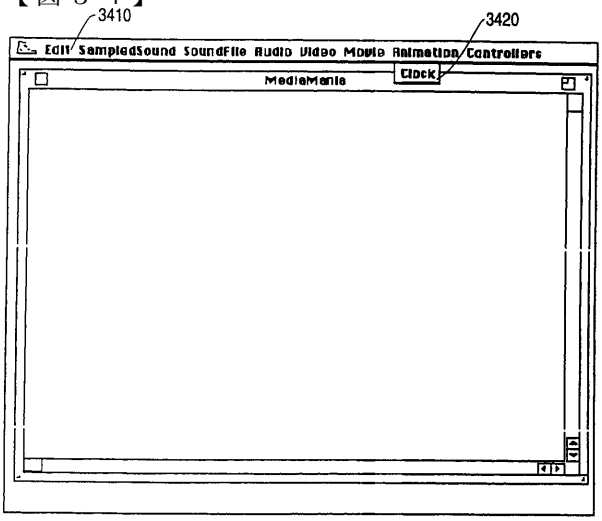


FIG. 34

【図３５】

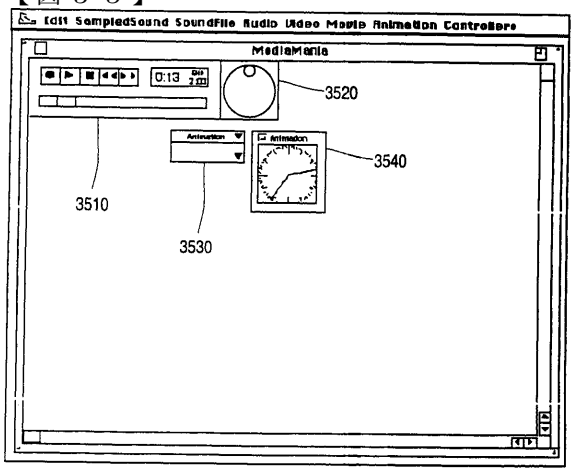


FIG. 35

【図３６】

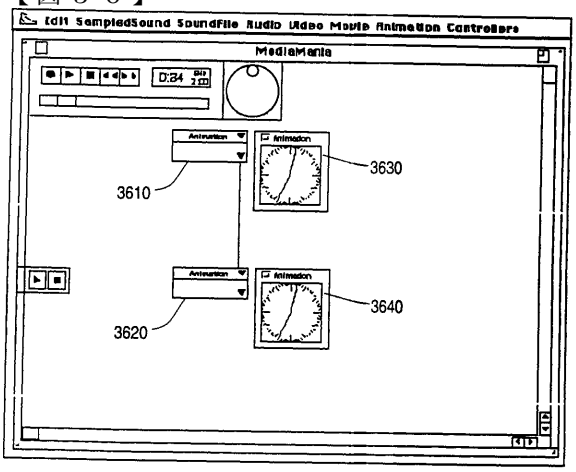


FIG. 36

【図３７】

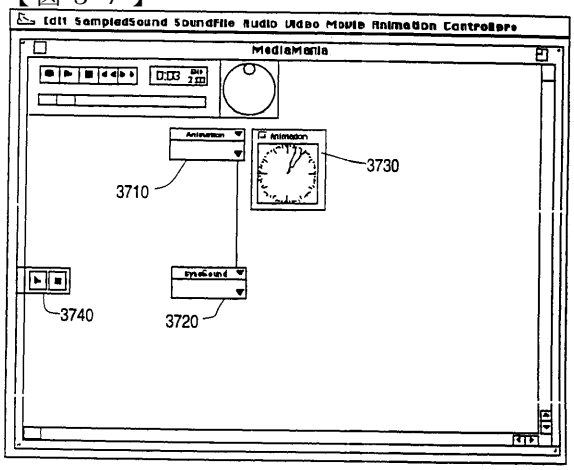

Edit SampledSound SoundFile Audio Video Movie Animation Controllers
MediaMania
Animation
Animation
3730
3710
3740
3720

FIG. 37

フロントページの続き

(72)発明者 デンマン，マチュー，エル.
アメリカ合衆国 ９５０３０ カリフォルニア州 ロス ゲイトス ミラー カットオフ ロード ２４５６５
(72)発明者 チンデル，ジェイムズ，マイケル
アメリカ合衆国 ９４０２０ カリフォルニア州 ラ ホンダ ピー．オー．ボックス ２４３
(72)発明者 トブラス，ジョン，チルコット
アメリカ合衆国 ９４０８７ カリフォルニア州 サニーベイル アリソン ウエイ ８１９
(72)発明者 ディルツ，マイケル，ラッセル
アメリカ合衆国 ９５０７０ カリフォルニア州 サラトガ ベイラー アベニュ １８４３０
(72)発明者 エデルマン，ブラッドレイ，アラン
アメリカ合衆国 ９５０１４ カリフォルニア州 クパチーノ ホームステッド ロード ２０８００ ナンバー ３２イー

審査官 間野 裕一

(56)参考文献 特開昭６２－２３９２３３（ＪＰ，Ａ）
特開平０４－０３０３８４（ＪＰ，Ａ）
特開平２－２０８６９８（ＪＰ，Ａ）
Gerold Blakowski,et al.,Tool support for the synchronization and presentation of distributed multimedia,Computer Communicatyions,1992年12月，Vol.15,No.10,pages611-618,London
F.Horn,et al.,On Programming and Supporting Multimedia Object Synchronization,The Computer Journal,1993年,Vol.36,No.1,pages4-18
坂上秀和他，マルチメディアオブジェクト構成モデルとその実現方式，情報処理学会研究報告，社団法人情報処理学会，1992年12月4日，第92巻，第98号，第31－36頁，92-OS-57-5
高橋信之，マックと音楽と映像を身近にするために ＭＩＤＩインターフェース大研究，マイコンＢＡＳＩＣマガジン，電波新聞社，1989年8月1日，第8巻，第8号，8月号別冊付録，第３４－３７頁，（ＬＥＴ’Ｓ ＰＬＡＹ！ コンピュータ・ミュージック）

(58)調査した分野(Int.Cl.^{7}，ＤＢ名)
G06F 9/46 - 9/54
G04G 7/00
G04G 9/00
G06F 3/14
H04N 5/265