(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第3781454号
(P3781454)
(45)発行日　平成18年5月31日(2006.5.31)　(24)登録日　平成18年3月17日(2006.3.17)

(51)Int.Cl.			F I		
G06F	9/54	(2006.01)	G06F	9/46	480B
H04N	7/26	(2006.01)	H04N	7/13	Z
H04N	7/173	(2006.01)	H04N	7/173	630

請求項の数 25　(全 24 頁)

(21)出願番号　特願平7-135568
(22)出願日　平成7年4月26日(1995.4.26)
(65)公開番号　特開平8-46935
(43)公開日　平成8年2月16日(1996.2.16)
審査請求日　平成14年4月18日(2002.4.18)
(31)優先権主張番号　234144
(32)優先日　平成6年4月28日(1994.4.28)
(33)優先権主張国　米国(US)

(73)特許権者　500036093
オープン　ティーヴィー　インコーポレイテッド
アメリカ合衆国　９４１１１　カリフォルニア州　サンフランシスコ　サクラメント　ストリート　２７５
(74)代理人　100059959
弁理士　中村　稔
(74)代理人　100067013
弁理士　大塚　文昭
(74)代理人　100082005
弁理士　熊倉　禎男
(74)代理人　100065189
弁理士　宍戸　嘉一

最終頁に続く

(54)【発明の名称】オーディオ・ビデオ対話信号を処理する方法と装置

(57)【特許請求の範囲】
【請求項１】
複数のモジュールを含むパケットサービスを受信するオーディオ・ビデオ対話型受信機であって、各モジュールは名前を有し、時間多重化伝送ユニットを含み、各伝送ユニットはモジュールの名前を含むヘッダパケットを含むオーディオ・ビデオ対話型受信機におけるモジュールを処理する方法において、
前記受信機内で実行するアプリケーションからのリクエストを受信し、名前により識別されたモジュールに関するプロセスを実行して、受信された名前とリクエストされたプロセスを含むエントリをエントリリストに挿入するステップと、
前記受信されたパケットサービス内でヘッダパケットを検出するステップと、
検出されたヘッダパケット内の名前が前記リスト内のあるエントリの名前と一致するかどうかを判定するステップと、
一致するエントリが存在する場合は、一致するエントリ内の前記要求されたプロセスを実行するステップと、
前記プロセスが実行されたことを示す前記アプリケーションへのメッセージを送信するステップと
を具備したことを特徴とするモジュールを処理する方法。
【請求項２】
請求項１において、前記受信するステップは、完全なモジュールを処理するためのリクエストを受信するステップを含み、

前記送信するステップは、前記完全なモジュールが処理されるとき前記リストから前記一致するエントリを除去するステップを含むことを特徴とする方法。

【請求項３】

請求項１において、前記要求されたプロセスの実行の開始に続いて、前記アプリケーションへの前記メッセージを送信するのに先立って、前記方法が、前記プロセスの実行を停止するためのリクエストを受信するステップと、

前記受信された名前とリクエストされたプロセスを含むエントリを前記エントリリストから除去することを停止するステップと

をさらに備えたことを特徴とする方法。

【請求項４】

請求項１において、各伝送ユニットは各ヘッダパケットと関連する複数のデータパケットを具備し、

前記受信するステップは、前記モジュールをメモリにロードすることにより前記モジュールを実行するためのリクエストを受信するステップと、前記モジュールを含むように前記メモリ内のバッファを割り振るステップとを具備し、

前記実行するステップは、前記割り振られたバッファ内の各ロケーションに前記検出されたヘッダパケットと関連する各データパケットを貯えるステップと、前記モジュール内の各伝送ユニット内の各データパケットが前記割り振られたバッファ内に格納されたとき、前記リスト内の一致するエントリを除去するステップとを具備した

ことを特徴とする方法。

【請求項５】

請求項４において、伝送ユニット内の各ヘッダパケットと関連する前記複数のデータパケットは、前記モジュールの開始から個々のオフセット位置に位置する前記モジュール内のデータを含み、

前記ヘッダパケットは前記関連する複数のデータパケット内に前記データのオフセット位置をさらに含み、

前記格納するステップは、前記割り振られたバッファのオフセット位置であって、前記ヘッダパケット内の前記オフセット位置に等しいオフセット位置に、前記関連する複数のデータパケットを格納するステップを具備した

ことを特徴とする方法。

【請求項６】

請求項５において、前記パケットサービスは、前記モジュールの前記開始から、順次オフセット位置におかれたモジュールデータを含む連続的な時間多重された伝送ユニットにより運ばれるモジュールを、連続的かつ繰り返し供給し、

前記検出するステップは、前記検出されたヘッダパケットのオフセット位置を指す書き込みポインタを設定するステップをさらに具備し、

前記実行するステップは、モジュールをロードするリクエストが開始ロード位置として受信された後、最初に検出されたヘッダパケット内に前記オフセット位置を格納するステップをさらに具備し、

前記格納するステップは、

前記書き込みポインタにより指された位置の検出されたヘッダパケットと関連する各データパケットを格納するステップと、

そして、前記データパケットが格納された後、前記割り振られたバッファ内の次の順次位置を指す前記書き込みポインタを更新するステップと、

その後、前記書き込みポインタと前記格納された開始ロード位置とを比較し、それらが等しいとき、前記除去するステップを実行するステップと

を具備したことを特徴とする方法。

【請求項７】

請求項６において、前記検出するステップは、前記設定するステップの前に、

前記書き込みポインタと、前記検出されたヘッダパケット内の前記オフセット位置とを

比較するステップと、
それらが等しくないとき、前記検出されたヘッダパケット内の前記オフセット位置と前記開始ロード位置を交換するステップと
をさらに具備したことを特徴とする方法。
【請求項８】
請求項４において、各モジュールは埋め込みエラー検出コードをさらに含み、
前記格納するステップは、前記埋め込みエラー検出コードを前記割り振られたバッファ内に格納し、
前記送信するステップは、
前記埋め込みエラー検出コードを用いてエラーを検出するため、前記割り振られたバッファ内の前記データをチェックするステップと、
前記チェックするステップでエラーが検出された場合、エラーメッセージを送信するステップと、
エラーが検出されない場合は、モジュールにロードされたメッセージを送信するステップと
をさらに具備したことを特徴とする方法。
【請求項９】
請求項８において、前記埋め込みエラー検出コードは埋め込みＣＲＣコードであり、
前記チェックするステップは、前記割り振られたバッファ内の前記データに対するＣＲＣ値を計算し、計算されたＣＲＣ値と前記埋め込みＣＲＣコードとを比較し、それらが異なる場合、エラーを検出することを特徴とする方法。
【請求項１０】
請求項４において、各伝送ユニットは複数のデータパケットを具備し、
前記受信するステップは、
前記検出されたヘッダパケットと関連する各データパケットを格納するステップと、
格納された伝送ユニットと関連する各データパケットにエラーがないか否かを判定するステップと、
前記伝送ユニットのデータパケットにエラーがある場合は、伝送ユニット全体を破棄するステップと
を具備したことを特徴とする方法。
【請求項１１】
請求項１において、前記受信するステップは、前記パケットサービス内のモジュールをスパイするリクエストを受信するステップを有することを特徴とする方法。
【請求項１２】
複数のモジュールを含むパケットデータのソースであり、各モジュールは名前を持ち、時間多重化伝送ユニットを含み、各伝送ユニットはモジュールの名前を含むヘッダパケットを含み、複数のモジュールを含むパケットデータストリームのソースと、
複数のエントリを含むリクエストキューを格納するためのメモリであって、前記複数のエントリが処理リクエストとモジュールの名前を含むメモリと、モジュール名前を含むロケーションを含むヘッダパケットバッファと、
前記メモリに接続されたリクエスト処理回路であって、名前で識別されるモジュールに対する処理を実行するリクエストを受信し、前記識別されたモジュールの名前と、前記リクエストされた処理とを含む前記リクエストキューにエントリを格納するリクエスト処理回路と、
前記データストリームソースと前記メモリの間に接続されたヘッダパケットメモリ格納コントローラであって、ヘッダパケットが前記データストリームに現れたとき、前記ヘッダパケットを前記ヘッダパケットバッファに格納し、ヘッダパケット受信信号を生成するヘッダパケットメモリ格納コントローラと、
前記ヘッダパケット受信信号に応答するモジュール処理回路であって、前記ヘッダパケットバッファから前記モジュール名を読み出し、前記リクエストバッファ内の各エントリ

内の名前と前記モジュール名を比較し、前記モジュール名前がエントリ名前と一致した場合、前記一致リクエストキューエントリ内のリクエストされた処理を行い、その後メッセージを送信するモジュール処理回路と
を具備したことを特徴とするオーディオ対話型受信機。
【請求項１３】
請求項１２において、前記モジュール処理回路はプロセッサを備え、前記ヘッダパケット受信信号は前記プロセッサに対する割り込み信号であることを特徴とする受信機。
【請求項１４】
請求項１２において、前記ヘッダパケットメモリ格納コントローラはダイレクト・メモリ・アクセス・コントローラであることを特徴とする受信機。
【請求項１５】
請求項１２において、前記リクエスト処理回路は完全なモジュールを処理するためのリクエストを受信し、
前記モジュール処理回路は前記完全なモジュールのリクエストされた処理を行った後、前記リクエストキューエントリを除去する
ことを特徴とする受信機。
【請求項１６】
請求項１５において、前記メモリバッファはモジュールバッファをさらに格納し、
前記データストリームソースは、伝送ユニット内に含まれた前記ヘッダパケットと関連する複数のデータパケットをそれぞれ含む前記伝送ユニットを生成し、　前記受信機は前記データストリームソースと前記メモリの間に接続され、データパケットが前記データストリームに現れたとき、伝送ユニットからの前記複数のデータパケットを選択的にモジュールバッファに格納し、その後イネーブル信号に応答してデータ完了信号を生成するモジュールメモリ格納コントローラをさらに備え、
前記リクエスト処理回路は名前により識別された完全なモジュールをロードするための処理リクエストを受信し、前記識別された名前と前記ロードリクエストとを含む前記リクエストキュー内にエントリを格納し、
前記モジュール処理回路は前記モジュール名前とエントリ名前が一致した場合に、前記イネーブル信号を生成し、前記完全モジュールを前記モジュールバッファにロードした後、前記データ完了信号に応答して、前記リクエストキューエントリを除去する
ことを特徴とする受信機。
【請求項１７】
請求項１６において、前記データストリームソースは、埋め込みエラー検出コードを含む各モジュールを生成し、
前記モジュール処理回路は、前記完全なモジュールが前記モジュールバッファ内にロードされた後、前記埋め込みエラー検出コードを用いて、前記モジュールバッファ内の前記データをエラーチェックし、エラーが検出された場合は、エラーメッセージを送信し、そうでない場合は、モジュールロードメッセージを送信する
ことを特徴とする受信機。
【請求項１８】
請求項１７において、前記埋め込みエラー検出コードはＣＲＣコードであり、前記モジュール処理回路は、前記モジュールバッファ内の前記データに対するＣＲＣ値を計算し、計算されたＣＲＣ値と前記埋め込みＣＲＣコードとを比較し、それらが異なる場合はエラーを検出することを特徴とする受信機。
【請求項１９】
請求項１６において、前記データストリームソースは、前記モジュールの開始から順次オフセット位置に位置するモジュールデータを含む連続的な伝送ユニットを生成し、前記関連するデータパケットの前記データの前記オフセット位置をさらに含む前記ヘッダパケットを生成し、
前記メモリの前記ヘッダパケットバッファは、前記オフセット位置を含む位置をさらに

含み、
前記モジュール処理回路は、前記ヘッダパケットバッファから前記オフセット位置を抽出し、そのオフセット位置を前記モジュールメモリ格納コントローラに供給し、
前記モジュールメモリ格納コントローラは、前記モジュール処理回路からの前記オフセット位置で書き込みポインタを初期化し、該書き込みポインタにより指された位置にデータパケットをそれぞれロードし、その後、伝送ユニットの全てのデータパケットがロードされるまで、前記モジュールバッファ内の次の順次位置を指すように前記書き込みポインタを更新し、その後、前記データ完了信号を生成する
ことを特徴とする受信機。
【請求項２０】
請求項１９において、前記データストリームソースは、モジュールを連続的かつ繰り返し生成し、
前記メモリは開始ロードアドレス位置をさらに含み、
前記モジュール処理回路は、最初の一致を引き起こす前記ヘッダパケットがロードされた後、前記ヘッダパケットバッファから前記オフセット位置を抽出し、前記開始ロードアドレス位置に前記オフセット位置を格納し、前記モジュールメモリ格納コントローラからの前記データ完了信号に応答して、前記モジュールメモリ格納コントローラからの前記書き込みポインタを前記開始ロードアドレス位置と比較し、それらが等しいとき前記メッセージを送信する
ことを特徴とする受信機。
【請求項２１】
請求項２０において、前記モジュール処理回路は、前記ヘッダパケット受信信号に応答して、前記書き込みポインタと前記オフセット位置を比較し、それらが等しくないとき、前記オフセット位置で前記開始ロードアドレス位置を交換することを特徴とする受信機。
【請求項２２】
請求項１６において、前記モジュール処理回路は、プロセッサを備え、前記データ完了信号は前記プロセッサへの割り込み信号であることを特徴とする受信機。
【請求項２３】
請求項１６において、前記モジュールメモリ格納コントローラはダイレクト・メモリ・アクセス・コントローラであることを特徴とする受信機。
【請求項２４】
請求項１２において、前記リクエスト処理回路は、名前により識別されたモジュールの処理を停止するためのリクエストを受信し、前記識別されたモジュールの名前と、停止するようリクエストされた前記処理とを含む前記リクエストキュー内の前記エントリを除去することを特徴とする受信機。
【請求項２５】
請求項１２において、前記ヘッダパケットメモリ格納コントローラは、イネーブル信号に応答してヘッダパケットを選択的に格納し、
前記リクエスト処理回路は、エントリが前記リクエストキュー内に格納された後、前記イネーブル信号を生成し、
前記モジュール処理回路は、前記リクエストキュー内にエントリがない場合、前記処理がなされた後、前記イネーブル信号をさらに除去することを特徴とする受信機。
【発明の詳細な説明】
【０００１】
【産業上の利用分野】
本発明は、オーディオ・ビデオ対話（ＡＶＩ：ａｕｄｉｏ　ｖｉｄｅｏ　ｉｎｔｅｒａｃｔｉｖｅ）信号を受信して処理する方法と装置に関する。
【０００２】
【発明の背景】
対話型テレビジョンシステムが提案されている。対話型テレビジョンシステムでは、テレ

ビジョン受信機は、放送機器によりプログラム可能であり、ユーザにより入力されたデータに応答し、放送されたビデオ上にオーバレイされたオンスクリーン・グラフィック・ディスプレイを生成し、放送されたオーディオと合成された音声を生成し、かつ／あるいは、放送機器、または他の外部のデータ処理サービスとデータを交換する処理装置を含む。そのようなシステムでは、放送ロケーションはコンピュータシステムを含み、このコンピュータシステムは、対話型アプリケーションプログラム情報を生成し、追加のコンポーネントとしての対話型アプリケーションプログラム情報と、関連するテレビジョン信号のビデオおよびオーディオ・コンポーネントとを合成する。テレビジョン受信器の処理装置は、対話型アプリケーションプログラム情報を放送機器から受信し、その情報により表された対話型アプリケーションプログラムを実行し、テレビジョンの映像とオーディオとに合成されるグラフィックスと音声を生成し、リモートコントロールユニットを介して受信されるユーザ入力を処理する。

【０００３】

提案されているＡＶＩシステムでは、放送機器からのＡＶＩ信号はパケットデータストリームの形態で放送され、複数の時間多重されたパケットサービスを含む。各パケットサービスはＡＶＩ信号の異なる信号コンポーネントを有する。例えば、あるサービスはビデオコンポーネントを有し、あるサービスはオーディオコンポーネントを有し、あるサービスは対話型アプリケーションプログラム情報を有する。また、あるサービスはステレオとＳＡＰオーディオチャンネル、および／または、クローズドキャプション情報等を有する。さらに、パケットデータストリームの中には、２つ以上のＡＶＩプログラムに対するコンポーネントを有するパケットサービスを含むことができるものもある。各パケットサービスは、パケットサービスに関連する一意のサービスコンポーネント識別子（ＳＣＩＤ）を有し、そのパケットサービス内のパケットは、それぞれ、そのサービス識別子を含む。米国特許出願第２３４，７７３号（発明の名称：ＡＰＰＡＲＡＴＵＳ　ＡＮＤ　ＭＥＴＨＯＤ　ＦＯＲ　ＦＯＲＭＵＬＡＴＩＮＧ　ＡＮ　ＩＮＴＥＲＡＣＴＩＶＥ　ＴＶ　ＳＩＧＮＡＬ　を参照されたい。この米国特許出願番号を付して、このようなパケットデータストリームがどのように形成されるかを詳細に説明する説明の一部とする。

【０００４】

さらに、提案されたＡＶＩシステムでは、１つのパケットサービスがプログラムガイドを有し、予め定めたサービス識別子を含む。プログラムガイドパケットサービスにより運ばれるデータは、ＡＶＩプログラムのコンポーネントと、それらのコンポーネントを運ぶパケットサービスのサービス識別子とを関係付ける。このデータを用いて、望ましいＡＶＩプログラムのコンポーネントを運ぶパケットサービスを、パケットストリームから取り出すことができる。

【０００５】

ＡＶＩ信号パケットデータストリーム内のコンポーネントは、複数のパケットからなる１以上の伝送ユニットにより運ばれる。任意の伝送ユニット内の最初のパケットはヘッダパケットであり、残りのパケットは関連するデータパケットである。ヘッダパケットは後続データについての情報を含み、関連するデータパケットはコンポーネントのその部分を構成するデータを運ぶ。異なる伝送ユニットは異なる数のデータパケットを含み、モジュールの伝送ユニットへの分割は、所望の時間に、視聴者のロケーションに完全なモジュールを供給するために必要なタイミングにより影響されるか、あるいは、他のリアルタイムに考慮すべきことにより影響される。

【０００６】

対話型アプリケーションプログラム情報コンポーネントは、１つ以上の（実行可能なコードを含む）コードモジュールと、可能な場合は、１つ以上のデータモジュールと、ディレクトリモジュールとによりなる。このディレクトリモジュールは対話型アプリケーションプログラムのコンポーネントを構成するコードモジュールとデータモジュールを記述するデータを含む。これらのモジュールは、アプリケーション・プログラム・コンポーネントのフロー内で連続して繰り返される。既に述べたが、モジュールは伝送ユニットにより運

ばれる。伝送ユニット内のヘッダパケットは、さらに、後続のデータパケット内のデータが属するモジュール内のロケーションを含む。

【０００７】

ＡＶＩ受信機の処理装置は、最初、データフローからディレクトリモジュールを取り出し、そのディレクトリに含まれた情報を利用して、どのコードモジュールを最初に実行するべきかを決定する。そして、そのコードモジュールがデータフローから取り出され、メモリ内の周知のロケーションにロードされる。最初のコードモジュールがメモリ内に完全にロードされると、処理装置がそのコードモジュールを実行し始める。実行している間、そのコードモジュールは、ディレクトリモジュール内で識別されたデータモジュールからのデータを要求することができる。そして、これらのデータモジュールが取り出され、メモリ内にロードされる。データモジュールがメモリ内に完全にロードされると、要求されているコードモジュールが通知され、引き続きそのデータを処理する。１つのコードモジュールを後続のコードモジュールにチェインすることも可能である。このような場合、現コードモジュールは、ディレクトリモジュールにリストアップされた新しいコードモジュールにチェインする要求を発行し、そのメモリ空間は解放される。そして、要求されたコードモジュールはデータフローから取り出され、メモリにロードされる。要求されたコードモジュールがメモリに完全にロードされると、実行される。他の機能も可能であり、次に説明する。

【０００８】

メモリ容量が充分な場合は、２つ以上のコードモジュールを、メモリの個々の部分に同時に貯えることができる。そのような場合、モジュールからモジュールにチェイニングすると、必然的に、新たなモジュールの実行を開始するステップと、新しいモジュールがフローからロードされるまで待機する要件を除くステップとを伴う。メモリに２つ以上のデータモジュールをロードすることも可能である。データモジュールの変更はコードモジュールよりも頻繁なので、データモジュールがコードモジュールにより要求されると、データモジュールがデータフローから取り出されることが考えられる。

【０００９】

提案されているＡＶＩ受信機の処理装置は、大容量記憶装置や、プリロードされたＡＶＩプログラムを持つ必要がない。代わりに、対話型アプリケーション・プログラム・パケット・サービスで運ばれるモジュールが連続して繰り返されるので、必要なときは、どのモジュールもデータフローから取り出すことができ、大容量記憶装置は必要でない。しかしながら、このため、ＡＶＩ受信機の処理装置は、要求されたモジュールがデータフローに存在するか否かを連続的に監視しなければならない。データフローを効率よく監視し、できるだけ速やかに所望のモジュールを処理する方法と装置が要望されている。

【００１０】

【課題を解決するための手段】

本発明の原理によれば、オーディオ・ビデオ対話型（ＡＶＩ）受信機は、複数のモジュールを含むパケットサービスを受信する。各モジュールは名前を持ち、時間多重された伝送ユニットを含む。各伝送ユニットはヘッダパケットを含み、ヘッダパケットはモジュールの名前を含む。このような受信機でモジュールを処理する方法は次のステップを含む。最初に、名前により識別されたモジュールに対するプロセスを実行する要求が受信され、受信されたモジュール名と要求されたプロセスを含むエントリがエントリリストに挿入される。そして、パケットサービスのヘッダパケットが検出される。検出されたヘッダパケット内のモジュール名がリストのエントリの名前と一致した場合、パケット処理回路が一致するエントリでプロセスを実行する。最後に、プロセスがパフォームされたことを示すメッセージが送信される。

【００１１】

本発明の原理による装置は、複数のモジュールを含むパケットデータストリームのソースを含む。各モジュールは名前を持ち、時間多重された伝送ユニットを含む。各伝送ユニットはモジュールの名前を含むヘッダパケットを含む。メモリは複数のエントリを含む要求

キューを貯える。各エントリは処理要求とモジュールの名前を含む。メモリはヘッダパケットバッファも貯えるッダパケットバッファはモジュール名前を含むロケーションを含む。要求処理回路は名前により識別されたモジュールに対する処理を実行する要求を受信し、識別されたモジュールの名前と、要求された処理とを含む要求キューにエントリを貯える。ヘッダパケット・メモリ・ストア・コントローラは、ヘッダパケットがデータストリームに現れると、ヘッダパケットをヘッダパケットバッファにストアし、ヘッダパケット受信信号を生成する。モジュール処理回路はヘッダパケット受信信号に応答して、ヘッダパケットバッファからモジュール名を読み出し、要求バッファの各エントリの名前と、モジュール名を比較し、ヘッダパケットのモジュール名がエントリ名と一致すると、一致要求キューエントリで、要求された処理を実行する。

【００１２】

【実施例】

図１は本発明に係るオーディオ・ビデオ対話型信号デコーダの一部を示すブロック図である。図１に示すデコーダは、オーディオ・ビデオ対話型プログラムへの参加を希望する視聴者ロケーションに設けられている。図１において、トランスポート機構（図示せず）はデコーダの入力端子５に接続されている。入力端子５はチューナ１０の入力端子に接続されている。チューナ１０の出力端子は、オーディオ・ビデオ対話型プログラム・コンポーネント検出器３０のデータ入力端子に接続されている。プログラム・コンポーネント検出器３０のデータ出力端子は、処理ユニット４０のシステムバス４１６に接続されている。処理ユニット４０は、ＣＰＵ（ｃｅｎｙｒａｌ　ｐｒｏｃｅｓｓｉｎｇ　ｕｎｉｔ）４１０と、リード／ライトメモリ（ＲＡＭ）４１２と、ＲＯＭ（ｒｅａｄ　ｏｎｌｙ　ｍｅｍｏｒｙ）４１４を含み、共に、システムバス４１６に周知の方法で接続されている。ストリームＩ／Ｏアダプタ４０８は、システムバス４１６と、プログラム・コンポーネント検出器３０の制御端子間に双方向接続される。

【００１３】

オーディオ処理装置４１８はシステムバス４１６に接続されており、オーディオ・ビデオ対話型オーディオ出力端子２５にオーディオ信号を供給する。ビデオ処理装置４２０も、システムバス４１６に接続されており、ビデオ信号をオーディオ・ビデオ対話型ビデオ出力端子１５に供給する。さらに、入出力機能がＩ／Ｏポート４２２により与えられ、双方向端子４５を介して、図示しないローカル処理装置に結合されている。ユーザＩ／Ｏアダプタ４２４はユーザからのデータを入力端子３５を介して受信する。モデム４２６は双方向端子５５を介して外部コンピュータ（図示せず）に接続されている。ユーザＩ／Ｏアダプタ４２４と、モデム４２６も、周知の方法でシステムバス４１６に接続される。他の装置、例えば、数値計算プロセッサ、他のＩ／Ｏアダプタ等を、周知の方法で、システムバス４１６に接続することができる。その上、デコーダのケースの外部のケース内の他の機器と接続するためのバス拡張装置を含ませることができる。

【００１４】

動作時、トランスポート機構は、例えばデコーダへの直接ＲＦ衛星リンクか、ケーブルシステムフィード（ｃａｂｌｅ　ｓｙｓｔｅｍ　ｆｅｅｄ）か、あるいは光ファイバーリンクでも可能であり、複数のオーディオ・ビデオ対話型信号を搬送する。複数のオーディオ・ビデオ対話型信号は、ユーザにより選択され視聴される。直接衛星リンクでは、例えば、複数のオーディオ・ビデオ対話型データストリームは、個々のＲＦ搬送波信号を変調することにより、トランスポート機構上で周波数多重化されたものでもよい。各ＲＦ搬送波信号は、衛星内の各トランスポンダから視聴者に再び放送される。チューナ１０は、周波の方法で、処理ユニット４０の制御により、所望のＲＦ変調信号を選択する。例えば、直接衛星システムでは、所望のオーディオ・ビデオ対話型プログラム信号のコンポーネントを運ぶパケットサービスを含むＲＦ変調信号を、周知のＲＦチューナにより同調をとることができる。チューナ１０の出力は、それらのパケットサービスを含むベースバンドのデジタル・パケット・データストリームである。

【００１５】

ＣＰＵ４１０は、次のようにして、所望のパケットサービスをプログラム・コンポーネント検出器３０に対して要求する。すなわち、所望のサービス識別子とＲＡＭ４１２バッファのロケーションを、ストリームＩ／Ｏアダプタ４０８を介して、適正なＳＣＩＤ（ｓｅｒｖｉｃｅ　ｃｏｍｐｏｎｅｎｔ　ｉｄｅｎｔｉｆｉｅｒ）と、プログラム・コンポーネント検出器３０内のＤＭＡ（ｄｉｒｅｃｔｍｅｍｏｒｙ　ａｃｃｅｓｓ）コントローラ・レジスタとに書き込むことにより、要求を行っている。そして、プログラム・コンポーネント検出器３０は、所望のパケットサービスのため、パケットデータストリームを監視する。所望のパケットサービスから、ヘッダパケットが受信されると、そのヘッダパケットは、周知のＤＭＡ書き込み技法を用いて、ＲＡＭ４１２内の予め定めたヘッダパケットバッファに貯えられ、ヘッダパケット割り込みが発生される。所望のパケットサービスからデータパケットが受信されると、そのデータパケットは周知のＤＭＡ書き込み技法を使用して、ＲＡＭ４１２内の前もって特定されているバッファロケーションに貯えられる。伝送ユニット内の全てのデータパケットが受信されると、データ完了割り込みが発生される。パケットサービスからのヘッダパケット、および／または、データパケットの受信は、ＣＰＵ４１０の制御によりイネーブルにされるか、あるいは、ディスエーブルにされる。１９９４年４月２２日に出願された米国特許出願第２３２，７８７号（発明者：Ｋ．Ｅ．Ｂｒｉｄｇｅｗａｔｅｒ外、発明の名前：ＰＡＣＫＥＴ　ＶＩＤＥＯ　ＳＩＧＮＡＬ　ＩＮＶＥＲＳＥ　ＴＲＡＮＳＰＯＲＴ　ＰＲＯＣＥＳＳＯＲ　ＭＥＭＯＲＹ　ＡＤＤＲＥＳＳ　ＤＩＲＣＵＩＴＲＹ）を参照されたい。ここに、この特許出願番号を付して、プログラム・コンポーネント検出器３０の詳細な説明の一部とする。

【００１６】

例えば、新しいＲＦ変調信号がチューナ１０により同調されると、固定されたプログラムガイドサービス識別子を、プログラム・コンポーネント検出器３０内のサービス識別子レジスタに供給することにより、プログラムガイドを含むパケットサービスが、ＣＰＵ４１０により要求される。プログラムガイドパケット内のデータが受信され、メモリに貯えられると、そのデータにより、ＣＰＵ４１０は所望のオーディオ・ビデオ対話型プログラムのためのパケットデータサービスを要求することができる。

【００１７】

要求されたパケットサービス内のパケットがプログラム・コンポーネント検出器３０により受信され、しかも、ＤＭＡにより、ＲＡＭ４１２内の予め特定されたバッファロケーションに書き込まれた後に、ビデオ処理装置４２０とオーディオ処理装置４１８が、プログラム・コンポーネント検出器３０の制御の下に、周知のＤＭＡ読み出し技法を用いて、各パケットサービスと関連するＲＡＭ４１２バッファロケーションからデータを読み出す。そして、ビデオ処理装置４２０とオーディオ処理装置４１８は、圧縮され、符号化されたデータを復号し、出力端子１５にオーディオ・ビデオ対話型ビデオ信号を出力し、出力端子２５にオーディオ・ビデオ対話型オーディオ信号を出力する。ＣＰＵ４１０は、復号プロセスにおいて、ビデオ処理装置４２０および／またはオーディオ処理装置４１８と協力することも可能である。データコンポーネントパケットサービスパケットは、次のような方法により、ＣＰＵ４１０の制御の下に処理される。

【００１８】

上述したように、プログラム・コンポーネント検出器３０により、要求されたパケットサービスからヘッダパケットが受信されるごとに、そのヘッダパケットはそのパケットサービスのためにＲＡＭ４１２内の予め定めたロケーションに貯えられ、ヘッダパケット割り込み信号がＣＰＵ４１０のために生成される。ヘッダパケット割り込み信号に応答して、割り込みハンドラ（ｈａｎｄｌｅｒ）が実行される。割り込みハンドラは、ヘッダパケットの内容を解析し、プログラム・コンポーネント検出器３０内のＤＭＡレジスタのＲＡＭ４１２バッファロケーションを適正に更新し、ＤＭＡ転送をイネーブルにするか、あるいは伝送ユニットが望まれていない場合は、ＤＭＡ転送をディスエーブルにする。ＤＭＡ転送が一度イネーブルにされると、データパケット内のデータはＤＭＡ制御によりＲＡＭ４１２内にロードされる。データパケットのロードが完了すると、プログラム・コンポーネ

ント検出器３０はデータ完了割り込み信号を発生する。データ完了割り込み信号に応答して、割り込みハンドラが実行され、クリーンアップ（ｃｌｅａｎ－ｕｐ）機能を実行し、次のヘッダパケットのための準備をする。

【００１９】

図２は、図１に示す処理ユニット４０により実行されるソフトウェア２００の構造を示す。図２はオーディオ・ビデオ対話型処理マルチタスク・オペレーティングシステムを構成する主なソフトウェア・コンポーネントを示す。図２を説明する。コンポーネントは全てＲＯＭ４１４に貯えられる。ただし、黒く塗られたアプリケーション・プログラムは除く。アプリケーション・プログラムは、オーディオ・ビデオ対話型信号のデータコンポーネントにより運ばれ、放送ロケーションから受信され、ＲＡＭ４１２に貯えられる。図２に示すソフトウェア・コンポーネントは、実行可能なコードおよび関連する定数データを表す。コードが実行されると、変数データを生成し、変数データにアクセスする。変数データはＲＡＭ４１２に貯えられる。

【００２０】

提案されたオーディオ・ビデオ対話型放送システムでは、例えば、異なる製造業者からの異なる命令セットを用いて、異なるデコーダはＣＰＵを使用することができる。このシステムでは、アプリケーション・プログラムは、プロセッサに無関係な中間コードである。各デコーダのソフトウェアは、アプリケーション中間コードを解釈するコンポーネント（ＩＮＴＥＲＰＲＥＴＥＲ：インタプリタ）を含む。インタプリタにより、放送されたアプリケーション・プログラムは、どのような形式のＣＰＵ４１０を含むデコーダ上でも実行することができる。このインタープリタは、ＲＡＭ４１２からオーディオ・ビデオ対話型データコンポーネント命令を中間コードの形で読み出し、メモリを操作し、ＡＰＩ（ａｐｐｌｉｃａｔｉｏｎ　ｐｒｏｇｒａｍ　ｉｎｔｅｒｆａｃｅ）を介して、他のソフトウェア・コンポーネントを介してハードウェアと対話する。このＡＰＩは、基本的には、アプリケーション・プログラムで利用可能なサブルーチンのリストであって、サブルーチンを呼び出すのに必要な情報である。ＡＰＩはアプリケーション・プログラムにより発行され、デコーダ要素をアクセスするために、アプリケーション・プログラムにより使用することができる。

【００２１】

数学ライブラリは整数演算と浮動小数点演算を実行するのに必要な機能を全て実行する。フロー・オペレーティング・システムは、オーディオ・ビデオ対話型信号のデータコンポーネントを監視するのに必要な全てのドライバーと、プロセス要求されたモジュールとを制御する。このことは後程詳細に説明する。ユーザインターフェース管理コンポーネントは、ユーザとの対話を全て扱い、ユーザと通信するため、イベントマネジャとグラフィックス・ライブラリを利用する。このグラフィックス・ライブラリは、受信されたオーディオ・ビデオ対話型ビデオ上にオーバレイする全てのグラフィックイメージを生成し、数学ライブラリを用いて、複雑な曲線を描く。

【００２２】

デコーダソフトウェアの異なるソフトウェアコンポーネントは、互いにメッセージを送信して、他と非同期に通信する。各プログラムコンポーネントは、メッセージキュー（ｍｅｓｓａｇｅ　ｑｕｅｕｅ）を有し、そのキューから次のメッセージを繰り返し読み出し、そのメッセージを処理し、他のプログラムコンポーネントにメッセージを送信することにより、オペレートする。メッセージ保留がない場合は、次のメッセージを待つ。イベントマネジャは、メッセージを適正にルーティングし、メッセージキューを維持することにより、他のソフトウェアコンポーネントとの間でのメッセージ通信を管理する。

【００２３】

各ハードウェアアダプタは、関連するソフトウェアドライバも含む。ドライバは、関連するハードウェアアダプタ内のレジスタとＣＰＵ４１０との間で、システムバス４１６を介して、実際の対話を行う。例えば、モデム４２６と、外部Ｉ／Ｏポート４２２と、ストリームＩ／ＯアダプタＩ／Ｏ　４２４に対して、ドライバが存在する。さ

らに、別のドライバがソフトウェアタイマを保守し、デコーダのフロントパネルを操作する。これらのドライバはイベントマネジャに密接に依存している。上記のコンポーネントは、全て、マルチタスクカーネルにより供給される共通の機能を使用する。例えば、カーネルは、プロセスの優先順位と、アクティブタスクキューと、信号と、セマフォアと、プリエンプティブ・スイッチング・クロック・チックと、割り込み（ハードウェアおよびソフトウェア）と、プロセススタックを保守する。さらに、カーネルは、ハードウェア初期化を行い、システムローダである最初のシステムタスクを開始する。

【００２４】

開始時に、システムローダはフロー・オペレーティング・システムへのＡＰＩコールを実行し、フロー・オペレーティング・システムはストリームドライバをコールし、ストリームＩ／Ｏアダプタ４０８を介して、プログラム・コンポーネント検出器３０に適正なデータを送る。システムローダからのこれらのＡＰＩコールにより、ディレクトリ・モジュールに対して、データコンポーネントパケットサービスのスキャンを開始する。この方法は以下に詳細に説明する。ディレクトリ・モジュールが見つけられると、ディレクトリ・モジュールがＲＡＭ４１２にロードされ、そのプログラムを実行するのに必要な資源が全て利用可能か否かを知るため、チェックされる。肯定判定された場合は、システムローダはオートスタート・モジュールと呼ばれる最初のモジュールに対してオーディオ・ビデオ対話型データコンポーネントの走査を開始し、それによりオーディオ・ビデオ対話型プログラムは起動されることになる。オートスタートモジュールが見つかると、データコンポーネント・パケットサービスから取り出され、ＲＡＭ４１２にロードされる。このオートスタートモジュールは、中間コードの形式であり、インタプリタによりインタプリートされて実行される。オートスタートモジュールは初期化の残りを実行し、オーディオ・ビデオ対話型プログラムの実行を開始する。このプログラムは他のコードモジュールとデータモジュールをロードすることができ、全て、ＡＰＩコールにより、他のコードモジュールとチェイン（ｃｈａｉｎ）することができる。このようにして、システムローダはクラシックのＵＮＩＸ（登録商標）シェルと同様に動作する。

【００２５】

さらに、システムローダは引き続きデータコンポーネント・パケットサービスをスキャンし、伝送されたディレクトリ・モジュールと、ＲＡＭ４１２内の現在のディレクトリ・モジュールとを比較する。伝送されたディレクトリ・モジュールがＲＡＭ４１２に格納されているディレクトリ・モジュールと異なる場合、データコンポーネント・パケットサービスが変更されたこと、例えば、視聴者がチャンネルを変えたときとか、対話型コマーシャルが放送されているときのようなことを示す。この場合、メッセージが、イベントマネジャを介して、ＡＰＩを用いて、アプリケーション・プログラムに送られる。このメッセージに応答して、アプリケーション・プログラムは全ての資源の割り振りを解除し、処理要素４０内に最小限存在するように維持する。例えば、コードモジュールとデータモジュールの全てを格納するために使用されるメモリを解放することができ、アプリケーションの実行状態だけがＲＡＭ４１２に保たれる。アプリケーション・プログラムの最小化が完了すると、メッセージがシステムローダに送られる。

【００２６】

そして、新しいディレクトリモジュールにより表わされるオーディオ・ビデオ対話型プログラムを実行するのに必要な資源を、システムローダは割り振る。新しいディレクトリ・モジュールがオーディオ・ビデオ対話型データコンポーネント・パケットサービス内で検出されたとき、前に最小化されたアプリケーションのリストが検索される。そして、新しいディレクトリにより表されるアプリケーションが存在する場合は、そのアプリケーションは、データコンポーネントフローから、必要なコードモジュールとデータモジュールを再ロードすることにより再開される。また、前に停止した所から実行を再開することにより再開される。このモジュールは、介在する対話型コマーシャルの終了時に起きる。このプロセスは繰り返すことができる。ただし、第２のオーディオ・ビデオ対話型プログラムに、第３のオーディオ・ビデオ対話型プログラムが割りこむことができる。後に、再起動

される。

【００２７】

図３はフローとメモリのレイアウトを示す。図４はオーディオ・ビデオ対話型プログラム中のデータコンポーネントからのモジュール抽出を理解するのに有用な詳細なメモリレイアウトと（図１の）プログラム・コンポーネント検出器３０の更に詳細なブロック図である。図４において、（図１の）チューナ１０からのベースバンドのデジタルパケットストリームが、プログラム・コンポーネント検出器３０内でヘッダパケットＤＭＡコントローラ３４と、データＤＭＡコントローラ３２との各データ入力端子に接続されている。データＤＭＡコントローラ３２と、ヘッダパケットＤＭＡコントローラ３４の各データ出力端子は、処理ユニット４０のシステムバス４１６に接続されている。ストリームＩ／Ｏアダプタ４０８は、システムバス４１６と、データＤＭＡコントローラ３２の制御入力端子と、ヘッダパケットＤＭＡコントローラ３４の制御入力端子との間に接続されている。動作時には、ストリームＩ／Ｏアダプタ４０８は、（図１の）ＣＰＵ４１０から、データＤＭＡコントローラ３２とヘッダパケットＤＭＡコントローラ３４に、制御情報、例えば、バッファロケーション開始アドレスおよび終了アドレスと、読み出しアドレス／書き込みアドレスと、転送計数値を、周知の方法で供給する。そして、ＣＰＵ４１０の制御の下に、ストリームＩ／Ｏアダプタ４０８は、データＤＭＡコントローラ３２および／またはヘッダパケットＤＭＡコントローラ３４をイネーブルにして、周知の方法で、データパケットまたはヘッダパケットを、パケットストリームからバッファに転送するか、あるいは、このような転送をディスエーブルにする。データＤＭＡコントローラ３２がデータ転送を完了すると、ＣＰＵ４１０に対して、データ完了割り込みを生成する。ヘッダパケットＤＭＡコントローラ３４がヘッダパケットのローディンを完了すると、ＣＰＵ４１０に対して、ヘッダパケット割り込みを生成する。

【００２８】

図４でも、ＲＡＭ４１２を大きなブロックで表し、データ構造を大きなブロック内の小さいブロックで表す。図４のブロックは図示のためだけであり、絶対記憶位置または相対記憶位置の何れかを示すものではなく、データ構造に対して、ＲＡＭ４１２に割り振られたサイズを示すものでもない。参照番号４１２に、モジュールリクエストキュー３２２と、ヘッダパケットバッファ３２４と、ディレクトリ・モジュールバッファ３２６と、モジュールバッファ３２８のデータ構造を示す。テータ構造内の情報のフィールドは、そのフィールド内に含まれる情報のタイプの名前を含む水平スライスとして示されている。これらについては以下に詳細に説明する。

【００２９】

図３は、ヘッダパケットデータコンポーネント・パケットサービスからモジュールを取り出し、しかも、そのモジュールをＲＡＭ４１２内のバッファに格納する際の後続の手順を示す。同様な手順は他のモジュール処理に対しても続けられる。これも以下に説明する。図３では、アプリケーション・プログラム（またはシステムローダ）で取られるアクションを、“ＡＰＰＬＮＰＲＯＧ”のヘッドを付けた左カラムに示す。ブロック３０２で、アプリケーション・プログラムは、ＡＰＩを用いて、フロー・オペレーティング・システムにリクエストを出し、識別子ＩＤを有するモジュールを、オーディオ・ビデオ対話型プログラムコンポーネントパケットサービスからロードする。上述したように、ＡＰＩコールは、基本的には、オペレーティングシステム機能へのサブルーチンコールである。したがって、プログラムの実行はフロー・オペレーティング・システム（ＦＯＳ）に移行される。フロー・オペレーティング・システムのアクションは“フロー・オペレーティング・システム”のヘッドを付けた右隣のカラムに示す。リクエストはモジュールのロードを含むので、ブロック３１２で、フロー・オペレーティング・システムはモジュールのローディングを含むので、ブロック３１２にて、フロー・オペレーティング・システムは、メモリマネジャから、そのモジュールを含むだけのサイズのメモリの割り振りをリクエストする。例えば、リクエストされたモジュールがコードモジュールまたはデータモジュールである場合は、（図４の）前に格納されたディレクトリ・モジュール３２６は、モジュールＩ

Ｄの長さ（ＬＥＮＧＴＨ）を含んでいるフィールドを含む。この場合、メモリマネジャは、開始アドレスＳＴＡＲＴと終了アドレスＥＮＤを有するモジュールメモリバッファ（図４の３２８）を割り振る。そして、ブロック３１４においてリクエスト、例えば、モジュールの識別子ＩＤと、リクエストＲＥＱＵＥＳＴ（この場合はモジュールを取り出しロードするリクエスト）のタイプを示す情報と、割り振られたバッファ開始アドレスＳＴＡＲＴと終了アドレスＥＮＤが、全て、リクエストキュー（ＱＶＥＶＥ）３２２内のエントリに格納される。そして、ヘッダパケットがパケットストリーム内に生じたとき、ヘッダパケットＤＭＡコントローラがイネーブルにされ、ヘッダパケットをＲＡＭ４１２にロードする。

【００３０】

リクエストがディレクトリ・モジュールに対するものである場合は、その長さが元々分かっていない。この場合、比較的大きいメモリ割り振りがリクエストされる。この割り振りがあまりにも小さい場合は、より大きなメモリ割り振りをリクエストした後、ディレクトリ・モジュールがロードされるか、あるいはそれをロードするだけのメモリがないと決定されるまでリクエストは繰り返えされ、その場合、オーディオ・ビデオ対話型プログラムをランさせる試みは放棄される。

【００３１】

また、フロー・オペレーティング・システムはコールしたアプリケーション・プログラムに直ちに戻る。次いで、アプリケーション・プログラムは他の処理、例えば、他のモジュールのリクエストを発し、他の初期化等の処理を実行することができる。リクエストされたモジュールへのアクセスが必要とされると、アプリケーション・プログラムは、ブロック３０４にて、カーネル内の待ち機能にＡＰＩコールを発する。この機能により、リクエストされたモジュールのロードが成功したことを示すメッセージが、そのアプリケーション・プログラムにより受信されるまで、アプリケーション・プログラムの実行は中断させられる。そのようなメッセージが受信されると、アプリケーション・プログラムは再びアクティベートされ、そのメッセージを処理する。あるいはまた、例えば、より速くユーザ入力に応答するため、アプリケーション・プログラムはアクティブのままであり、そのメッセージキューを周期的にポーリングし、リクエストされたモジュールロードの成功を示すメッセージをチェックし、そのメッセージが受信されたとき、メッセージを処理するようにしてもよい。

【００３２】

上述したように、ヘッダパケットＤＭＡコントローラ３４は、メモリマネジャにより、前に割り振られたＲＡＭ４１２のヘッダパケット（ＨＤＲ　ＰＫＴ）バッファ３２４（図４）にヘッダパケットをロードし、ＣＰＵ４１０にヘッダパケット割り込みを発する。カーネル内のヘッダ割り込みハンドラにより実行される処理の一部を、図３に“ＨＥＡＤＥＲ　ＩＮ　ＴＲ”のヘッドを付して示す。ブロック３３２にて、伝送ユニットに入れて運ばれているモジュールの識別子、この場合、ヘッダパケットは、ヘッダパケットバッファ３２４内の知られたロケーション、すなわち、ＩＤから取り出される。ブロック３３４にて、リクエストキュー３２２が試験され、このモジュールに対する保留中のリクエストが存在かるか否かが判定される。

【００３３】

そのモジュールに対する保留中のリクエストがある場合には、ブロック３３６において、プログラム・コンポーネント検出器３０のデータパケットＤＭＡコントロール３２は、リクエストキュー３２２からのアドレスＳＴＡＲＴで始まり、アドレスＥＮＤで終了するモジュールバッファ３２８と；モジュールバッファ３２８の開始アドレスＳＴＡＲＴと送信ユニットのデータオフセットＯＦＦＳＥＴの和、すなわち（ＳＴＡＲＴ＋ＯＦＦＳＥＴ）である書き込みアドレスと；ＳＴＡＲＴ＋ＯＦＦＳＥＴ＋ＳＩＺＥ（あるいは代りに、最後の書き込みアドレスに代わってヘッダパケットバッファ３２４からのサイズＳＩＺＥであるロード計数値）である最後の書き込みアドレスと；で初期化される。そして、データパケットＤＭＡコントローラ３２がイネーブルにされる。

【００３４】
ブロック３３８にて、これがロードリクエストの行われた後受信された最初のヘッダパケットである場合は、リクエストキュー３２２に格納される最初の書き込みアドレスのポインタＦＩＲＳＴが、この最初の送信ユニットの書き込みアドレス（すなわち、ＦＩＲＳＴ＝ＳＴＡＲＴ＋ＯＦＦＳＥＴ）に初期化される。さらに、予想される次の書き込みアドレスを指すポインタＮＥＸＴも、リクエストキュー３２２に格納されており、最初の送信ユニットの書き込みアドレス（すなわち、ＮＥＸＴ＝ＳＴＡＲＴ＋ＯＦＦＳＥＴ）に初期化される。そして、他の処理がブロック３３８にて行われる。これについては以下に詳細に説明する。例えば、現在処理されているリクエストのリクエストキュー３２２内のロケーションを指す特殊なポインタＣＵＲＲ　ＲＥＱは、ＲＡＭ４１２内の予め決められたロケーション（図示せず）に格納される。その後、ブロック３３９にて、割り込みハンドラーはリターンする（３３９）。

【００３５】
データパケットＤＭＡコントローラ３２は、先に受信した書き込みアドレス（ＳＴＡＲＴ＋ＯＤＤＳＥＴ）を指す書き込みポインタ（ＷＰ）を初期化し、オーディオ・ビデオ対話型プログラムコンポーネントパケットサービス内の後続のデータパケットから、データをＲＡＭ４１２内のモジュールバッファ３２８内の順次ロケーションにロードする。送信ユニット内の全てのデータがＲＡＭ４１２にロードされると、データ完了割り込みが生成される。カーネル内のデータ完了割り込みハンドラにより実行される処理の一部を、図３の右カラムに“ＤＡＴＡ　ＣＯＭＰＬ　ＩＮＴＲ”のヘッドを付けて示す。

【００３６】
ブロック３４２にて、ＤＭＡ転送の現在の状態に関連するクリーンアップ機能が実行される。現在のリクエストポインタ（ＣＶＲＲ　ＲＥＱ）は、ヘッダパケット割り込みハンドラ内に前にセットされており、リクエストキュー３２２内のエントリーを指しており、伝送ユニットはリクエストキューへのロードを丁度終了したばかりである。現在のリクエスト内で、予想される次の書き込みアドレスポインタＮＥＸＴは、ヘッダパケットバッファ３２４からの値ＳＩＺＥによりインクリメントされ、次の伝送ユニットに対して予想される書き込みアドレスを指している。予想される次の書き込みアドレスポインタ、ＮＥＸＴの値が、モジュールバッファ３２８の終了アドレス、ＥＮＤに等しい場合は、ラップアランドして、書き込みアドレスポインタＮＥＸＴはモジュールバッファ３２８の開始アドレスＳＴＡＲＴにリセットされる。

【００３７】
ブロック３４４にて、リクエストされたモジュールの全体がメモリにロードされたか否かが判定される。予想される次の書き込みアドレスポインタ、ＮＥＸＴの値は、最初のロードされたアドレス，ＳＴＡＲＴの値と比べられるそれらの値が同じである場合は、モジュール全体はロードされている。ブロック３４６で、メッセージは、イベントマネジャにより、リクエストしているアプリケーション・プログラムに送られ、リクエスト・モジュールが、図３に破線で示すように、完全に取り出されたことを示す。さらに、リクエストはリクエストキュー３２２から除去される。予想される次の書き込みアドレス　ＮＥＸＴの値が最初にロードされたアドレス，ＳＴＡＲＴと同じでない場合は、データ完了割り込みハンドラはリターンし（３４９）、リクエストされたモジュールのためのデータを含む次の伝送ユニットが、先に説明したように、ヘッダパケット割り込みハンドラにより処理される。どちらの場合も、現在のリクエストポインタ（ＣＶＲＲ　ＲＥＱ）はクリアされる。

【００３８】
伝送ユニットのある部分がプログラム・コンポーネント検出器３０により適正に受信されない場合、後続のヘッダパケットが受信されてから、先行するヘッダパケットからのデータ完了割り込み信号がプログラム・コンポーネント検出器３０内のＤＭＡ回路により生成される。よって、先行するデータ完了割り込み信号が生成される前に後続のヘッダパケット割り込み信号が生成される。ヘッダパケット割り込みハンドラとデータ完了割り込みハ

ンドラは、協力して処理し、このような状況を識別することができ、そのようなエラーを処理することができる。

【００３９】

ヘッダパケット割り込みハンドラでは、データパケットＤＭＡコントローラ３４がイネーブルにされて、次の伝送ユニットを受信した後、そのような処理が（図３の）ブロック３３８で実行される。受信された各ヘッダパケットに対し、データ完了割り込みハンドラによって前に更新された現在のリクエストキューエントリの予想される次の書き込みアドレス，ＮＥＸＴが、新たに受信されるヘッダパケットのための書き込みアドレス（ＳＴＡＲＴ＋ＯＦＦＳＥＴ）と比較される。それらのアドレスが同じである場合は、前の伝送ユニットの受信が成功したことになる。しかしながら、最後の終了アドレスが新らしいオフセットと同じでない場合は、前の伝送ユニットのＤＭＡ転送が完全に成功しなかったことを意味する。この場合、前の伝送ユニットのＤＭＡ転送が完全に成功しなかったことを意味する。この場合、最初の書き込みアドレス，ＦＩＲＳＴと、予想される次の書き込みアドレス，ＮＥＸＴは、現在の書き込みアドレス（ＳＴＡＲＴ＋ＯＦＦＳＥＴ）に更新される。すなわち、前にロードされた伝送ユニットは必然的に破棄され、モジュールのロードは現在の伝送ユニットから再スタートされる。前に成功りにロードされた伝送ユニットは、再ロードされるときエラーを生じるかもしれないので、データ紛失というエラーから回復する時間が長くかかることがある。しかしなから、このような回復を用いることにより、ヘッダパケット割り込みハンドラと、データ完了割り込みハンドラにより実行されるタスクを、最小化することができ、２つのポインタのみがメモリに必要なだけである。

【００４０】

モジュール完了メッセージ処理の一部として、イベントハンドラが、受信されたモジュールに対してエラーチェックを行う。例えば、ＣＲＣ（ｃｙｃｌｉｃ　ｒｅｄｕｎｄａｎｃｙ　ｃｈｅｃｋ）コードがモジュールの埋め込み部分として伝送される。イベントハンドラは、ＲＡＭ４１２内のモジュールバッファ３２８内の受信されたモジュール全体のＣＲＣを計算し、そのＣＲＣと埋め込みＣＲＣとを比較する。新たに計算されたＣＲＣが埋め込みＣＲＣと等しい場合は、モジュールは正しく受信されたことを示し、そうでない場合は、エラーが起きたことを示し、モジュールは先に述べたように再ロードされる。

【００４１】

リクエストされたモジュールがメモリ内に完全にロードされると、アプリケーションモジュールによる更なる処理が、図３に、待ち機能３０４に対するＡＰＩコールの底から直線で示すように、推論により続行することができる。しかしながら、アプリケーション・プログラム内の別のタスクが、そのアプリケーション・プログラムのメッセージキューからのメッセージの受信に応答してアクティブにされる。

【００４２】

先に説明したＡＰＩすなわちアプリケーション・プログラム・インタフェースは、インタプリタを介してアプリケーション・プログラムか、またはシステムローダによりデータストリームへアクセスする機能を含んでいる。アプリケーションプログラマは、発行されたＡＰＩ記述を用いて、望ましいデータストリーム機能へアクセスするためのＡＰＩコールを定式化することになる。最初のグループの機能は、モジュールのディレクトリに関する。最初の機能ＤＩＲ＿ＮＥＷは、新しいディレクトリのリクエストである。先に説明したように、このＡＰＩ機能に応答して、メモリの割り振りがなされ、その後、リクエストはデータストリームに次のディレクトリ・モジュールをロードするためにエンキューされ、ＡＰＩ機能はリターンする。ディレクトリがロードされたとき、メッセージは、リクエスト・プログラムに送られる。他の機能ＤＩＲ＿ＦＲＥＥは、現在のディレクトリにより占められているメモリ空間を解放する。機能ＤＩＲ＿ＳＥＬＥＣＴは、どのディレクトリ・モジュールが後続のＡＰＩコールで使用されるかを示した。機能ＤＩＲ＿ＣＵＲＲＥＮＴは、現在選択されているディレクトリにハンドラを戻る。

【００４３】

機能ＤＩＲ＿ＳＰＹとＤＩＲ＿ＳＴＯＰ＿ＳＰＹはＤＩＲ＿ＮＥＷ機能と同様である。Ｄ

ＩＲ＿ＳＰＹ　ＡＰＩコールに応答して、リクエストは、ディレクトリ・モジュールのリクエストキューにエンキューされる。しかし、ディレクトリ・モジュールをロードし、しかも、ディレクトリ・モジュールがロードされたときメッセージを送信する代わりに、この機能は、ディレクトリモジュールがデータフロー（ディレクトリ・モジュールはロードされていない）内で検出されたときにはいつでもメッセージを送る。また、リクエストは、ＤＩＲ＿ＳＴＯＰ＿ＳＰＹ　ＡＰＩコールが行われるまで、リクエストキュー内に残ったままである。ＤＩＲ＿ＳＴＯＰ＿ＳＰＹ　ＡＰＩコールがなされたとき、リクエストキュー内にディレクトリ・スパイ・リクエストを検索し、そのエントリが除去される。これらの機能は、データストリーム内の現在のディレクトリから何らかの変更をスパイするのに有用である。最後に、現在のディレクトリについての情報を抽出するためのＡＰＩコール、すなわちＤＩＲ＿ＩＤＥＮＴＩＦＩＥＲと、ＤＩＲ＿ＲＥＱＵＩＲＥＭＥＮＴと、ＤＩＲ＿ＮＢ＿ＭＯＤＵＬＥＳが存在する。

【００４４】

モジュール内に埋め込みＣＲＣコードがあるので、モジュールをロードするためのメモリ割り振りリクエストは、このコードを考慮しなければならない。３つのＡＰＩコールがこれを処理するために与えられている。機能ＭＯＤＵＬＥ＿ＡＬＬＯＣは、ＣＲＣまたはメモリ要件を考慮して、引数としてモジュール識別子を取り、そのモジュールをロードするための適当なメモリ容量割り振りをリクエストする。機能ＭＯＤＵＬＥ＿ＦＲＥＥは、モジュールにより取られるメモリ領域を解放する。機能ＭＯＤＵＬＥ＿ＣＨＥＣＫは、ロードされたモジュールのＣＲＣチェックを行い、その結果を戻す。これは、ＣＲＣがメモリ内にロードされたモジュール内に埋め込まれているので、何時なされてもよい。

【００４５】

ＡＰＩコールの他のセットは、モジュールを取り扱い、現在選択されているディレクトリを用いてそれらを識別する。モジュールについての情報を抽出するためのＡＰＩコールがあり、それらは、ＭＯＤＵＬＥ＿ＲＥＱＵＩＲＥＭＥＮＴとＭＯＤＵＬＥ＿ＳＩＺＥと、ＭＯＤＵＬＥ＿ＦＬＡＧである。これらにより、システムは、モジュールがロードおよび／または実行することができるか否かを判定する。機能ＭＯＤＵＬＥ＿ＲＵＮは、先に述べたように実行可能なモジュールをロードし、新しいプロセスを作成し、モジュールのエントリポイントで実行を開始するために使用される。この機能は、オーディオ・ビデオ対話型プログラムの実行を開始するためのシステムローダにより使用される。機能ＭＯＤＵＬＥ＿ＣＨＡＩＮは、後続の実行可能なモジュールをロードし、現在のモジュールの実行を終了し、そのエントリポイントで新たにロードされるモジュールの実行を開始するために使用される。この場合には新しいプロセスは生成されるが、実行を開始しない。上述したように、メッセージは、モジュールのロードが完了したとき、リクエスト・プログラムに送られる。機能ＭＯＤＵＬＥ＿ＥＸＥＣは、新たなプロセスを生成し、ＭＯＤＵＬＥ＿ＬＯＡＤ　ＡＰＩコールにより前にロードされているモジュールの実行をエントリポイントで開始するために使用される。

【００４６】

機能ＭＯＤＵＬＥ＿ＬＩＮＫは、現在のプロセスと、資源と、変数を用いて新しいモジュールを実行する。それは、新しいモジュールへの動的なリンクを与えることにより、モジュール内からのサブルーチンのようなコールを許可する。これにより、必要なときだけダイナミックにリンクすることができる、より小さいモジュールにオーディオ・ビデオ対話型プログラムを分割することができる。ＭＯＤＵＬＥ＿ＬＩＮＫ機能は、リロケーションとジャンプテーブルを保持する。ＭＯＤＵＬＥ＿ＳＰＹとＭＯＤＵＬＥ＿ＳＴＯＰ＿ＳＰＹは、ＤＩＲＥＣＴＲＹ＿ＳＰＹとＤＩＲＥＣＴＲＹ＿ＳＴＯＰ＿ＳＰＹと同様に作動する。ただし、識別されたモジュールに対してである。ＭＯＤＵＬＥ＿ＳＰＹ　ＡＰＩコールは、モジュールの識別子を含むリクエストキュー内にエントリを挿入する。同じ識別子を有するヘッダモジュールがデータストリーム内に検出されたときにはいつでも、メッセージがリクエスト・プログラムに送られる。これは、ＭＯＤＵＬＥ＿ＳＴＯＰ　ＳＰＹ　ＡＰＩコールがなされるまで続く。ＭＯＤＵＬＥ＿ＳＴＯＰ＿ＳＰＹ　ＡＰＩコールに応

答して、識別されたモジュールに対するスパイリクエストを含むエントリが、リクエストキューから除去される。ＭＯＤＵＬＥ＿ＳＴＯＰ＿ＬＯＡＤ機能はプロセス内の現在のモジュールロードリクエストを停止し、リクエストキューからロードリクエストエントリを除去する。機能ＦＬＯＷ＿ＭＥＳＳＡＧＥとＦＬＯＷ＿ＳＴＯＰ＿ＭＥＳＳＡＧＥは、データストリームに関して、中断されたデータフローか、あるいはデータフローの終了のような特別な連絡パケットが生起したとき、メッセージシのリクエストを除去する。そのようなイベントが生起したとき、メッセージはリクエスト・プログラムに送られる。

【００４７】

先に説明したように、システムローダは、システムの初期化を実行し、アプリケーション・プログラムの実行が受信されたオーディオコンポーネントとビデオコンポーネントとに同期することを確実にするためにデータストリームを監視する。図５は、システムロードの初期化機能を示すフロー図である。図５のブロック５２で、（１７の）デコーダの種々のハードウェアとソフトウェアのコンポーネントが初期化される。また、ＲＡＭ４１２内のロケーションが、種々のデータ構造のために割り振られ初期化される。これらの初期化機能は周知であり、デコーダの他のソフトウェアコンポーネントに依存する。システムプログラマは、どのハードウェアとソフトウェアの初期化が必要か、どのデータ構造が必要か、初期化をどのように行うかを理解するであろう。従って、このブロックは以下では詳細に説明しない。

【００４８】

ブロック５４にて、上述したＤＩＲ＿ＮＥＷ　ＡＰＩコールが行われる。このＡＰＩコールは、オーディオ・ビデオ対話型プログラム・コンポーネント・パケットサービス内に現れる次のディレクトリ・モジュールを、ＲＡＭ４１２内の割り振られたバッファ内にロードする。このＡＰＩコールは、ディレクトリが後までＲＡＭ４１２内にロードされないとしても直ちにシステムローダに戻る。システムローダは、他の機能を実行し、必要なら、ディレクトリ・モジュールがロードされたことを示すメッセージが、イベントマネジャにより、受信されるまで、ＡＰＩコール（図示せず）を実行する。ブロック５６にて、（図１の）デコーダで使用可能な資源はディレクトリモジュール内で必要な資源を示すデータと比較される。デコーダが、オーディオ・ビデオ対話型プログラムを実行するだけの資源を有している場合は、ＭＯＤＵＬＥ＿ＲＵＮ　ＡＰＩコールが、上述したように、前にロードされているディレクトリモジュール内で識別されるオートスタートコードモジュールをロードするために行われる。再び、ＡＰＩコールは直ちにリターンし、コードモジュールは、ある時間が経過するまで、データストリームから完全にはロードされなくてもよい。オートスタートコードモジュールが完全にロードされた後、他のタスクが、インタプリタを介してオーディオ・ビデオ対話型プログラムを実行するためのマルチタスクカーネルを用いて周知の方法で生成される。

【００４９】

ブロック５８にて、システムローダは、実行信号とディレクトリの変更のためのオーディオ・ビデオ対話型プログラムのコンポーネントを監視し始め、以下に説明するオーディオ・ビデオ対話型プログラムへメッセージを送ることにより、オーディオ・ビデオ対話型プログラムの実行を制御する。図６は、システムローダの監視機能を示す状態遷移図であり、システムローダの動作を理解するのに有用である。ディレクトリがオーディオ・ビデオ対話型プログラム・コンポーネント・パケットサービス内で検出された場合は、視聴者が選択したプログラムは対話型プログラムである。ディレクトリがＲＡＭ４１２内に一旦ロードされ、オートスタートコードモジュールがオーディオ・ビデオ対話型コンポーネントパケットサービスからリクエストされると、システムローダの制御により、オーディオ・ビデオ対話型プログラムはＩＮＡＣＴＩＶＥ状態６１に入る。ＩＮＡＣＴＩＶＥ状態６１では、アプリケーションを開始するための全ての資源は割り振られており、アプリケーションは部分的に、あるいは完全にロードされるが、視聴者との対話はない。例えば、オートスタートモジュールがロードされているとき、オーディオ・ビデオ対話型プログラムはＩＮＡＣＴＩＶＥ状態６１のままである。その上、オートスタートモジュールがロードさ

れた後でさえ、視聴者はオーディオ・ビデオ対話型プログラムを運ぶチャンネルを介してチャンネルを変更するだけであり、オーディオ・ビデオ対話型プログラムと対話する意図はない。あるいは、視聴者は対話を決意する前に、オーディオ・ビデオ対話型プログラムを観察したいかもしれない。どの場合にも、リモートコントロールが対話モードではなく、通常のチャンネル変更モードで働くと言うことが重要である。これがＩＮＡＣＴＩＶＥ状態６１の目的である。見ているチャンネルが対話型プログラムを放送していることを視聴者に知らせるために、特別の対話型プログラムロゴまたはアイコンがオーディオ・ビデオ対話型ビデオ上に重ね合わされる。

【００５０】

視聴者がオーディオ・ビデオ対話型プログラムと実際に対話し始めるため、ＡＣＴＩＶＡＴＥ　ＫＥＹと呼ばれる特別なキーがリモートコントロール上に設けられている。対話型プログラムロゴまたはアイコンが表示されているとき、視聴者はＡＣＴＩＶＡＴＥ　ＫＥＹを押すことができる。ＡＣＴＩＶＡＴＥ　ＫＥＹの押下に応答して、システムローダはＡＣＴＩＶＡＴＥメッセージをＡＣＴＩＶＡＴＥ状態６３に入るオーディオ・ビデオ対話型プログラムに送る。ＡＣＴＩＶＥ状態６３では、インタプリタはそのエントリポイントで、前にロードされたオーディオ・ビデオ対話型プログラムを実際に実行し始める。オーディオ・ビデオ対話型プログラムのオートスタートモジュールが実行を開始するとき、それは、ＲＡＭ４１２内にそれ自身のデータ構造を割り振り、初期化し、他のコードモジュールおよび／またはデータモジュールをロードし、リモートコントロールとフロントコントロールパネルの全てのユーザアクションを制御する。

【００５１】

オーディオ・ビデオ対話型プログラムは全てのユーザ対話を制御するので、それはユーザがチャンネルを変更することを妨げるか、あるいは他の通常のリモートコントロール機能を実行することを妨げる。通常のリモートコントロール機能に戻るには、視聴者は、先ず現在のオーディオ・ビデオ対話型プログラムを停止しなければならない。視聴者がＡＣＴＩＶＡＴＥ　ＫＥＹを再び押下すると、プログラムは非アクティブにされる。このキーの押下に応答して、システムローダは、ＤＥＡＣＴＩＶＡＴＥメッセージを実行中のオーディオ・ビデオ対話型プログラムに送り、そのプログラムは、ＡＣＴＩＶＥ状態６３を離れ、ＩＮＡＣＴＩＶＥ状態６１に戻る。再び、特別の対話型プログラムロゴまたはアイコンが表示され、オーディオ・ビデオ対話型プログラムはロードされているが実行されていないということを示す。視聴者は、その後、チャンネルを変更し、あるいは他の通常のリモートコントロール機能を実行し、あるいはＡＣＴＩＶＥ　ＫＥＹを再び押下することによりオーディオ・ビデオ対話型プログラムを再びアクティブにしてもよい。こうして、ＡＣＴＩＶＡＴＥ　ＫＥＹは、それが押されたとき、ＡＣＴＩＶＥ状態６３とＩＮＡＣＴＩＶＥ状態６１との間で切り替わるトグルとして働く。ＡＣＴＩＶＥとＤＥＡＣＴＩＶＡＴＥのメッセージはＡＣＴＩＶＥ　ＴＯ　ＧＧＬＥメッセージとして考えてもよく、その意味（ＡＣＴＩＶＥあるいはＤＥＡＣＴＩＶＥ）はＡＣＴＩＶＡＴＥ　ＫＥＹが押下されるとき、オーディオ・ビデオ対話型プログラムの状態（それぞれＩＮＡＣＴＩＶＥまたはＡＣＴＩＶＥ）に依存する。

【００５２】

オーディオ・ビデオ対話型プログラムがＡＣＴＩＶＥ状態６３にて実行されると、その実行を中断したいときがある。例えば、非対話型コマーシャルが放送されるべきとき、送信されたオーディオとビデオは（図１の）デコーダ１０により生成される音やグラフィックスとは一致せず、視聴者が、通常通り、リモートコントロールを使用することができることが望ましい。しかしながら、アプリケーションのプログラマはそのような中断が必要となることを前もって知ることはできない。こうして、この場合、オーディオ・ビデオ対話型プログラムとは独立な放送機器はオーディオ・ビデオ対話型プログラム・コンポーネント・パケットサービス内に中断信号パケットと呼ばれる（上記のような）特別の信号パケットを繰り返し含めてもよい。このような各パケットは現在実行中のオーディオ・ビデオ対話型プログラムが実行を中断すべきであるということを示すデータを含んでいる。

【００５３】
ＦＬＯＷ＿ＭＥＳＳＡＧＥ　ＡＰＩコールを介して、システムローダは、そのようなパケットがオーディオ・ビデオ対話型プログラム・コンポーネント・パケットサービス内で認識されたときにはいつでもメッセージを受信する。例えば、中断信号パケットが受信されたとき、システムローダは、中断信号メッセージを受信し、最初の中断信号メッセージに応答してオーディオ・ビデオ対話型プログラムにＳＵＳＰＥＮＤメッセージを送り、そのプログラムは、実行を中断してＳＵＳＰＥＮＤＥＤ状態６５に入る。ＳＵＳＰＥＮＤＥＤ状態６５では、オーディオ・ビデオ対話型プログラムの実行は、それを、中断されたポイントから再び開始することができるように停止する。すなわち、オーディオ・ビデオ対話型プログラムを実行するために必要な資源の全てが割り当てられたままであり、オーディオ・ビデオ対話型プログラムの実行状態はＲＡＭ４１２内のあるロケーションに格納される。また、前に実行中の対話型プログラムが中断されたが、許可されたとき回復の用意ができていることを示す第２のロゴまたはアイコンが現在のビデオ画像上に重ね合わされる。

【００５４】
割り込み（例えば、非対話型コマーシャル）が終わったとき、放送機器はオーディオ・ビデオ対話型プログラム・コンポーネント・パケットサービス内に中断信号パケットを含めることを止める。システムローダは、中断信号メッセージを受信することなく予め決められた期間後、オーディオ・ビデオ対話型プログラムにＣＯＮＴＩＮＵＥメッセージを送り、そのプログラムは前に中断されたところから実行を再開して、上記のＡＣＴＩＶＥ状態６３に入る。

【００５５】
上述したＳＵＳＰＥＮＤ／ＣＯＮＴＩＮＵＥ信号構成の他の実施例は、放送機器がオーディオ・ビデオ対話型プログラムの実行を中断することを望むとき、オーディオ・ビデオ対話型プログラム・コンポーネント・パケットサービス内に単一の中断信号パケットを含めることである。その後、放送機器は、オーディオ・ビデオ対話型プログラムの実行を再開することが望まれるとき、オーディオ・ビデオ対話型プログラム・コンポーネント・パケットサービス内に継続信号パケットと呼ばれる他の特別な信号パケットを含める。このパケットは現在中断されているオーディオ・ビデオ対話型プログラムが実行を再開することを指令するデータを含む。システムローダは、継続信号パケットを認識し、オーディオ・ビデオ対話型プログラムにＣＯＮＴＩＮＵＥメッセージを送り、そのプログラムは実行を再開し、上述したようにＡＣＴＩＶＥ状態６３に入る。

【００５６】
視聴者が、中断されたオーディオ・ビデオ対話型プログラムの実行を停止することも可能である。プログラム中断ロゴまたはアイコンが表示されているとき、視聴者が、ＤＥＡＣＴＩＶＡＴＥメッセージを、中断されたオーディオ・ビデオ対話型プログラムに送り、そのプログラムは上述したＩＮＡＣＴＩＶＥ状態６１に入る。視聴者がＡＣＴＩＶＡＴＥ　ＫＥＹを押したとき、プログラムは、ＩＮＡＣＴＩＶＥ状態６１から実行を再開するだけであり、システムローダにオーディオ・ビデオ対話型プログラムに対してＡＣＴＩＶＡＴＥメッセージを送らせ、そのプログラムにＡＣＴＩＶＥ状態６３に入らせる。システムローダが依然として中断信号パケットを受信している場合は、別のＳＵＳＰＥＮＤメッセージが直ちにオーディオ・ビデオ対話型プログラムに送られ、そのプログラムは再びＳＵＳＰＥＮＤＥＤ状態６５に入る。ＩＮＡＣＴＩＶＥ状態６１と、ＡＣＴＩＶＥ状態６３と、ＳＵＳＰＥＮＤＥＤ状態６５は、オーディオ・ビデオ対話型プログラムがシステムローダから送られるメッセージに応答して切り替わる状態である。しかしながら、システムローダにより直接コントロールされて入る２つの他の状態が存在する。

【００５７】
オーディオ・ビデオ対話型プログラムはその実行の終了に到達することができる。例えば、放送機器は、実行終了信号パケットと呼ばれる他の特別な信号パケットをオーディオ・ビデオ対話型プログラム・コンポーネント・パケットサービスに含めてもよい。システム

ローダは、実行終了信号パケットがオーディオ・ビデオ対話型プログラム・コンポーネント・パケットサービス内に認識された時、ＦＬＯＷ＿ＭＥＳＳＡＧＥ　ＡＰＩコールを介して実行終了メッセージを受信する。実行終了メッセージに応答して、システムローダは、ＥＸＩＴメッセージをオーディオ・ビデオ対話型プログラムに送る。オーディオ・ビデオ対話型プログラムが、ＩＮＡＣＴＩＶＥ状態６１か、ＡＣＴＩＶＥ状態６３か、あるいはＳＵＳＰＥＮＤＥＤ状態６５のどの状態にあるかにかかわらず、オーディオ・ビデオ対話型プログラムは、その資源の割り振りを解除し、（図１の）デコーダ１０からそれ自体の全てのレコードを除去することによりＥＸＩＴメッセージに応答する。このプログラムはＨＡＬＴＥＤ状態６９に入ったと思い、デコーダ１０から消える。プログラムそれ自身がユーザコマンドを介してあるいはそれ自身の実行により、その実行が終了に達したことを認識することができる場合もある。オーディオ・ビデオ対話型プログラムがその実行の終了を認識したとき、ＥＸＩＴメッセージが受信された場合に行われる同じ処理を行い、それ自身によりＨＡＬＴＥＤ状態６９に入る。

【００５８】

オーディオ・ビデオ対話型プログラムがＳＵＳＰＥＮＤＥＤ状態にあるとき、別の対話型オーディオ・ビデオ対話型プログラムを、オーディオ・ビデオ対話型プログラム・コンポーネント・データフロー上で受信することが可能である。例えば、オーディオ・ビデオ対話型プログラムがコマーシャルにより中断されていれば、そのコマーシャルはそれ自身対話型プログラムであってもよいし、ユーザが別のオーディオ・ビデオ対話型プログラムを放送するチャンネルにチャンネルを変更してもよい。これら両方の場合は、新しいオーディオ・ビデオ対話型プログラムはディレクトリ・モジュールを含み、中断されたオーディオ・ビデオ対話型プログラムのディレクトリ・モジュールとは異なる。

【００５９】

システムローダは、ＤＩＲ＿ＳＰＹ　ＡＰＩコールを介して、ディレクトリがオーディオ・ビデオ対話型プログラム・コンポーネント・パケットサービス内に検出されるときにはいつでもメッセージを受信する。システムローダは、現在アクティブなディレクトリを検出されたばかりのディレクトリと比較する。システムローダが、別のディレクトリがオーディオ・ビデオ対話型プログラム・コンポーネント・パケットサービス内に存在すると認識すると、それはそのディレクトリにより表わされるオーディオ・ビデオ対話型プログラムのロードを始める。

【００６０】

最初に、メッセージが現在中断されているオーディオ・ビデオ対話型プログラムに送られ、プログラム・コンポーネント・パケットサービスはそのプログラムをもはや放送していないということ、あるいはプログラムは“フローを失った”ということを示す。このメッセージは現在実行しているプログラムがそれ自身を最小にするリクエストであり、すなわちＭＩＮＩＭＩＺＥメッセージである。ＭＩＮＩＭＩＺＥメッセージに応答して、現在中断しているオーディオ・ビデオ対話型プログラムは、先ずその現在の実行状態と環境を、持続時間とオーディオ・ビデオ対話型プログラムの識別子を含めて、ＲＡＭ４１２内の小さいブロック内に格納する。これらについては以下に説明する。それから、中断されたプログラムはその資源の割り振りを開始する。最小化されたオーディオ・ビデオ対話型プログラムはどんなコードも含まず、メッセージに応答して状態を変更することはできず、それ自身再スタートすることもできない。

【００６１】

システムローダは新たに検出されたディレクトリとオートスタートモジュールをロードし、新しいオーディオ・ビデオ対話型プログラムをＩＮＡＣＴＩＶＥ状態６１におき、先に説明したように対話型プログラムロゴまたはアイコンを表示する。視聴者は、ＡＣＴＩＶＡＴＥ　ＫＥＹを押すことにより、この新しいオーディオ・ビデオ対話型プログラムとの対話を開始あるいは停止することができ、そのプログラムをそれ自身中断してもよいし継続してもよい。

【００６２】

最小化プロセスは繰り返しプロセスである。例えば、この新しいオーディオ・ビデオ対話型プログラムは、中断された場合、さらに他のオーディオ・ビデオ対話型プログラムがオーディオ・ビデオ対話型プログラム・コンポーネント・パケットサービス内に検出された場合、最小化することができる。この場合、メモリの他のブロックが割り振られ、オーディオ・ビデオ対話型プログラムの実行状態と環境が、その識別子と継続時間と共に、このメモリブロックに格納される。その後先に説明したように、新たに検出されたプログラムの実行状態と環境を含む全てのメモリブロックを格納するのに必要なメモリ量によってのみ制限される。

【００６３】

新しいディレクトリ・モジュールをロードするか、あるいは前にロードされているディレクトリ・モジュールにより表わされるプログラムを実行するのに十分な使用可能なメモリ空間がない場合であって、しかも、最小化されたプログラムを表わす割り振られたメモリブロックがある場合には、システムローダは充分なメモリ空間を得るという意図で、（最も古いメモリブロックの割り振りを最初に解除するか、あるいは、元のアプリケーションにより拡張可能とマークされたメモリブロックの割り振りを最初に解除するというような）アルゴリズムにしたがって、メモリブロックの全て、あるいはいくらかの割り振りを自動的に解除することができる。あるいは、システムローダは、視聴者に最小化されたアプリケーションのリストを提供し、削除すべきアプリケーションを視聴者が選択することができるようにしてもよい。そして、選択された最小化アプリケーションを表すブロックは、充分なメモリ空間を得るため割り振りが解除される。

【００６４】

一方、前に最小化されたオーディオ・ビデオ対話型プログラムの実行状態と環境を含むメモリブロックは、メモリ内に割り振られたままである。上述したように、そのようなメモリブロック内には継続時間がある。あるブロック内の継続時間を超えたとき、前に最小化されたオーディオ・ビデオ対話型プログラムはタイムアウトする。この場合、そのプログラムはＨＡＬＴＥＤ状態６９に入ったと考えられ、実行状態と環境を含むメモリブロックは割り振りが解除され、前に最小化されたオーディオ・ビデオ対話型プログラムの全ての記録は失われる。

【００６５】

しかしながら、（２３の）デコーダ１０は、前に最小化されたオーディオ・ビデオ対話型プログラムのディレクトリと、コードモジュールと、データモジュールを含むオーディオ・ビデオ対話型プログラム・コンポーネントを再び受信することができるか、あるいはそのオーディオ・ビデオ対話型プログラムは“フローを再取得”することができる。例えば、対話型コマーシャルが終了し、ＨＡＬＴＥＤ状態６９に入ってもよいし、視聴者がこのチャンネルにチャンネルを戻してもよい。システムローダは、“新しい”ディレクトリをオーディオ・ビデオ対話型プログラム・コンポーネント・パケットサービス内にロードし始める。新しいディレクトリがロードされるといつでも、アプリケーション識別子とＲＡＭ４１２内に現在格納されている実行状態と環境を含む全てのブロック内の識別子とが比較される。一致するブロックが見つかった場合は、コードモジュールとデータモジュールがロードされ、オーディオ・ビデオ対話型プログラムはＩＮＡＣＴＩＶＥ状態６１におかれる。しかし、その実行状態は、それが最小化される直前の実行状態に更新される。視聴者がＡＣＴＩＶＡＴＥ　ＫＥＹを押すと、オーディオ・ビデオ対話型プログラムはＡＣＴＩＶＥ状態６３に入り、それが前に停止された場所から実行を開始する。このようにして、他のオーディオ・ビデオ対話型プログラムをランするためオーディオ・ビデオ対話型プログラムを一時的に停止してもよく、それから両方のプログラムのために充分な資源を必要とすることなくメモリ内に同時に残るように再開してもよい。

【００６６】

【発明の効果】

以上説明したように、本発明によれば、パケットストリームを受信するオーディオ・ビデオ対話型受信機において、オーディオ・ビデオ対話型プログラムを効率的に実行する制御

方法が得られる。

【図面の簡単な説明】

【図１】本発明に係るオーディオ・ビデオ対話型信号デコーダの一部を示すブロック図である。

【図２】図１に示す処理ユニット４０により実行されるソフトウェアの構造を示す図である。

【図３】オーディオ・ビデオ対話型プログラムのデータコンポーネントからモジュールを抽出することを理解する際に有用なフロー図とメモリレイアウト図である。

【図４】オーディオ・ビデオ対話型プログラムのデータコンポーネントからモジュールを抽出することを理解する際に有用な部分的にブロック形式の、また部分的にメモリレイアウト形式の図である。

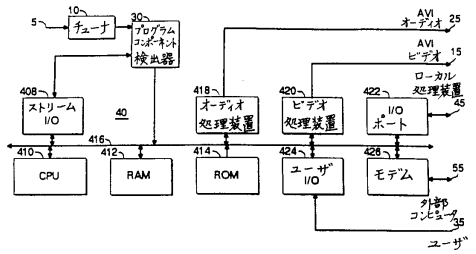
【図５】システムローダの初期化機能を示すフロー図である。

【図６】システムローダのデータストリーム監視機能を示す状態遷移図である。

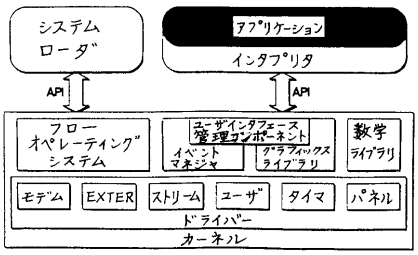
【符号の説明】

１０　チューナ
１５　オーディオ・ビデオ対話型ビデオ出力端子
２５　オーディオ・ビデオ対話型オーディオ出力端子
３０　プログラム・コンポーネント検出器
３２　データパケットＤＭＡコントローラ
３４　ヘッダパケットＤＭＡコントローラ
４０　処理ユニット
４５　ローカル処理装置
５５　外部コンピュータ
４０８　ストリームＩ／Ｏ
４１０　ＣＰＵ
４１２　ＲＡＭ
４１４　ＲＯＭ
４１８　オーディオ処理装置
４２０　ビデオ処理装置
４２２　Ｉ／Ｏポート
４２６　モデム

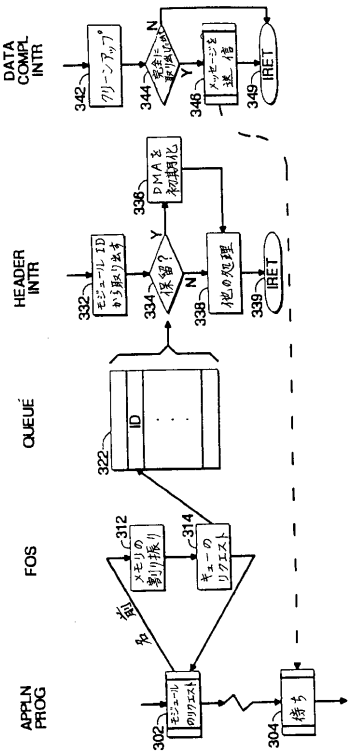
【図１】



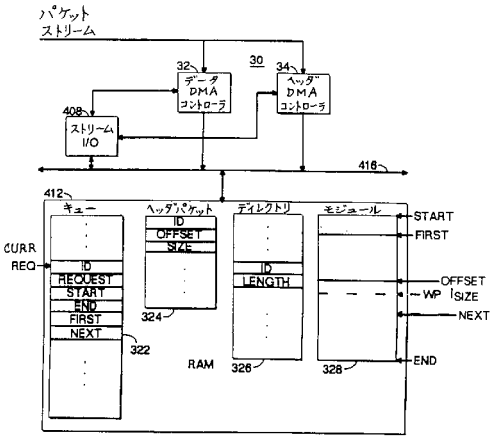
【図２】



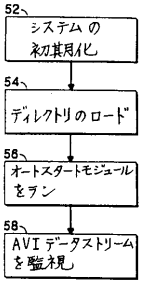
【図３】



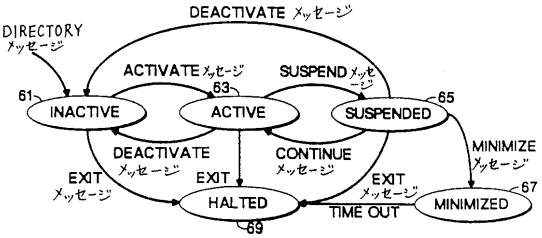
【図４】



【図５】



【図６】



フロントページの続き

(74)代理人 100096194
弁理士 竹内 英人
(74)代理人 100074228
弁理士 今城 俊夫
(74)代理人 100084009
弁理士 小川 信夫
(74)代理人 100082821
弁理士 村社 厚夫
(74)代理人 100086771
弁理士 西島 孝喜
(74)代理人 100084663
弁理士 箱田 篤
(72)発明者 ジーン－レネ メナンド
アメリカ合衆国 カリフオルニア州 マリナ・デル・レイ パラワン・ウエイ １４００１
(72)発明者 アラン デルパツク
アメリカ合衆国 カリフオルニア州 ロスアンゼルスパーネル・アベニユー ２２２１

審査官 川崎 優

(56)参考文献 菊池ほか，次世代通信網の光と陰，日経コミュニケーション，１９９４年 4月 4日，第171号，P.40-58
永松ほか，分散マルチメディア環境を実現するＡＶサーバ歌舞伎，情報処理学会研究報告，１９９３年 7月 9日，Vol.93,No.58，P.83～90
南部ほか，分散マルチメディア・プラットフォームにおけるアプリケーションとプロトコルのアーキテクチャ，電子情報通信学会技術研究報告，１９９３年１０月 8日，Vol.93,No.263，P.1-8

(58)調査した分野(Int.Cl.，ＤＢ名)
G06F 9/46-54
H04N 7/16-173
H04N 7/26