(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号

特許第6428689号
(P6428689)

(45)発行日　平成30年11月28日(2018.11.28)　(24)登録日　平成30年11月9日(2018.11.9)

(51)Int.Cl.			F I		
G10H	7/02	(2006.01)	G10H	7/02	
G10H	1/00	(2006.01)	G10H	1/00	Z

請求項の数 13　(全 35 頁)

(21)出願番号　特願2016-58773(P2016-58773)
(22)出願日　平成28年3月23日(2016.3.23)
(65)公開番号　特開2017-173522(P2017-173522A)
(43)公開日　平成29年9月28日(2017.9.28)
審査請求日　平成29年11月22日(2017.11.22)

(73)特許権者　000001443
カシオ計算機株式会社
東京都渋谷区本町１丁目６番２号
(74)代理人　100074099
弁理士　大菅　義之
(72)発明者　佐藤　博毅
東京都羽村市栄町３丁目２番１号　カシオ計算機株式会社羽村技術センター内

審査官　上田　雄

最終頁に続く

(54)【発明の名称】波形読込み装置、方法、プログラム、及び電子楽器

(57)【特許請求の範囲】
【請求項１】
１つの音色に対応する複数の波形データを含み、かつ、複数のサイズの波形データが含まれる波形データのセットを音色波形データとして記憶している一次記憶装置から、指定された音色に対応する前記音色波形データを取得する取得処理と、
前記取得した前記音色波形データに含まれる複数の波形データのそれぞれを複数のグループのいずれかに割り当てる処理であって、それぞれのグループに含まれる波形データのサイズが第１のサイズを超えないようにして１または複数の波形データを組み合わせて各グループに割り当てる割当処理と、
それぞれのセグメントのサイズが前記第１のサイズ以上になるように複数のセグメントに分割された記憶領域を有する二次記憶装置に対して、前記割り当てを行った複数のグループの波形データを、同じグループに含まれる複数の波形データが同じセグメントに格納されるようにして書き込む書込み処理と、
を実行する処理部を備えた波形読込み装置。
【請求項２】
前記書込み処理は、前記二次記憶装置に対して、前記複数のグループの波形データを書き込む場合に、１つのセグメントには、１つのグループに含まれる全ての波形データが格納され、他のグループに含まれる波形データが混在して格納されないようする、
請求項１に記載の波形読込み装置。
【請求項３】

前記一次記憶装置は、複数の音色に対応する複数の前記音色波形データを記憶しており、

前記第１のサイズは、前記音色波形データとして前記一次記憶装置に記憶されている複数の波形データのうちの最も大きいサイズの波形データが格納可能であり、かつ、前記音色波形データとして前記一次記憶装置に記憶されている複数の波形データのうちの少なくとも一部の複数の波形データを組み合わせて格納可能であるようにして決められたサイズである、

請求項１または２に記載の波形読込み装置。

【請求項４】

前記割当処理は、前記音色波形データに含まれる複数の波形データのそれぞれが、前記複数のグループのうちのいずれのグループに属するかを示すグループ情報を記憶する記憶手段から、前記グループ情報を取得するグループ情報取得処理を含み、前記取得したグループ情報に基づいて、前記音色波形データに含まれる複数の波形データを複数のグループに分割して割り当てる、

請求項１乃至３のいずれかに記載の波形読込み装置。

【請求項５】

前記複数のセグメントは、全てが同じ前記第１のサイズである、

請求項１乃至４のいずれかに記載の波形読込み装置。

【請求項６】

前記一次記憶装置は、フラッシュメモリを備え、二次記憶装置は、一次記憶装置より容量の小さいランダムアクセスメモリを備えた、請求項１乃至５のいずれかに記載の波形読込み装置。

【請求項７】

複数の音色のいずれかに対応する前記音色波形データを指定する指定処理と、

前記指定処理により指定された前記音色波形データが前記二次記憶装置に記憶されているか否かを判別する判別処理と、

前記指定された前記音色波形データが前記二次記憶装置に記憶されていないと判別された場合に、前記音色波形データを分割するグループの数に対応する数のセグメントを前記二次記憶装置内で確保するセグメント確保処理

を実行し、

前記書込み処理は、前記セグメント確保処理により確保された複数のセグメントのそれぞれに、前記音色波形データを分割した複数のグループのそれぞれに含まれる波形データを書き込む、

請求項１乃至６のいずれかに記載の波形読込み装置。

【請求項８】

前記二次記憶装置が有する複数のセグメントのうち、空きのセグメント数を取得する取得処理

を実行し、

前記セグメント確保処理は、前記取得された空きのセグメント数が前記指定された前記音色波形データのグループ数と同じあるいはそれ以上の場合は、前記指定された前記音色波形データのグループ数と同じ数の空きのセグメントを確保し、前記取得された空きのセグメント数が前記指定された前記音色波形データのグループ数より少ない場合は、前記空きのセグメントと、空いていないセグメントの中で予め定められた条件を満たすセグメントとを確保する処理を実行する、請求項７に記載の波形読込み装置。

【請求項９】

前記予め定められた条件を満たすセグメントは、前記二次記憶装置に記憶されている前記音色波形データのうち、記憶されている期間が最も長い前記音色波形データを記憶しているセグメントである、請求項８に記載の波形読込み装置。

【請求項１０】

前記書込み処理は、前記二次記憶装置に対して、複数の音色の波形データを書き込む場

合に、１つのセグメントには、異なる音色の波形データが混在して格納されないようする
ン
請求項１乃至９のいずれかに記載の波形読込み装置。
【請求項１１】
請求項１乃至１０のいずれかに記載の波形読込み装置と、
前記一次記憶装置と、
前記二次記憶装置と、
前記二次記憶装置に書き込まれた前記波形データから楽音を生成する音源と、
を備えた電子楽器。
【請求項１２】
波形読込み装置に用いられる波形読込み方法であって、前記波形読込み装置が、
１つの音色に対応する複数の波形データを含み、かつ、複数のサイズの波形データが含まれる波形データのセットを音色波形データとして記憶している一次記憶装置から、指定された音色に対応する前記音色波形データを取得する取得処理と、
前記取得した前記音色波形データに含まれる複数の波形データのそれぞれを複数のグループのいずれかに割り当てる処理であって、それぞれのグループに含まれる波形データのサイズが第１のサイズを超えないようにして１または複数の波形データを組み合わせて各グループに割り当てる割当処理と、
それぞれのセグメントのサイズが前記第１のサイズ以上になるように複数のセグメントに分割された記憶領域を有する二次記憶装置に対して、前記割り当てを行った複数のグループの波形データを、同じグループに含まれる複数の波形データが同じセグメントに格納されるようにして書き込む書込み処理と、
を実行する波形読込み方法。
【請求項１３】
波形読込み装置として用いられるコンピュータに、
１つの音色に対応する複数の波形データを含み、かつ、複数のサイズの波形データが含まれる波形データのセットを音色波形データとして記憶している一次記憶装置から、指定された音色に対応する前記音色波形データを取得する取得処理と、
前記取得した前記音色波形データに含まれる複数の波形データのそれぞれを複数のグループのいずれかに割り当てる処理であって、それぞれのグループに含まれる波形データのサイズが第１のサイズを超えないようにして１または複数の波形データを組み合わせて各グループに割り当てる割当処理と、
それぞれのセグメントのサイズが前記第１のサイズ以上になるように複数のセグメントに分割された記憶領域を有する二次記憶装置に対して、前記割り当てを行った複数のグループの波形データを、同じグループに含まれる複数の波形データが同じセグメントに格納されるようにして書き込む書込み処理と、
を実行させるプログラム。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、波形読込み装置、方法、プログラム、及びその装置を用いた電子楽器に関する。
【背景技術】
【０００２】
波形読込み方式により楽音波形を発生する音源装置では、より多数の、より長時間の波形データを利用できるようにするために、使用しない波形データは例えばＲＯＭ（一次記憶装置)に保存しておき、使用する波形データのみを音源装置が直接アクセスできる波形メモリとして機能するＲＡＭ（二次記憶装置）に転送して発音させるというシステムを採用するものがある。つまり、高価なＲＡＭが有する記憶容量以上の記憶容量の波形データを安価なＲＯＭに保持しておき、必要な場合のみ移動して発音に使用するというコスト的

には効率的な方法であると言える。
【０００３】
このようなシステムの場合、ＲＯＭからＲＡＭ上に新たな音色波形を読み込む際に空きのエリアが不足した場合は、別の音色の波形が読み込まれているＲＡＭ上の記憶領域に上書きを行うことになる。
【０００４】
このような制御を行うための従来技術として、次のような技術が知られている（例えば特許文献１に記載の技術）。ＲＯＭには、音色ごとに１つ以上の波形データが格納されている。音源ＬＳＩ（大規模集積回路）は、指定された曲の曲データを参照して、それらの波形データのなかで楽音の発音に必要なものを特定し、必要と特定した波形データはそのなかで必要な部分を更に特定する。それにより、楽音の発音に必要な波形データはその必要な部分のみをＲＯＭから読み出してＲＡＭに転送し格納させる。これにより、発音させるべき楽音の波高値生成用に波形データをＲＡＭに格納する場合に、そのデータ量をより抑えることが可能な楽音発生装置を提供するものである。
【先行技術文献】
【特許文献】
【０００５】
【特許文献１】特開２００７－２７１８２７号公報
【発明の概要】
【発明が解決しようとする課題】
【０００６】
ところで、一般に１つの音色で複数の波形データを持つケースでも、１つの波形データは分断してＲＡＭ上に配置することは音源ＬＳＩの読出し構造上許されず、必ず連続して配置される必要がある。つまり、新たに音色を選択してＲＡＭ上に配置する際には各波形データが連続して配置されるだけの空きエリアを用意する必要がある。
【０００７】
しかしながら、従来技術では、波形の読込みや上書きを繰り返すことにより、ＲＡＭ上での波形の使用領域と未使用領域の配置は無秩序になり、これによって発生する問題として、ＲＡＭ上に読み込まれた不要な音色波形を捨てて容量的に十分と思われる空きエリアを確保しても、連続性が無いと波形が読み込めないケースがあることである。例えば４００ＫＢ（キロバイト）の空きエリアが１０箇所あっても、５００ＫＢの波形をロードすることはできない。そのため、さらなるＲＡＭ上の音色波形を犠牲にして、所望の音色の波形の読込みを行う必要が生ずる。これではＲＡＭの使用効率が非常に悪くなるケースが存在するばかりでなく、同じ音色の波形がＲＡＭ上に存在したとしても、ある音色を読み込めるのかどうか予想がつかないという使いにくさも生じる。例えばＡ、Ｂ、Ｃという音色がＲＡＭ上にある場合に、新たにＤという音色をＲＯＭからＲＡＭに読み込もうとした場合、或るケースではそのまま読み込め、他のケースではＡを上書きする必要があり、更に他のケースではＡ、Ｂを上書きする必要があるなど、予想がつかない。
【０００８】
一般のコンピュータ上などでも同様に一次記憶装置、二次記憶装置を持つため、同様の問題が発生するが、これを解決するために、各ファイルが使用している記憶領域を一箇所に集める、いわゆるデフラグメンテーションあるいはガベージコレクションという処理もあり、楽器のように常にリアルタイム性を要求されないコンピュータの世界では、ユーザがそれを意識的に行うこともある。
【０００９】
しかしながら、これらの処理は非常に時間を要する処理なので、電子楽器で音色選択したような時に実行できるものではないという課題があった。かといって演奏していない時にユーザが実行するというのもユーザに余計な精神的負担を課することになり、楽器として好ましいものではないという課題があった。
【００１０】

そこで、本発明は、二次記憶装置上に波形が存在しない音色が選択されて新たな音色波形データを一次記憶装置から二次記憶装置に読み込む際に、確実に新しい波形を読込み可能とすることにより、読み込みたい波形の容量よりもはるかに大きな空きエリアがありながら読み込めないという非効率な状態を回避することを目的とする。

【課題を解決するための手段】

【００１１】

態様の一例では、１つの音色に対応する複数の波形データを含み、かつ、複数のサイズの波形データが含まれる波形データのセットを音色波形データとして記憶している一次記憶装置から、指定された音色に対応する前記音色波形データを取得する取得処理と、前記取得した前記音色波形データに含まれる複数の波形データのそれぞれを複数のグループのいずれかに割り当てる処理であって、それぞれのグループに含まれる波形データのサイズが第１のサイズを超えないようにして１または複数の波形データを組み合わせて各グループに割り当てる割当処理と、それぞれのセグメントのサイズが前記第１のサイズ以上になるように複数のセグメントに分割された記憶領域を有する二次記憶装置に対して、前記割り当てを行った複数のグループの波形データを、同じグループに含まれる複数の波形データが同じセグメントに格納されるようにして書き込む書込み処理と、を実行する処理部を備える。

【発明の効果】

【００１２】

本発明によれば、二次記憶装置上に波形が存在しない音色が選択されて新たな音色波形データを一次記憶装置から二次記憶装置に読み込む際に、確実に新しい波形を読込み可能となり、読み込みたい波形の容量よりもはるかに大きな空きエリアがありながら読み込めないという非効率な状態を回避することが可能となる。

【図面の簡単な説明】

【００１３】

【図１】本発明による電子鍵盤楽器の実施形態の外観図である。

【図２】電子鍵盤楽器の実施形態のハードウェア構成例を示す図である。

【図３】本実施形態における波形データセグメントの概念を説明する図である。

【図４】フラッシュメモリ音色情報テーブルと音色選択優先度テーブルのデータ構成例を示す図である。

【図５】音色ごとの波形データの大容量フラッシュメモリから波形メモリへの転送動作を説明する図である。

【図６】波形スプリットの説明図である。

【図７】ＲＯＭに記憶される構造体の定数データの例を示す図である。

【図８】ＲＡＭに記憶される変数データの例を示す図である。

【図９】制御処理の全体処理の例を示すメインルーチンのフローチャートである。

【図１０】初期化処理の詳細例を示すフローチャートである。

【図１１】音色切替え処理の詳細例を示すフローチャートである。

【図１２】ＲＡＭ波形データ調査ルーチン及び波形領域確保ルーチンの詳細例を示すフローチャートである。

【図１３】空きセグメント数調査ルーチンの詳細例を示すフローチャートである。

【図１４】波形領域開放ルーチンの詳細例を示すフローチャートである。

【図１５】波形読込み指示ルーチンの詳細例を示すフローチャートである。

【図１６】波形読込み処理の詳細例を示すフローチャートである。

【図１７】波形転送ルーチンの詳細例を示すフローチャートである。

【図１８】押鍵処理の詳細例を示すフローチャートである。

【図１９】スプリット波形検索ルーチンの詳細例を示すフローチャートである。

【図２０】セグメント内波形検索ルーチンの詳細例を示すフローチャートである。

【発明を実施するための形態】

【００１４】

以下、本発明を実施するための形態（以下「本実施形態」と記載する）について図面を参照しながら詳細に説明する。本実施形態は、例えば電子鍵盤楽器に適用される、音高（鍵域）や音量（ベロシティ：打鍵の速さ）などの演奏情報によって音色が変化するのを再現するために、大容量の一次記憶装置（例えばフラッシュメモリ）から二次記憶装置（例えばＲＡＭによる波形メモリ）に、音高又は音量ごとの波形データ（以下これを「スプリット波形」と呼ぶ）を読み込む楽音発生装置を対象とする。本実施形態は、このような楽音発生装置において、二次記憶装置上に波形が存在しない音色が選択されて新たな音色波形データを一次記憶装置から二次記憶装置に読み込む際に、確実に新しい波形を読込み可能となり、読み込みたい波形の容量よりもはるかに大きな空きエリアがありながら読み込めないという非効率な状態を回避することができる電子楽器を実現するものである。

【００１５】

図１は、本発明による電子鍵盤楽器の実施形態の外観図である。本実施形態は、電子鍵盤楽器１００として実施される。電子鍵盤楽器１００は、演奏操作子としての複数の鍵からなる鍵盤１０１と、音色選択操作子としての音色選択を行うための音色選択ボタン１０２及び音色以外の各種機能選択を行う機能選択ボタン１０３からなるスイッチ・パネルと、ピッチベンドやトレモロ、ビブラート等の各種モジュレーション（演奏効果）を付加するベンダ／モジュレーション・ホイール１０４、音色や音色以外の各種設定情報を表示するＬＣＤ（Ｌｉｑｕｉｄ　Ｃｒｙｓｔａｌ　Ｄｉｓｐｌａｙ：液晶ディスプレイ）１０５等を備える。また、電子鍵盤楽器１００は、特には図示しないが、演奏により生成された楽音を放音するスピーカを裏面部、側面部、又は背面部等に備える。

【００１６】

音色選択ボタン１０２は、図１に示されるように、ピアノ（図中「Ｐｉａｎｏ」）、エレクトリックピアノ（図中「Ｅ．ｐｉａｎｏ」）、オルガン（図中「Ｏｒｇａｎ」）、ギター（図中「Ｇｕｉｔａｒ）等の各種音色のカテゴリを選択するためのボタン群である。ユーザは、この音色選択ボタン１０２を押下することにより、例えば１６音色のうちの何れかを選択することができる。

【００１７】

図２は、図１の電子鍵盤楽器１００の実施形態のハードウェア構成例を示す図である。図２において、電子鍵盤楽器１００は、ＣＰＵ（中央演算処理装置）２０１、ＲＯＭ（リードオンリーメモリ）２０２、ＲＡＭ（ランダムアクセスメモリ）２０３、大容量フラッシュ（Ｆｌａｓｈ）メモリ２０４、波形メモリ２０６が接続される音源ＬＳＩ（大規模集積回路）２０５、図１の鍵盤１０１と図１の音色選択ボタン１０２及び機能選択ボタン１０３からなるスイッチ・パネルとが接続されるキー・スキャナ２０７、図１のベンダ／モジュレーション・ホイール１０４が接続されるＡ／Ｄコンバータ２０８、図１のＬＣＤ１０５が接続されるＬＣＤコントローラ２０９、及びＭＩＤＩ（Ｍｕｓｉｃａｌ　Ｉｎｓｔｒｕｍｅｎｔ　Ｄｉｇｉｔａｌ　Ｉｎｔｅｒｆａｃｅ）入力を受け付けるＭＩＤＩ　Ｉ／Ｆ（インタフェース）２１３が、それぞれシステムバス２１４に接続される構成を備える。また、音源ＬＳＩ２０５から出力されるデジタル楽音波形データは、Ｄ／Ａコンバータ２０８によりアナログ楽音波形信号に変換され、アンプ２１１で増幅された後に、特には図示しないスピーカ又は出力端子から出力される。

【００１８】

ＣＰＵ２０１は、ＲＡＭ２０３をワークメモリとして使用しながらＲＯＭ２０２に記憶された制御プログラムを実行することにより、図１の電子鍵盤楽器１００の制御動作を実行する。また、ＲＯＭ２０２は、上記制御プログラム及び各種固定データを記憶する。

【００１９】

大容量フラッシュメモリ２０４（一次記憶装置）は、複数の音色を構成する波形データ等の大容量データの格納領域であり、シーケンシャルアクセスにより順次アクセスされる。一方、音源ＬＳＩ２０５には、波形データを展開するＲＡＭで構成される波形メモリ２０６（二次記憶装置）が接続され、発音される楽音の波形データは必ず、この波形メモリ２０６上に配置されている必要がある。ＣＰＵ２０１は、ユーザが図１の音色選択ボタン

１０２を操作することにより指定した音色に対応する１組以上の波形データを大容量フラッシュメモリ２０４からシーケンシャルに読みだし、それを音源ＬＳＩ２０５経由で波形メモリ２０６に転送することで、音色データを入れ替えることができる。

【００２０】

ＬＣＤコントローラ２０９は、ＬＣＤ１０５を制御するＩＣ（集積回路）である。キー・スキャナ２０７は、鍵盤１０１や音色選択ボタン１０２又は機能選択ボタン１０３等のスイッチ・パネルの状態を走査して、ＣＰＵ２０１に通知するＩＣである。Ａ／Ｄコンバータ２０８は、ベンダ／モジュレーション・ホイール１０４の操作位置を検出する集積回路である。

【００２１】

以上の構成を有する電子鍵盤楽器１００の実施例における動作仕様の概要についてまず説明する。

【００２２】

まず、ユーザインタフェースについて説明する。本実施形態における電子鍵盤楽器１００では、ユーザが図１の音色選択ボタン１０２を押下することで、例えば１６の音色のうち、いずれかを選択して演奏することができる。

【００２３】

次に、波形データの読込み転送動作について説明する。図３は、本実施形態における波形データセグメントの概念を説明する図である。波形メモリ（ＲＡＭ）上でセグメント管理がされない従来の電子鍵盤楽器では、データ配置３０１として示されるように、波形データの読込が繰り返されると、使用領域と空き領域が無秩序な配置になり、たとえ合計では十分な空き領域があったとしても、連続性が無いばかりに読み込むエリアが確保できないというケースが起こりえていた。

【００２４】

これに対して本実施形態では、３０２として示されるデータ配置を有する。このデータ配置においては、波形メモリ２０６の記憶領域において、一定の間隔で跨ぐことができない境界線が設けられる。本実施形態において、この隣り合う境界線と境界線の間の領域をセグメントと呼ぶ。この１セグメントのデータサイズは、全て同じであり、各音色を構成する１つ以上の波形データの中で最もサイズが大きいものよりも下回らないように設定される。本実施形態では、大容量フラッシュメモリ２０４から波形メモリ２０６への波形データの読込みは、このセグメントを単位として実行される。本実施形態において、１つの音色は１つ又は複数の波形データを持つ。１つの音色の１つ以上の波形データの読込みにおいては、１つ又は複数のセグメント単位で波形メモリ２０６の記憶領域が確保され、読込が実行される。この場合に、確保された複数のセグメントは、連続して配置されている必要はない。また、１つのセグメントには、同じ音色の波形データを複数読み込むことができるが、異なる音色の波形データが混在して読み込まれないように、制御が実行される。

【００２５】

大容量フラッシュメモリ２０上には例えば全１６音色分の波形データ群が記録されており、その中からユーザが指定した音色を構成する１つ以上の波形データを音源ＬＳＩ２０５の波形メモリ２０６に読み込むことで、発音が可能となる。この音色情報は、図２のＲＯＭ２０２に記憶されるフラッシュメモリ音色情報テーブルによって管理される。図４（ａ）は、フラッシュメモリ音色情報テーブルのデータ構成例を示す図である。図４（ａ）の表として例示されるフラッシュメモリ音色情報テーブルの２行目以降の各行として示される各音色のエントリには、図４（ａ）の表の１行目に示されるように、音色番号を示す「番号」項目値と、「音色名」項目値と、大容量フラッシュメモリ２０４の波形記憶領域の先頭からの記憶アドレスのオフセット（１６進数）を示す「波形アドレスオフセット」項目値と、その音色に含まれる波形データ群の合計の波形サイズ（１６進数）を示す「波形サイズ」項目値と、それらの波形データ群が使用する「使用セグメント数」項目値とが記憶されている。

【００２６】
上記１６音色の各々は、１音色あたり最大で例えば６４種類の波形データから構成され、大容量フラッシュメモリ２０４に連続的に格納されている。音色内の各波形データのうち、波形メモリ２０６内の１つのセグメント内に転送される１つ以上の波形データは１つのグループにグルーピングされる。波形メモリ２０６内の１つのセグメント内に転送される波形データは、大容量フラッシュメモリ２０４上でも必ず連続している。１つの音色の各波形データの情報は、図２のＲＯＭ２０２に記憶される音色波形情報テーブルによって管理される。図４（ｂ）は、１音色あたり最大で６４の波形データを管理する音色波形情報テーブルのデータ構成例を示す図である。図４（ｂ）の表として例示される音色波形情報テーブルの２行目以降の各行として示される各波形データのエントリには、図４（ｂ）の表の１行目に示されるように、０から６３までの「波形番号」項目値が記憶される。またこのエントリには、演奏時に演奏されたキーやベロシティ（強度）によってどの波形を読みだして発音させるかを判断するための情報（スプリットゾーンパラメータ）として、「最低ベロシティ」項目値、「最高ベロシティ」項目値、「最低キー番号」項目値、及び「最高キー番号」項目値が記憶される。またこのエントリには、その波形データが転送先の波形メモリ２０６のセグメントのどのアドレスから記憶されているかを示す「セグメント先頭からのアドレス」項目値と、その波形データのサイズを示す「波形サイズ」項目値が記憶される。なお、値の左端に付与されている記号「Ｈ」は、その値が１６進数であることを示す。更にこのエントリには、その波形データが含まれるグループを示す「セグメントグループ」項目値が記憶される。

【００２７】
本実施形態においては、ユーザによって指定された音色を構成する波形データ群が波形メモリ２０６上に記憶されていない場合には、次の制御処理が実行される。まず、セグメントを単位として、大容量フラッシュメモリ２０４から波形メモリ２０６に新たに読み込まれる音色を構成する１つ以上の波形データ群が必要とする数のセグメントの空き領域が、波形メモリ２０６上で探索されて確保される。必要なセグメント数は、ＲＯＭ２０２に記憶されている図４（ａ）に例示されるフラッシュメモリ音色情報テーブルにおいて、指定された音色に対応するエントリの「使用セグメント数」項目値が参照されることにより決定される。波形メモリ２０６上で必要な数の空きセグメントが無い場合には、必要なセグメント数が確保されるまで、古い順に使用されていた音色に対応するセグメントが順次開放されて確保される。

【００２８】
次に、大容量フラッシュメモリ２０４に記憶されている指定された音色に対応する波形データ群が、波形メモリ２０６上で確保された空きセグメントに順次転送される。このとき、指定された音色に対応するＲＯＭ２０２に記憶されている図４（ｂ）に例示される音色波形情報テーブルにおいて、「セグメントグループ」項目値が（０から）小さい順に同じ値を有する波形データ群は、波形メモリ２０６上の（番号が小さい順の）１つの空きセグメントに転送される。図５は、音色ごとの波形データの大容量フラッシュメモリ２０４から波形メモリ２０６への転送動作を説明する図である。例えば図５（ａ）に示されるＧｕｉｔａｒ音色を構成する大容量フラッシュメモリ２０４上の０番から１０番の波形データ群が図５（ｂ）の波形メモリ２０６に転送される場合、「セグメントグループ」項目値＝０である０番から２番までの連続する３組の波形データ群は図５（ｂ）の波形メモリ２０６上の空きセグメント＝Ｓｅｇ０に転送され、「セグメントグループ」項目値＝１である３番から５番までの連続する３組の波形データ群は波形メモリ２０６上の空きセグメント＝Ｓｅｇ４に転送され、「セグメントグループ」項目値＝２である６番から８番までの連続する３組の波形データ群は波形メモリ２０６上の空きセグメント＝Ｓｅｇ６に転送され、「セグメントグループ」項目値＝３である９番と１０番の連続する２組の波形データ群は波形メモリ２０６上の空きセグメント＝Ｓｅｇ６３に転送される。一方例えば、例えば図５（ｃ）に示されるＳａｘｏｐｈｏｎｅ音色を構成する大容量フラッシュメモリ２０４上の０番から８番の波形データ群が図５（ｂ）の波形メモリ２０６に転送される場合、

「セグメントグループ」項目値＝０である０番から２番までの連続する３組の波形データ群は図５（ｂ）の波形メモリ２０６上の空きセグメント＝Ｓｅｇ２に転送され、「セグメントグループ」項目値＝１である３番から５番までの連続する３組の波形データ群は波形メモリ２０６上の空きセグメント＝Ｓｅｇ３に転送され、「セグメントグループ」項目値＝２である６番と７番の連続する２組の波形データ群は波形メモリ２０６上の空きセグメント＝Ｓｅｇ５に転送され、「セグメントグループ」項目値＝３である８番の１組の波形データ群は波形メモリ２０６上の空きセグメント＝Ｓｅｇ７に転送される。転送先のセグメントに対応付けて、そのセグメントに転送された波形データに対応する「セグメントグループ」項目値が、例えばＲＡＭ２０３（図２）に記憶される。

【００２９】

各セグメントは、前述したように最もサイズが大きい波形データに合わせた、例えば１００００００Ｈ（「Ｈ」は１６進数を示す）バイト＝１ＭＢ（メガバイト）のサイズを有する。従って、１つの波形データは必ず、１つのセグメント内の記憶領域に連続して転送することが可能となる。合計で例えば１ＭＢ以下になる同じ「セグメントグループ」項目値を有する連続する複数の波形データも、１つのセグメントに転送することができる。これにより、偶然性に依存して空き容量よりも遥かに小さいサイズの波形データが読み込めないということは起こらなくなる。

【００３０】

ユーザによって指定された音色を構成する波形データ群が波形メモリ２０６上に記憶されている場合には、それらの波形データ群は大容量フラッシュメモリ２０４からあらためて読み込む必要は無いので、そのまま発音処理に使用することができる。この際、読込みは発生しないものの、履歴上は最新の音色として記録される。

【００３１】

図６は、押鍵が発生した場合の発音処理時の波形スプリットの説明図である。本実施形態では、図６に例示されるように、演奏時のキー番号とベロシティ（演奏強さ）が最大で６４領域に２次元的に分割され、それぞれの分割された領域（スプリットエリア）に、上述のように波形メモリ２０６に転送された１音色あたり１組以上最大で６４組の波形データがそれぞれ割り当てられる。図２において、ＣＰＵ２０１は、キー・スキャナ２０７を介して鍵盤１０１における押鍵の発生を検出すると、現在演奏中の音色に対応するＲＯＭ２０２（図２）内の図４（ｂ）に例示される音色波形情報テーブルのエントリのうち、押鍵発生時のキー番号値が「最低キー番号」項目値から「最高キー番号」項目値までの範囲に含まれ、押鍵発生時のベロシティが「最低ベロシティ」項目値から「最高ベロシティ」項目値までの範囲に含まれるエントリが検索される。そして、検索された当該エントリの「セグメントグループ」項目値と同じセグメントグループの値が対応付けられている波形メモリ２０６上のセグメント内の、当該エントリの「セグメント先頭からのアドレス」項目値に対応するアドレスから、当該エントリの「波形サイズ」項目値に対応する波形サイズ分だけの波形データが、押鍵発生時のキー番号に対応する速度で読み出されて押鍵発生時のベロシティに対応する強度で発音される。

【００３２】

ＣＰＵ２０１は、ユーザによって指定された音色を構成する波形データ群が波形メモリ２０６上に記憶されていない場合には、これらの波形データ群が大容量フラッシュメモリ２０４から波形メモリ２０６に転送されるが、この波形転送はバックグラウンド処理として１組の波形データごとに分割して実行される。従って、押鍵発生時にＣＰＵ２０１がスプリットエリアを判別してそのエリアに対応する波形データの波形メモリ２０６からの読出しを音源ＬＳＩ２０５に指示したときに、該当する波形データが大容量フラッシュメモリ２０４から波形メモリ２０６への転送の最中である場合には、まだ波形データが波形メモリ２０６上に揃っていないため、音源ＬＳＩ２０５は発音を行わない。

【００３３】

ＲＯＭ２０２（図２）に記憶される図４（ａ）の例のフラッシュメモリ音色情報テーブル及び図４（ｂ）の例の音色波形情報テーブルは、ＣＰＵ２０１（図２）が実行する制御

プログラム上からは、構造体の定数データとしてアクセスすることができる。図７は、ＲＯＭ２０２に記憶される構造体の定数データの例を示す図である。

【００３４】

まず、図４（ａ）に例示されるフラッシュメモリ音色情報テーブルは、図７に示される、構造体ＴＯＮＥ＿ＩＮＦ［１６］（［］内の数字は配列数を示す）と、その構造体を構成するメンバ定数である配列定数ＮＡＭＥ［１６］（［］内の数字は配列数を示す）、定数ＷＡＶＥ＿ＴＯＴＡＬ＿ＳＩＺＥ、及び定数ＮＵＭ＿ＳＥＧとによって構成される。構造体ＴＯＮＥ＿ＩＮＦ［１６］は例えば、ＲＡＭ２０３（図２）上の変数Ｔによって、ＴＯＮＥ＿ＩＮＦ［Ｔ］としてアクセスされる。変数Ｔが例えば０から１５までの値域をとることにより、図４（ａ）のフラッシュメモリ音色情報テーブルにおける「番号」項目値が０から１５までの各音色のエントリが指定される。メンバ定数ＮＡＭＥ［１６］は、図４（ａ）の例の「音色名」項目値に対応する１６文字までの音色表示名を格納する。このメンバ定数へは例えば、ＲＡＭ２０３上の変数ｉによって、ＴＯＮＥ＿ＩＮＦ［Ｔ］．ＮＡＭＥ［ｉ］というフォーマットでアクセスされる。変数ｉが例えば０から１５までの値域をとることにより、変数Ｔの値に対応する音色の音色表示名（０～１５文字目）が表される。メンバ定数ＷＡＶＥ＿ＴＯＴＡＬ＿ＳＩＺＥは、図４（ａ）の「波形サイズ」項目値に対応する音色内の波形データの合計サイズを格納する。このメンバ定数へは例えば、ＴＯＮＥ＿ＩＮＦ［Ｔ］．ＷＡＶＥ＿ＴＯＴＡＬ＿ＳＩＺＥというフォーマットでアクセスされる。これは、変数Ｔの値に対応する音色の波形データの合計サイズを示す。メンバ定数ＮＵＭ＿ＳＥＧは、図４（ａ）の「使用セグメント数」項目値に対応する音色内の波形データが使用するセグメント数を格納する。このメンバ定数へは例えば、ＴＯＮＥ＿ＩＮＦ［Ｔ］．ＮＵＭ＿ＳＥＧというフォーマットでアクセスされる。これは、変数Ｔの値に対応する音色の使用セグメント数を示す。

【００３５】

次に、図４（ｂ）に例示される音色波形情報テーブルは、図７に示される、上位構造体ＴＯＮＥ＿ＩＮＦ［１６］（［］内の数字は配列数を示す）と、その構造体を構成するメンバ構造体である中位構造体ＷＡＶＥ［６４］（［］内の数字は配列数を示す）と、その中位構造体を構成するメンバ定数である定数ＶＥＬ＿ＬＯ、ＶＥＬ＿ＨＩ、ＫＥＹ＿ＬＯ、ＫＥＹ＿ＨＩ、ＦＬＡＳＨ＿ＡＤＲＳ、ＳＥＧ＿ＡＤＲＳ、ＳＩＺＥ、ＳＥＧ＿ＧＲＯＵＰ、及び下位構造体ＴＧ＿ＩＮＦとによって構成される。上位構造体ＴＯＮＥ＿ＩＮＦ［１６］については前述した通りである。中位構造体ＷＡＶＥ［６４］は例えば、ＲＡＭ２０３（図２）上の変数ｗによって、ＷＡＶＥ［ｗ］としてアクセスされる。変数ｗが例えば０から６３までの値域をとることにより、図４（ｂ）の音色波形情報テーブルにおける「波形番号」項目値が０から６３までの各波形データのエントリが指定される。メンバ定数ＶＥＬ＿ＬＯおよびＶＥＬ＿ＨＩは、図４（ｂ）の「最低ベロシティ」項目値及び「最高ベロシティ」項目値にそれぞれ対応し、波形データが発音する範囲の本来の最低ベロシティ及び最高ベロシティの各値を格納する。これらのメンバ定数へは例えば、ＴＯＮＥ＿ＩＮＦ［Ｔ］．ＷＡＶＥ［ｗ］．ＶＥＬ＿ＬＯ又はＴＯＮＥ＿ＩＮＦ［Ｔ］．ＷＡＶＥ［ｗ］．ＶＥＬ＿ＨＩというフォーマットでアクセスされる。これは、変数Ｔの値に対応する音色の、変数ｗの値に対応する波形データの、最低ベロシティ又は最高ベロシティを示す。メンバ定数ＫＥＹ＿ＬＯおよびＫＥＹ＿ＨＩは、図４（ｂ）の「最低キー番号」項目値及び「最高キー番号」項目値にそれぞれ対応し、波形データが発音する範囲の本来の最低キー番号及び最高キー番号の各値を格納する。これらのメンバ定数へは例えば、ＴＯＮＥ＿ＩＮＦ［Ｔ］．ＷＡＶＥ［ｗ］．ＫＥＹ＿ＬＯ又はＴＯＮＥ＿ＩＮＦ［Ｔ］．ＷＡＶＥ［ｗ］．ＫＥＹ＿ＨＩというフォーマットでアクセスされる。これは、変数Ｔの値に対応する音色の、変数ｗの値に対応する波形データの、最低キー番号又は最高キー番号を示す。メンバ定数ＦＬＡＳＨ＿ＡＤＲＳは、図４（ｂ）には明示されていないが、波形データの先頭の大容量フラッシュメモリ２０４上でのアドレスを格納する。このメンバ定数へは例えば、ＴＯＮＥ＿ＩＮＦ［Ｔ］．ＷＡＶＥ［ｗ］．ＦＬＡＳＨ＿ＡＤＲＳというフォーマットでアクセスされる。これは、変数Ｔの値に対応する音色の、変数ｗの値に対応

する波形データの、先頭の大容量フラッシュメモリ２０４上でのアドレスを示す。メンバ定数ＳＥＧ＿ＡＤＲＳは、図４（ｂ）の「セグメント先頭からのアドレス」項目値に対応し、波形データが波形メモリ２０６上で読み込まれるべきセグメントの先頭からのアドレスオフセットを格納する。このメンバ定数へは例えば、ＴＯＮＥ＿ＩＮＦ［Ｔ］．ＷＡＶＥ［ｗ］．ＳＥＧ＿ＡＤＲＳというフォーマットでアクセスされる。これは、変数Ｔの値に対応する音色の、変数ｗの値に対応する波形データの、波形メモリ２０６上で読み込まれるべきセグメントの先頭からのアドレスオフセットを示す。メンバ定数ＳＩＺＥは、図４（ｂ）の「波形サイズ」項目値に対応し、波形データのサイズを格納する。このメンバ定数へは例えば、ＴＯＮＥ＿ＩＮＦ［Ｔ］．ＷＡＶＥ［ｗ］．ＳＩＺＥというフォーマットでアクセスされる。これは、変数Ｔの値に対応する音色の、変数ｗの値に対応する波形データのサイズを示す。ＴＯＮＥ＿ＩＮＦ［Ｔ］．ＷＡＶＥ［ｗ］．ＳＩＺＥ＝０のときは、波形データが存在しないことを示す。メンバ定数ＳＥＧ＿ＧＲＯＵＰは、図４（ｂ）の「セグメントグループ」項目値に対応し、波形データが波形メモリ２０６上の１つのセグメントに転送されるときに、そのセグメントに付与されるセグメントグループの値を格納する。このメンバ定数へは例えば、ＴＯＮＥ＿ＩＮＦ［Ｔ］．ＷＡＶＥ［ｗ］．ＳＥＧ＿ＧＲＯＵＰというフォーマットでアクセスされる。これは、変数Ｔの値に対応する音色の、変数ｗの値に対応する波形データの、セグメントグループ値を示す。ＴＯＮＥ＿ＩＮＦ［Ｔ］．ＷＡＶＥ［ｗ］．ＳＥＧ＿ＧＲＯＵＰの値が同じ波形データは、波形メモリ２０６上の同じセグメントに転送される。メンバ構造体ＴＧ＿ＩＮＦは、音源ＬＳＩ２０５（図２）が波形メモリ２０６上の１つのセグメントに転送された波形データに対して発音処理を実行する場合における各種パラメータを、その構造体のメンバ定数（図５では省略）として記憶する。それらのパラメータは例えば、波形データの読出しのスタートアドレス、繰返し位置を示すループアドレス、終了位置を示すエンドアドレス、音量、チューニング情報等である。

【００３６】

図８は、ＣＰＵ２０１（図２）が実行する制御処理において使用するＲＡＭ２０３に記憶８される主要な変数の一覧を示す図である。変数ＴＩＭＥＳＴＡＭＰは、ユーザが音色選択ボタン１０２（図１）を操作することにより発生する音色選択イベントごとに１ずつインクリメントする履歴番号を格納する。セグメントが割り当てられる際に、後述する構造体メンバ変数ＳＥＧ＿ＩＮＦ［ｓ］．ＴＩＭＥＳＴＡＭＰに格納される。変数ＣＵＲ＿ＴＯＮＥは、現在演奏のために選択されている音色番号を格納する。この値が「－１」の場合は該当無し、即ち現在演奏が行われていないことを示す。変数ＲＥＡＤＩＮＧ＿ＷＡＶＥは、現在バックグラウンド処理において波形転送中の波形データの波形番号を格納する。この値が「－１」の場合は、全て読込み済みであることを示す。

【００３７】

次に、波形メモリ２０６中のセグメントごとの情報が、構造体ＳＥＧ＿ＩＮＦ［６４］（［］内の数字は配列数を示す）と、その構造体を構成するメンバ変数ＴＯＮＥ、ＳＥＧ＿ＧＲＯＵＰ、及びＴＩＭＥＳＴＡＭＰとによって保持される。構造体ＳＥＧ＿ＩＮＦ［６４］は例えば、ＲＡＭ２０３（図２）上の変数ｓによって、ＳＥＧ＿ＩＮＦ［ｓ］としてアクセスされる。変数ｓが例えば０から６３までの値域をとることにより、波形メモリ２０６上の０から６３までの６４セグメントの各々が指定される。メンバ変数ＴＯＮＥは、当該セグメントが割り当てられている音色番号を格納する。この値が「－１」の場合は、そのセグメントが空状態（未使用）であることを示す。このメンバ変数へは例えば、ＳＥＧ＿ＩＮＦ［ｓ］．ＴＯＮＥというフォーマットでアクセスされる。これは、変数ｓの値に対応するセグメントに割り当てられている音色番号を示す。メンバ変数ＴＯＮＥには、このセグメントへの波形転送が実行されたときの音色番号を保持するＲＡＭ２０３上の変数Ｔ値がコピーされる。メンバ変数ＳＥＧ＿ＧＲＯＵＰは、当該セグメントに割り当てられている音色のセグメントグループ値を格納する。このメンバ変数へは例えば、ＳＥＧ＿ＩＮＦ［ｓ］．ＳＥＧ＿ＧＲＯＵＰというフォーマットでアクセスされる。これは、変数ｓの値に対応するセグメントに割り当てられている音色のセグメントグループの値を示

す。メンバ変数ＳＥＧ＿ＧＲＯＵＰには、このセグメントに転送される波形データに対応する、ＲＯＭ２０２に記憶されている図４（ｂ）の音色波形情報テーブル中の「セグメントグループ」項目値、すなわち図７のＴＯＮＥ＿ＩＮＦ［Ｔ］．ＷＡＶＥ［ｗ］．ＳＥＧ＿ＧＲＯＵＰの値がコピーされる。メンバ変数ＴＩＭＥＳＴＡＭＰは、当該セグメントが使用された履歴番号を格納する。このメンバ変数へは例えば、ＳＥＧ＿ＩＮＦ［ｓ］．ＴＩＭＥＳＴＡＭＰというフォーマットでアクセスされる。これは、変数ｓの値に対応するセグメントが使用された履歴番号を示す。ユーザが音色選択ボタン１０２（図１）を操作することにより発生する音色選択イベントごとに、そのタイミングで１ずつインクリメントされる変数ＴＩＭＥＳＴＡＭＰの値がコピーされる。同じタイミングで割り当てられたセグメントは、同じＴＩＭＥＳＴＡＭＰ値を有する。

【００３８】

次に、波形メモリ２０６中の各セグメント内に格納される１組以上の波形データに関する情報が、上位構造体ＳＥＧ＿ＩＮＦ［６４］（［］内の数字は配列数を示す）と、その構造体を構成するメンバ構造体である中位構造体ＷＡＶＥ＿ＩＮＦ［６４］（［］内の数字は配列数を示す）と、その中位構造体を構成するメンバ変数ＷＡＶＥ、ＷＡＶＥ＿ＮＵＭ、ＳＥＧ＿ＡＤＲＳ、及びＳＩＺＥとによって保持される。上位構造体ＳＥＧ＿ＩＮＦ［６４］については前述した通りである。中位構造体ＷＡＶＥ＿ＩＮＦ［６４］は例えば、ＲＡＭ２０３（図２）上の変数ｉによって、ＷＡＶＥ＿ＩＮＦ［ｉ］としてアクセスされる。変数ｉが例えば０から６３までの値域をとることにより、１つのセグメント内の０から６３までの各波形データが指定される。メンバ変数ＷＡＶＥは、当該セグメント中の当該波形データの有無を格納し、この値が０のときは波形データは無しであり、この値が１のときは波形データは有り（読込み済み）であることを示す。このメンバ変数へは例えば、ＳＥＧ＿ＩＮＦ［ｓ］．ＷＡＶＥ＿ＩＮＦ［ｉ］．ＷＡＶＥというフォーマットでアクセスされる。これは、変数ｓの値に対応するセグメントの、変数ｉの値に対応する波形データの、有無を示す。メンバ変数ＷＡＶＥの値は、当該セグメントへの当該波形データの転送が指示されたときに値０に、その後のバックグラウンド転送により当該セグメントへの当該波形データの転送が完了したときに値１に、それぞれセットされる。メンバ変数ＷＡＶＥ＿ＮＵＭは、当該セグメント中の当該波形データの音色中の波形番号を格納し、この値が「－１」であれば波形データの割当てが無いことを示す。このメンバ変数へは例えば、ＳＥＧ＿ＩＮＦ［ｓ］．ＷＡＶＥ＿ＩＮＦ［ｉ］．ＷＡＶＥ＿ＮＵＭというフォーマットでアクセスされる。これは、変数ｓの値に対応するセグメントの、変数ｉの値に対応する波形データの、音色中での波形番号を示す。メンバ変数ＳＥＧ＿ＡＤＲＳは、当該セグメント中での当該波形データのアドレスオフセットを格納する。このメンバ変数へは例えば、ＳＥＧ＿ＩＮＦ［ｓ］．ＷＡＶＥ＿ＩＮＦ［ｉ］．ＳＥＧ＿ＡＤＲＳというフォーマットでアクセスされる。メンバ変数ＳＥＧ＿ＡＤＲＳには、このセグメントに転送される波形データに対応する、ＲＯＭ２０２に記憶されている図４（ｂ）の音色波形情報テーブル中の「セグメント先頭からのアドレス」項目値、すなわち図７のＴＯＮＥ＿ＩＮＦ［Ｔ］．ＷＡＶＥ［ｗ］．ＳＥＧ＿ＡＤＲＳの値がコピーされる。メンバ変数ＳＩＺＥは、当該セグメント中での当該波形データのサイズを格納する。このメンバ変数へは例えば、ＳＥＧ＿ＩＮＦ［ｓ］．ＷＡＶＥ＿ＩＮＦ［ｉ］．ＳＩＺＥというフォーマットでアクセスされる。メンバ変数ＳＩＺＥには、このセグメントに転送される波形データに対応する、ＲＯＭ２０２に記憶されている図４（ｂ）の音色波形情報テーブル中の「波形サイズ」項目値、すなわち図７のＴＯＮＥ＿ＩＮＦ［Ｔ］．ＷＡＶＥ［ｗ］．ＳＩＺＥの値がコピーされる。

【００３９】

以下に、上述の動作を実現するためにＣＰＵ２０１が実行する制御処理の詳細例について説明する。

【００４０】

図９は、ＣＰＵ２０１が実行する制御処理の全体処理の例を示すメインルーチンのフローチャートである。この処理例は、ＣＰＵ２０１がＲＯＭ２０２に記憶された制御プログ

ラムを実行する処理例である。

【００４１】

ＣＰＵ２０１は、まずＲＡＭ２０３の内容を初期化した後（ステップＳ９０１）、ステップＳ９０２からＳ９１１の一連の処理で示される定常ループ処理に入る。

【００４２】

定常ループ処理では、ＣＰＵ２０１はまず、ユーザインタフェース処理（図中「ユーザＩ／Ｆ」と表示）を実行する（ステップＳ９０２）。ここでは、ＣＰＵ２０１は、図２のキー・スキャナ２０７を介して図１の音色選択ボタン１０２の状態を取得する。

【００４３】

次に、ＣＰＵ２０１は、ステップＳ９０２の処理の結果、ユーザが音色選択ボタン１０２を操作することにより音色選択イベントが発生したか否かを判定する（ステップＳ９０３）。

【００４４】

そして、ＣＰＵ２０１は、音色選択イベントが発生した場合（ステップＳ９０３の判定がＹｅｓの場合）は、音色切替え処理を実行する（ステップＳ９０４）。ステップＳ９０３の判定がＮｏならば、ＣＰＵ２０１は、ステップＳ９０４の処理はスキップする。

【００４５】

次に、ＣＰＵ２０１は、波形読込み処理を実行する（ステップＳ９０５）。ここでは、ＣＰＵ２０１は、図２の大容量フラッシュメモリ２０４から波形メモリ２０６へのバックグラウンド処理による波形転送を実行する。

【００４６】

次に、ＣＰＵ２０１は、鍵盤読込み処理を実行する（ステップＳ９０６）。ここでは、ＣＰＵ２０１は、図２のキー・スキャナ２０７を介して図１の鍵盤１０１の押鍵状態を取得する。

【００４７】

次に、ＣＰＵ２０１は、ステップＳ９０６の処理の結果、ユーザが鍵盤１０１上の何れかの鍵を押鍵することにより押鍵イベントが発生したか否かを判定する（ステップＳ９０７）。

【００４８】

そして、ＣＰＵ２０１は、押鍵イベントが発生した場合（ステップＳ９０７の判定がＹｅｓの場合）は、押鍵処理を実行する（ステップＳ９０８）。ステップＳ９０７の判定がＮｏならば、ＣＰＵ２０１は、ステップＳ９０８の処理はスキップする。

【００４９】

続いて、ＣＰＵ２０１は、ステップＳ９０６の処理の結果、ユーザが鍵盤１０１上の何れかの押鍵中の鍵を離鍵することにより離鍵イベントが発生したか否かを判定する（ステップＳ９０９）。

【００５０】

そして、ＣＰＵ２０１は、離鍵イベントが発生した場合（ステップＳ９０９の判定がＹｅｓの場合）は、離鍵処理を実行する（ステップＳ９１０）。ステップＳ９０９の判定がＮｏならば、ＣＰＵ２０１は、ステップＳ９１０の処理はスキップする。

【００５１】

その後、ＣＰＵ２０１は、音源定常サービス処理を実行する（ステップＳ９１１）。ここでは、例えば図１の機能選択ボタン１０３が押された場合に対応する処理や、図１のベンダ／モジュレーション・ホイール１０４が操作された場合に対応する処理等の、電子鍵盤楽器１００に対する一般的な処理が実行される。

【００５２】

その後、ＣＰＵ２０１は、ステップＳ９０２の定常ループ処理の先頭に戻る。

【００５３】

図１０は、図９のステップＳ９０１の初期化処理の詳細例を示すフローチャートである。

【００５４】
まず、ＣＰＵ２０１は、ＲＡＭ２０３上の図８に示される変数の値を、全て「０」にする（ステップＳ１００１）。
【００５５】
次に、ＣＰＵ２０１は、現在演奏のために選択されている音色番号を格納するＲＡＭ２０３上の変数ＣＵＲ＿ＴＯＮＥ（図８参照）に、該当無しを示す値「－１」を格納する（ステップＳ１００２）。
【００５６】
次に、ＣＰＵ２０１は、現在バックグラウンド処理において波形転送中の波形データの波形番号を格納するＲＡＭ２０３上の変数ＲＥＡＤＩＮＧ＿ＷＡＶＥに、全て読込み済みであることを示す値「－１」を格納する（ステップＳ１００３）。
【００５７】
次に、ＣＰＵ２０１は、セグメントをカウントするためのＲＡＭ２０３上の変数ｓについて、ステップＳ１００４において値「０」に初期設定した後、ステップＳ１０１１で＋１ずつインクリメントしながら、ステップＳ１０１０で値「６３」に達したと判定するまで、ステップＳ１００５からステップＳ１００９までの処理を繰り返し実行する。
【００５８】
この繰返し処理において、ＣＰＵ２０１はまず、変数ｓの値に対応するセグメントに割り当てられている音色番号を示す変数ＳＥＧ＿ＩＮＦ［ｓ］．ＴＯＮＥに、そのセグメントが空状態（未使用）であることを示す値「－１」を格納する（ステップＳ１００５）。
【００５９】
次に、ＣＰＵ２０１は、セグメント内の波形データの順番をカウントするためのＲＡＭ２０３上の変数ｗについて、ステップＳ１００６において値「０」に初期設定した後、ステップＳ１００９で＋１ずつインクリメントしながら、ステップＳ１００８で値「６３」に達したと判定するまで、ステップＳ１００７の処理を繰り返し実行する。即ち、ＣＰＵ２０１は、このステップＳ１００７において、変数ｓの値に対応するセグメント内のｉ番目（変数ｉが示す値）の波形データの音色中の波形番号を格納する変数ＳＥＧ＿ＩＮＦ［ｓ］．ＷＡＶＥ＿ＩＮＦ［ｉ］．ＷＡＶＥ＿ＮＵＭに、波形データの割当てが無いことを示す値「－１」を格納する。
【００６０】
以上の繰返し処理の後、ステップＳ１０１０の判定がＹｅｓになると、ＣＰＵ２０１は、ＣＰＵ２０１は、図１０のフローチャートで示される図９のステップＳ９０１の初期化処理を終了する。
【００６１】
図１１は、図９の音色選択イベント発生時（ステップＳ９０３の判定がＹｅｓのとき）のステップＳ９０４の音色切替え処理の、詳細例を示すフローチャートである。この処理の実行時には、ＲＡＭ２０３上の音色番号変数Ｔの値が引き渡される。音色番号変数Ｔには、図１０のステップＳ１００１の実行時には前述したように初期値「０」が格納され、図９のステップＳ９０４の実行時には、ユーザによる図１の音色選択ボタン１０２の操作に基づいて図９のステップＳ９０２のユーザインタフェース処理で取り込まれたユーザが選択した音色の音色番号が格納される。
【００６２】
図１１の音色切替え処理のフローチャートにおいて、ＣＰＵ２０１はまず、現在演奏のために選択されている音色番号を格納する変数ＣＵＲ＿ＴＯＮＥの値が、ＲＡＭ２０３の変数Ｔに格納されている、ユーザにより新たに指定された音色番号の値に等しいか否かを判定する（ステップＳ１１０１）。
【００６３】
ステップＳ１１０１の判定がＹｅｓならば、既に選択音色として選択済み又は現在大容量フラッシュメモリ２０４から波形メモリ２０６に該当する音色の波形データを転送中であるため、図１１のフローチャートで示される図９のステップＳ９０４の音色切替え処理

をそのまま終了する。

【００６４】

ステップＳ１１０１の判定がＮｏならば、ＣＰＵ２０１は、履歴番号を格納する変数ＴＩＭＥＳＴＡＭＰの値を＋１インクリメントする（ステップＳ１１０２）。

【００６５】

次に、ＣＰＵ２０１は、現在演奏のために選択されている音色番号を格納する変数ＣＵＲ＿ＴＯＮＥに、ＲＡＭ２０３の変数Ｔに格納されている、ユーザにより新たに指定された音色番号の値を格納する（ステップＳ１１０３）。

【００６６】

次に、ＣＰＵ２０１は、ＲＡＭ波形データ調査ルーチンの処理を実行する（ステップＳ１１０４）。ＣＰＵ２０１は、ＲＡＭ２０３上の変数Ｔに格納されている新たに指定された音色番号を引数としてこのルーチンに引き渡す。ＣＰＵ２０１は、このルーチンの処理の結果、ＲＡＭ２０３上の変数Ｒに、指定された音色番号に対応する音色の波形データが波形メモリ２０６上に有るか否かを示す戻り値（値０ならば無し、値１ならば有り）を得る（ステップＳ１１０４）。この処理の詳細については後述する。

【００６７】

ＣＰＵ２０１は、変数Ｒの値が１であるか否かを判定する（ステップＳ１１０５）。

【００６８】

ステップＳ１１０５の判定がＹｅｓならば、指定された音色番号に対応する音色の波形データを新たに波形メモリ２０６に転送する必要はないため、ＣＰＵ２０１は、図１１のフローチャートで示される図９のステップＳ９０４の音色切替え処理をそのまま終了する。

【００６９】

ステップＳ１１０５の判定がＮｏならば、ＣＰＵ２０１は、波形領域確保ルーチンの処理を実行する（ステップＳ１１０６）。ＣＰＵ２０１は、ＲＡＭ２０３上の変数Ｔに格納されている新たに指定された音色番号を引数としてこのルーチンに引き渡す。ＣＰＵ２０１は、このルーチンの処理の結果、指定された音色番号に対応する音色の波形データを大容量フラッシュメモリ２０４から転送するのに必要な数の空きセグメントを波形メモリ２０６上に確保する。この処理の詳細については後述する。

【００７０】

その後、ＣＰＵ２０１は、波形読込み指示ルーチンの処理を実行する（ステップＳ１１０７）。ＣＰＵ２０１は、ＲＡＭ２０３上の変数Ｔに格納されている新たに指定された音色番号を引数としてこのルーチンに引き渡す。ＣＰＵ２０１は、このルーチンの処理において、指定された音色番号に対応する音色の波形データを、図９のステップＳ９０５の波形読込み処理によって大容量フラッシュメモリ２０４から波形メモリ２０６に転送するための準備処理を実行する。

【００７１】

図１２（ａ）は、図１１のステップＳ１１０４のＲＡＭ波形データ調査ルーチンの詳細例を示すフローチャートである。

【００７２】

まず、ＣＰＵ２０１は、調査するセグメントをカウントするためのＲＡＭ２０３上の変数ｓを０にリセットする（ステップＳ１２０１）。

【００７３】

次に、ＣＰＵ２０１は、変数ｓの値に対応するセグメントに割り当てられている音色番号が格納されているＲＡＭ２０３上の変数ＳＥＧ＿ＩＮＦ［ｓ］．ＴＯＮＥ（図８参照）の値が、引数として引き渡された新たに指定された音色番号の値が格納されているＲＡＭ２０３上の変数Ｔの値に等しいか否かを判定する（ステップＳ１２０２）。

【００７４】

ステップＳ１２０２の判定がＮｏならば、ＣＰＵ２０１は、ステップＳ１２０３で変数ｓの値を＋１ずつインクリメントしながら、ステップＳ１２０４で変数ｓの値が６４に達

したと判定するまで、変数ｓにより指定される各セグメントについて、ステップＳ１２０２の判定処理を繰り返す（ステップＳ１２０３→Ｓ１２０４の判定がＮｏ→Ｓ１２０２→Ｓ１２０３の処理の繰返し）。

【００７５】

上記ステップＳ１２０２からＳ１２０４の処理の繰返しにおいて、ステップＳ１２０２の判定がＹｅｓになると、ＣＰＵ２０１はまず、変数ｓの値に対応するセグメントが使用された履歴番号を示す変数ＳＥＧ＿ＩＮＦ［ｓ］．ＴＩＭＥＳＴＡＭＰに、今回の音色選択イベントに基づいてインクリメントされたＲＡＭ２０３上の変数ＴＩＭＥＳＴＡＭＰの値（図１１のステップＳ１１０２参照）を格納して、当該セグメントの履歴番号を更新する（ステップＳ１２０５）。

【００７６】

続いて、変数ｓに対応するセグメントに指定された音色番号に対応する音色の波形データが格納されていることが判明したため、ＣＰＵ２０１は、ＲＡＭ２０３上の変数Ｒに、指定された音色番号に対応する音色の波形データが波形メモリ２０６上に有ることを示す戻り値１を格納する（ステップＳ１２０６）。その後、ＣＰＵ２０１は、図１２のフローチャートで示される図１１のステップＳ１１０４のＲＡＭ波形データ調査ルーチンの処理を終了する。

【００７７】

ステップＳ１２０２からＳ１２０４の処理の繰返しにおいて、変数ｓによるセグメントの指定が末尾まで達してステップＳ１２０４の判定がＹｅｓになると、指定された音色番号に対応する音色の波形データはどのセグメントにも格納されていないことが判明したため、ＣＰＵ２０１は、ＲＡＭ２０３上の変数Ｒに、指定された音色番号に対応する音色の波形データが波形メモリ２０６上に無いことを示す戻り値０を格納する（ステップＳ１２０７）。その後、ＣＰＵ２０１は、図１２のフローチャートで示される図１１のステップＳ１１０４のＲＡＭ波形データ調査ルーチンの処理を終了する。

【００７８】

図１２（ｂ）は、図１１のステップＳ１１０６の波形領域確保ルーチンの詳細例を示すフローチャートである。このルーチンは、前述したように、図１１のステップＳ１１０４のＲＡＭ波形データ調査ルーチンにより指定された音色番号に対応する音色の波形データが波形メモリ２０６上のどのセグメントにも無いことが判明した場合に実行される。

【００７９】

ＣＰＵ２０１はまず、引数として引き渡された、指定された音色番号が格納されているＲＡＭ２０３上の変数Ｔの値に対応する音色の使用セグメント数を保持するＲＯＭ２０２上の定数値ＴＯＮＥ＿ＩＮＦ［Ｔ］．ＮＵＭ＿ＳＥＧ（図７参照）を、ＲＡＭ２０３上の変数ｕに格納する（ステップＳ１２１０）。

【００８０】

次にＣＰＵ２０１は、空きセグメント数調査ルーチンを実行する。このルーチンでは波形メモリ２０６上で使用されていない空きセグメントの数が取得される。ＣＰＵ２０１は、このルーチンの処理の結果、ＲＡＭ２０３上の変数Ｒに、空きセグメント数の値を得る（ステップＳ１２１１）。この処理の詳細については後述する。

【００８１】

ＣＰＵ２０１は、変数Ｒに得られた空きセグメント数が、ステップＳ１２１０で変数ｕに格納された指定された音色番号に対応する音色における使用セグメント数以上となっているか否かを判定する（ステップＳ１２１２）。

【００８２】

ステップＳ１２１２の判定がＮｏならば、ＣＰＵ２０１は、波形領域開放ルーチンを実行する（ステップＳ１２１３）。このルーチンでは、波形メモリ２０６上で最も古い履歴番号を有する音色のセグメントが開放される。この処理の詳細については後述する。

【００８３】

その後、ＣＰＵ２０１は、ステップＳ１２１１に戻って、空きセグメント数調査ルーチ

ンを再度実行し、変数Ｒに得られた空きセグメント数が、ステップＳ１２１０で変数ｕに格納された使用セグメント数以上となったか否かを判定する（ステップＳ１２１２）。

【００８４】

以上のステップＳ１２１１からステップＳ１２１３までの処理の繰返しの結果、変数Ｒに得られた空きセグメント数が指定された音色番号に対応する音色における使用セグメント数以上となることによりステップＳ１２１２の判定がＹｅｓになると、必要な空きセグメントが確保できたため、ＣＰＵ２０１は、図１２のフローチャートで示される図１１のステップＳ１１０６の波形領域確保ルーチンを終了する。

【００８５】

図１３は、図１２のステップＳ１２１１の空きセグメント数調査ルーチンの詳細例を示すフローチャートである。前述したようにこのルーチンでは、波形メモリ２０６上での空きセグメントの数が取得される。

【００８６】

ＣＰＵ２０１はまず、セグメントをカウントするためのＲＡＭ２０３上の変数ｓを０にリセットすると共に、戻り値である空きセグメント数を格納するＲＡＭ２０３上の変数Ｒの値を０にリセットする（ステップＳ１３０１）。

【００８７】

次に、ＣＰＵ２０１は、ステップＳ１３０４で変数ｓの値を＋１ずつインクリメントしながら、ステップＳ１３０５で変数ｓの値が６４に達したと判定するまで、変数ｓの値によって指定されるセグメントごとに、ステップＳ１３０２とＳ１３０３の処理を繰り返し実行する。

【００８８】

上述の繰返し処理において、ＣＰＵ２０１はまず、変数ｓの値に対応するセグメントに割り当てられている音色番号が格納されている変数ＳＥＧ＿ＩＮＦ［ｓ］．ＴＯＮＥの値が、そのセグメントが空状態（未使用）であることを示す値「－１」に等しいか否かを判定する（ステップＳ１３０２）。

【００８９】

そのセグメントが空状態であることによりステップＳ１３０２の判定がＹｅｓならば、ＣＰＵ２０１は、戻り値である空きセグメント数を格納するＲＡＭ２０３上の変数Ｒの値を＋１インクリメントする（ステップＳ１３０３）。

【００９０】

ステップＳ１３０２の判定がＮｏならば、ＣＰＵ２０１は、ステップＳ１３０３の処理をスキップする。

【００９１】

以上の繰返し処理の後、ステップＳ１３０５の判定がＹｅｓになると、ＣＰＵ２０１は、図１３のフローチャートで示される図１１のステップＳ１２１１の空きセグメント数調査ルーチンの処理を終了する。この結果、ＲＡＭ２０３上の変数Ｒに、波形メモリ２０６上での空きセグメントの数が得られる。

【００９２】

図１４は、図１２のステップＳ１２１３の波形領域開放ルーチンの詳細例を示すフローチャートである。前述したようにこのルーチンでは、波形メモリ２０６上で最も古い履歴番号を有する音色のセグメントが開放される。

【００９３】

ＣＰＵ２０１はまず、セグメントをカウントするためのＲＡＭ２０３上の変数ｓを０にリセットすると共に、比較用の履歴番号を格納するＲＡＭ２０３上の変数ｔに十分大きな値、例えばＦＦＦＦＦＦＦＨ（末尾の「Ｈ」はその左側が１６進数であることを示す）をセットする（ステップＳ１４０１）。

【００９４】

次に、ＣＰＵ２０１は、ステップＳ１４０５で変数ｓの値を＋１ずつインクリメントしながら、ステップＳ１４０６で変数ｓの値が６４に達したと判定するまで、変数ｓの値に

よって指定されるセグメントごとに、ステップＳ１４０２からＳ１４０４の一連の処理を繰り返し実行する。

【００９５】

上述の繰返し処理において、ＣＰＵ２０１はまず、変数ｓの値に対応するセグメントに割り当てられている音色番号が格納されている変数ＳＥＧ＿ＩＮＦ［ｓ］．ＴＯＮＥの値が、そのセグメントが空状態（未使用）であることを示す値「－１」に等しいか否かを判定する（ステップＳ１４０２）。

【００９６】

ステップＳ１４０２の判定がＹｅｓならば、ＣＰＵ２０１は、ステップＳ１４０５に移行して、次のセグメントの繰返し処理に移る。

【００９７】

ステップＳ１４０２の判定がＮｏならば、ＣＰＵ２０１は、変数ｓの値に対応するセグメントが使用された履歴番号を示す変数ＳＥＧ＿ＩＮＦ［ｓ］．ＴＩＭＥＳＴＡＭＰの値が、比較用の履歴番号を格納するＲＡＭ２０３上の変数ｔの値よりも小さいか否かを判定する（ステップＳ１４０３）。

【００９８】

ステップＳ１４０３の判定がＹｅｓならば、ＣＰＵ２０１は、比較用の履歴番号を格納するＲＡＭ２０３上の変数ｔに、変数ｓの値に対応するセグメントが使用された履歴番号を示す変数ＳＥＧ＿ＩＮＦ［ｓ］．ＴＩＭＥＳＴＡＭＰの値を格納する（ステップＳ１４０４）。この結果、現時点で、変数ｓの値に対応するセグメントが音色指定により使用された履歴番号が、最も古い履歴番号として変数ｔに保持される。

【００９９】

ステップＳ１４０３の判定がＮｏならば、ＣＰＵ２０１は、ステップＳ１４０４の処理はスキップして、現在の変数ｔの値を維持する。

【０１００】

以上の繰返し処理の結果、全てのセグメントの履歴番号との比較が完了してステップＳ１４０６の判定がＹｅｓになると、変数ｔに最も小さい即ち最も古い履歴番号が得られる。

【０１０１】

次に、ＣＰＵ２０１は、ステップＳ１４０７でセグメントをカウントするための変数ｓの値を０にリセットした後、ステップＳ１４１０で変数ｓの値を＋１ずつインクリメントしながら、ステップＳ１４１１で変数ｓの値が６４に達したと判定するまで、変数ｓの値によって指定されるセグメントごとに、ステップＳ１４０８とＳ１４０９の繰返し処理として、変数ｔに得られた最も古い履歴番号を有するセグメントを開放する処理を実行する。

【０１０２】

上述の繰返し処理において、ＣＰＵ２０１はまず、変数ｓの値に対応するセグメントが使用された履歴番号を示す変数ＳＥＧ＿ＩＮＦ［ｓ］．ＴＩＭＥＳＴＡＭＰの値が変数ｔに格納されている最も古い履歴番号の値に等しいか否かを判定する（ステップＳ１４０８）。

【０１０３】

ステップＳ１４０８の判定がＹｅｓならば、ＣＰＵ２０１は、変数ｓの値に対応するセグメントに割り当てられている音色番号が格納されている変数ＳＥＧ＿ＩＮＦ［ｓ］．ＴＯＮＥに、そのセグメントが空状態（未使用）であることを示す値「－１」を格納する（ステップＳ１４０９）。これにより、そのセグメントが開放される。

【０１０４】

ステップＳ１４０８の判定がＮｏならば、ＣＰＵ２０１は、ステップＳ１４０９の判定をスキップして、変数ｓの値に対応するセグメントは開放しない。

【０１０５】

以上の繰返し処理の結果、ステップＳ１４１１の判定がＹｅｓになると、ＣＰＵ２０１

は、図１４のフローチャートで示される図１２ステップＳ１２１３の波形領域開放ルーチンを終了する。この結果、最も古い履歴番号を有する１つ以上のセグメントが開放される。

【０１０６】

図１５は、図１１のステップＳ１１０７の波形読込み指示ルーチンの詳細例を示すフローチャートである。前述したようにこのルーチンでは、指定された音色番号に対応する音色の波形データを、図９のステップＳ９０５の波形読込み処理によって大容量フラッシュメモリ２０４から波形メモリ２０６に転送するための準備処理が実行される。この準備処理は、大容量フラッシュメモリ２０４から波形データが転送される波形メモリ２０６上の空きセグメントに対応するＲＡＭ２０３上のＳＥＧ＿ＩＮＦ構造体を構成する各メンバ変数（図８参照）に必要な情報をセットし、また転送を制御するＲＡＭ２０３上の変数ＲＥＡＤＩＮＧ＿ＷＡＶＥが示す波形番号を初期値０にセットする処理である。

【０１０７】

ＣＰＵ２０１はまず、転送される波形データのセグメントグループをカウントするためのＲＡＭ２０３上の変数ｇの値と、転送される波形データの新たに指定される音色中での波形番号をカウントするためのＲＡＭ２０３の変数ｗの値と、波形メモリ２０６上でのセグメントをカウントするためのＲＡＭ２０３上の変数ｓの値を、それぞれ０にリセットする（ステップＳ１５０１）。

【０１０８】

次に、ＣＰＵ２０１は、ステップＳ１５１１で変数ｓの値を＋１ずつインクリメントしながら、ステップＳ１５１２で変数ｓの値が６４に達したと判定するまで、変数ｓの値によって指定されるセグメントごとに、ステップＳ１５０２からＳ１５１０の一連の処理を繰り返し実行する。

【０１０９】

上述の繰返し処理において、ＣＰＵ２０１はまず、変数ｓの値に対応するセグメントに割り当てられている音色番号が格納されている変数ＳＥＧ＿ＩＮＦ［ｓ］．ＴＯＮＥの値が、そのセグメントが空状態（未使用）であることを示す値「－１」に等しいか否かを判定する（ステップＳ１５０２）。

【０１１０】

ステップＳ１５０２の判定がＹｅｓならば、そのセグメントは空きセグメントではないため、ＣＰＵ２０１は、ステップＳ１５１１に移行して、次のセグメントの繰返し処理に移る。

【０１１１】

ステップＳ１５０２の判定がＮｏならば、ＣＰＵ２０１は、変数ｓの値に対応する空きセグメントに対応して音色番号を保持するＲＡＭ２０３上の変数ＳＥＧ＿ＩＮＦ［ｓ］．ＴＯＮＥに、引数として引き渡されたＲＡＭ２０３上の変数Ｔに格納されている新たに指定された音色番号の値を格納する。また、ＣＰＵ２０１は、変数ｓの値に対応する空きセグメントに対応してセグメントが使用された履歴番号を保持する変数ＳＥＧ＿ＩＮＦ［ｓ］．ＴＩＭＥＳＴＡＭＰに、今回の音色選択イベントに基づいてインクリメントされたＲＡＭ２０３上の変数ＴＩＭＥＳＴＡＭＰの値（図１１のステップＳ１１０２参照）を格納して、当該セグメントの履歴番号を更新する。更に、ＣＰＵ２０１は、変数ｓの値に対応する空きセグメントに対応する音色のセグメントグループ値を格納する変数ＳＥＧ＿ＩＮＦ［ｓ］．ＳＥＧ＿ＧＲＯＵＰに、変数ｇに格納されている現在転送を行っている波形データのセグメントグループ値を格納する（以上、ステップＳ１５０３）。

【０１１２】

その後、ＣＰＵ２０１は、変数ｓの値に対応する現在のセグメントに転送される波形データをカウントするＲＡＭ２０３上の変数ｉの値を０にリセットする（ステップＳ１５０４）。

【０１１３】

次に、ＣＰＵ２０１は、ステップＳ１５０６で、変数ｓの値に対応する現在のセグメン

トに転送される波形データをカウントする変数ｉの値と、転送される波形データの新たに指定される音色中での波形番号をカウントするための変数ｗの値を、それぞれ＋１ずつインクリメントしながら、ステップＳ１５０７及びＳ１５０８によりセグメントグループが変化したと判定されるまで、ステップＳ１５０５の処理を繰返し実行する。即ち、ステップＳ１５００５において、ＣＰＵ２０１はまず、変数ｓの値に対応する現在のセグメント中のｉ番目（＝変数ｉの値）の波形データに対応する、波形データが読込み済みであるか否かを示す変数ＳＥＧ＿ＩＮＦ［ｓ］．ＷＡＶＥ＿ＩＮＦ［ｉ］．ＷＡＶＥに、まだ波形が読み込まれていないことを示す値０をセットする。次に、ＣＰＵ２０１は、変数ｓの値に対応する現在のセグメント中のｉ番目の波形データに対応する、指定された音色中の波形番号を示す変数ＳＥＧ＿ＩＮＦ［ｓ］．ＷＡＶＥ＿ＩＮＦ［ｉ］．ＷＡＶＥ＿ＮＵＭに、ＲＡＭ２０３上の変数ｗが示す現在転送中の波形データの指定された音色中の波形番号をセットする。続いて、ＣＰＵ２０１は、変数ｓの値に対応する現在のセグメント中のｉ番目の波形データに対応する、当該セグメント中での当該波形データのアドレスオフセットを格納する変数ＳＥＧ＿ＩＮＦ［ｓ］．ＷＡＶＥ＿ＩＮＦ［ｉ］．ＳＥＧ＿ＡＤＲＳに、このセグメントに転送される音色番号Ｔ（＝変数Ｔの値）の音色中のｗ番目（＝変数ｗの値）の波形データに対応する、ＲＯＭ２０２中の定数ＴＯＮＥ＿ＩＮＦ［Ｔ］．ＷＡＶＥ［ｗ］．ＳＥＧ＿ＡＤＲＳ（図７参照）のアドレスオフセット値をセットする。更に、ＣＰＵ２０１は、変数ｓの値に対応する現在のセグメント中のｉ番目の波形データのサイズを格納する変数ＳＥＧ＿ＩＮＦ［ｓ］．ＷＡＶＥ＿ＩＮＦ［ｉ］．ＳＩＺＥに、このセグメントに転送される音色番号Ｔ（＝変数Ｔの値）の音色中のｗ番目（＝変数ｗの値）の波形データに対応する、ＲＯＭ２０２中の定数ＴＯＮＥ＿ＩＮＦ［Ｔ］．ＷＡＶＥ［ｗ］．ＳＩＺＥ（図７参照）のサイズ値をセットする（以上、ステップＳ１５０５）。

【０１１４】

上記ステップＳ１５０５のセット処理の後、ＣＰＵ２０１は、変数ｓの値に対応する現在のセグメントに転送される波形データをカウントする変数ｉの値と、転送される波形データの新たに指定される音色中での波形番号をカウントするための変数ｗの値を、それぞれ＋１ずつインクリメントする（ステップＳ１５０６）。

【０１１５】

その後、ＣＰＵ２０１は、このセグメントに転送される音色番号Ｔ（＝変数Ｔの値）の音色中の新たに指定されたｗ番目（＝変数ｗの値）の波形データに対応する、ＲＯＭ２０２中の定数ＴＯＮＥ＿ＩＮＦ［Ｔ］．ＷＡＶＥ［ｗ］．ＳＥＧ＿ＧＲＯＵＰのセグメントグループ値をＲＡＭ２０３上の変数ａにセットする。また、ＣＰＵ２０１は、このセグメントに転送される音色番号Ｔ（＝変数Ｔの値）の音色中の１つ前のｗ－１番目（＝変数ｗの値－１）の波形データに対応する、ＲＯＭ２０２中の定数ＴＯＮＥ＿ＩＮＦ［Ｔ］．ＷＡＶＥ［ｗ－１］．ＳＥＧ＿ＧＲＯＵＰのセグメントグループ値をＲＡＭ２０３上の変数ｂにセットする（以上、ステップＳ１５０７）。

【０１１６】

そして、ＣＰＵ２０１は、変数ａが示す新たに指定された音色中のｗ番目の波形データのセグメントグループの値が、変数ｂが示す１つ前のｗ－１番目の波形データのセグメントグループの値と等しいか否かを判定する（ステップＳ１５０８）。

【０１１７】

ステップＳ１５０８の判定がＹｅｓならば、同じセグメントに更に波形データを転送できるため、ステップＳ１５０５の処理に戻って新たな波形データに対応するセット処理を続行する。

【０１１８】

ステップＳ１５０８の判定がＮｏならば、ＣＰＵ２０１は、変数ｇに格納されているセグメントグループの値を＋１インクリメントする（ステップＳ１５０９）。

【０１１９】

その後、ＣＰＵ２０１は、変数ｇの値が、音色番号Ｔ（＝変数Ｔの値）に対応する定数ＴＯＮＥ＿ＩＮＦ［Ｔ］．ＮＵＭ＿ＳＥＧの使用セグメント数の値に達したか否かを判定

する（ステップＳ１５１０）。

【０１２０】

ステップＳ１５１０の判定がＮｏならば、ＣＰＵ２０１は、ステップＳ１５１１の処理に移行し、次の空きセグメントに対する転送処理を続行する。

【０１２１】

ステップＳ１５１０の判定がＹｅｓになると、指定された音色番号Ｔ（＝変数Ｔの値）に対応する全ての使用セグメント数分の波形データの転送処理が完了したため、ＣＰＵ２０１は、現在転送中の波形番号を示すＲＡＭ２０３上の変数ＲＥＡＤＩＮＧ＿ＷＡＶＥに、初期値０にセットする（ステップＳ１５１３）。その後、ＣＰＵ２０１は、図１５のフローチャートで示される図１１のステップＳ１１０７の波形読込み指示ルーチンの処理を終了する。

【０１２２】

図１６は、図９のステップＳ９０５の波形読込み処理の詳細例を示すフローチャートである。前述したようにここでは、ＣＰＵ２０１は、図２の大容量フラッシュメモリ２０４から波形メモリ２０６へのバックグラウンド処理による波形転送が実行される。

【０１２３】

ＣＰＵ２０１はまず、ＲＡＭ２０３上の変数ＲＥＡＤＩＮＧ＿ＷＡＶＥの値が、全て読込み済みであることを示す値「－１」であるか否かを判定する（ステップＳ１６０１）。

【０１２４】

ステップＳ１６０１の判定がＹｅｓならば、波形読込み処理を実行する必要はないため、ＣＰＵ２０１は、図１６のフローチャートで示される図９のステップＳ９０５の波形読込み処理をそのまま終了する。

【０１２５】

ステップＳ１６０１の判定がＮｏならば、ＣＰＵ２０１は、変数ＲＥＡＤＩＮＧ＿ＷＡＶＥが示す指定された音色中で次に転送されるべき波形番号の値を、ＲＡＭ２０３上の現在転送中の音色中の波形番号を示す変数ｗに格納する（ステップＳ１６０２）。

【０１２６】

次に、ＣＰＵ２０１は、セグメントをカウントするＲＡＭ２０３上の変数ｓの値を、ステップＳ１６０３で０にリセットした後、ステップＳ１６０５で＋１ずつインクリメントしながら、ステップＳ１６０４の判定による探索処理を実行する。ステップＳ１６０４で、ＣＰＵ２０１は、変数ｓに対応するセグメントが割り当てられている音色番号を示す変数ＳＥＧ＿ＩＮＦ［ｓ］．ＴＯＮＥの値が、ＲＡＭ２０３上の変数ＣＵＲ＿ＴＯＮＥに格納されている現在演奏のために選択されている音色番号（図１１のステップＳ１１０３参照）に等しく、かつ、変数ＳＥＧ＿ＩＮＦ［ｓ］．ＳＥＧ＿ＧＲＯＵＰにセットされている変数ｓに対応するセグメントが割り当てられているセグメントグループの値が、ＲＯＭ２０２上の定数ＴＯＮＥ＿ＩＮＦ［ＣＵＲ＿ＴＯＮＥ］．ＷＡＶＥ［ｗ］．ＳＥＧ＿ＧＲＯＵＰに格納されている変数ＣＵＲ＿ＴＯＮＥに対応する音色の変数ｗが示す現在転送中の音色中の波形番号に対応する波形データのセグメントグループの値に等しいか否かを判定する。

【０１２７】

前述した波形読込み指示ルーチン内のステップＳ１５０３の処理が予め実行されていることにより、ステップＳ１６０４とステップＳ１６０５の繰返し処理においてステップＳ１６０４の判定は必ずＹｅｓになる。ＣＰＵ２０１は、ステップＳ１６０４の判定がＹｅｓになると、変数ｓの値が示すセグメント内の波形データの順番をカウントするＲＡＭ２０３上の変数ｉの値を、ステップＳ１６０６で０にリセットした後、ステップＳ１６０８で＋１ずつインクリメントしながら、ステップＳ１６０７の判定による探索処理を実行する。ステップＳ１６０７で、ＣＰＵ２０１は、変数ＳＥＧ＿ＩＮＦ［ｓ］．ＷＡＶＥ＿ＩＮＦ［ｉ］．ＷＡＶＥ＿ＮＵＭに格納されている、変数ｓの値が示すセグメント内のｉ番目（＝変数ｉの値）の波形データの指定された音色中の波形番号が、変数ｗが示す現在転送中の波形データの指定された音色中の波形番号と等しいか否かを判定する。

【０１２８】
前述した波形読込み指示ルーチン内のステップＳ１５０５の処理が予め実行されていることにより、ステップＳ１６０７とステップＳ１６０８の繰返し処理においてステップＳ１６０７の判定は必ずＹｅｓになる。ステップＳ１６０７の判定がＹｅｓになると、ＣＰＵ２０１は、大容量フラッシュメモリ２０４中の変数ＣＵＲ＿ＴＯＮＥの値が示す音色中の変数ｗの値が示す波形番号の波形データを、波形メモリ２０６中の変数ｓの値が示すセグメント内のｉ番目（＝変数ｉが示す値）の波形データとして転送する波形転送ルーチンを実行する（ステップＳ１６０９）。このとき、ＣＰＵ２０１は、引数として、ＲＡＭ２０３上の変数ＦＡ、ＳＡ、Ｌ、及びＳをそれぞれセットする。ＣＰＵ２０１は、変数ＦＡには、転送される波形データの転送元の大容量フラッシュメモリ２０４上のアドレスとして、変数ＣＵＲ＿ＴＯＮＥと変数ｗとで参照されるＲＯＭ２０２上の定数ＴＯＮＥ＿ＩＮＦ［ＣＵＲ＿ＴＯＮＥ］．ＷＡＶＥ［ｗ］．ＳＥＧ＿ＡＤＲＳの値をセットする。また、ＣＰＵ２０１は、変数ＳＡには、波形データが転送される転送先の波形メモリ２０６中のアドレスオフセットとして、変数ｓと変数ｉとで参照されるＲＡＭ２０３上の変数ＳＥＧ＿ＩＮＦ［ｓ］．ＷＡＶＥ＿ＩＮＦ［ｉ］．ＳＥＧ＿ＡＤＲＳの値をセットする。さらに、ＣＰＵ２０１は、変数Ｌには、転送される波形データのサイズとして、変数ｓと変数ｉとで参照されるＲＡＭ２０３上の変数ＳＥＧ＿ＩＮＦ［ｓ］．ＷＡＶＥ＿ＩＮＦ［ｉ］．ＳＩＺＥの値をセットする。そして、ＣＰＵ２０１は、変数Ｓには、変数ｓの値をセットする。波形転送ルーチンの詳細については、後述する。
【０１２９】
ステップＳ１６０９の波形転送ルーチンの処理の後、ＣＰＵ２０１は、変数ｓの値に対応する現在のセグメント中のｉ番目（＝変数ｉの値）の波形データに対応する、波形データが読込み済みであるか否かを示す変数ＳＥＧ＿ＩＮＦ［ｓ］．ＷＡＶＥ＿ＩＮＦ［ｉ］．ＷＡＶＥに、波形が読み込まれたことを示す値１をセットする（ステップＳ１６１０）。
【０１３０】
更に、ＣＰＵ２０１は、音色中で次に転送されるべき波形番号の値を示す変数ＲＥＡＤＩＮＧ＿ＷＡＶＥの値を＋１インクリメントする（ステップＳ１６１１）。
【０１３１】
その後、ＣＰＵ２０１は、変数ＣＵＲ＿ＴＯＮＥが示す音色中の変数ｗが示す波形番号の波形データに対応する定数ＴＯＮＥ＿ＩＮＦ［ＣＵＲ＿ＴＯＮＥ］．ＷＡＶＥ［ｗ］．ＳＥＧ＿ＧＲＯＵＰのセグメントグループの値が、変数ＣＵＲ＿ＴＯＮＥが示す音色に対応する定数ＴＯＮＥ＿ＩＮＦ［ＣＵＲ＿ＴＯＮＥ］．ＮＵＭ＿ＳＥＧに格納された使用セグメント数に達したか否かを判定する（ステップＳ１６１２）。
【０１３２】
ステップＳ１６１２の判定がＮｏならば、ＣＰＵ２０１は、変数ＲＥＡＤＩＮＧ＿ＷＡＶＥの値をステップＳ１６１１でインクリメントされた値に維持して、図１６のフローチャートで示される図９のステップＳ９０５の波形読込み処理を終了する。この結果、次に、ステップＳ９０５の波形読込み処理が実行されるタイミングで、変数ＲＥＡＤＩＮＧ＿ＷＡＶＥの新たな値に対する波形番号の波形データの転送が実行される。
【０１３３】
一方、ステップＳ１６１２の判定がＮｏになったならば、新たに指定された音色の波形データの全ての転送が完了したため、ＣＰＵ２０１は、変数ＲＥＡＤＩＮＧ＿ＷＡＶＥに、全て読込み済みであることを示す値「－１」を格納する。その後、ＣＰＵ２０１は、図１６のフローチャートで示される図９のステップＳ９０５の波形読込み処理を終了する。
【０１３４】
図１７は、図１６のステップＳ１６０９の波形転送ルーチンの詳細例を示すフローチャートである。
【０１３５】
まず、ＣＰＵ２０１は、大容量フラッシュメモリ２０４の読出しアドレスを指定するＲ

ＡＭ２０３上の変数ｒｐ（リードポインタ）に、図１６のステップＳ１６０９の開始時にＲＡＭ２０３上の変数ＦＡとして引き渡された転送される波形データの転送元の大容量フラッシュメモリ２０４上のアドレスをセットする（ステップＳ１７０１）。

【０１３６】

次に、ＣＰＵ２０１は、波形メモリ２０６への書込みアドレスを指定するＲＡＭ２０３上の変数ｗｐ（ライトポインタ）に、図１６のステップＳ１６０９の開始時にＲＡＭ２０３上の変数Ｓ及びＳＡとしてそれぞれ引き渡されたセグメント番号及びセグメント内アドレスに基づいて、次式により計算される内容をセットする（ステップＳ１７０２）。

【０１３７】

ｗｐ＝１０００００Ｈ×Ｓ＋ＳＡ

【０１３８】

この式において、１セグメントのサイズは前述したように例えば１０００００Ｈバイトであるため、変数Ｓが示すセグメント番号のセグメントの先頭のバイトアドレスは、波形メモリ２０６上において、１０００００Ｈ×Ｓとなる。更に、今回の波形データが転送されるアドレスは、このアドレスに、変数ＳＡとして引き渡されるセグメント内アドレスだけオフセットしたアドレスとなる。

【０１３９】

その後、ＣＰＵ２０１は、転送されるバイトデータ数を指定するＲＡＭ２０３上の変数ｃの値を、ステップＳ１７０３で０にリセットした後、ステップＳ１７０６で１サンプルずつインクリメントしながら、ステップＳ１７０７でＲＡＭ２０３上の変数Ｌとして引き渡されるデータサイズに達したと判定されるまで、ステップＳ１７０４とＳ１７０５の処理を繰返し実行する。

【０１４０】

まず、ＣＰＵ２０１は、大容量フラッシュメモリ２０４上の、変数ｒｐの値と変数ｃの値を加算して得たアドレスから、波形データのサンプル（バイト値）を読み出してＲＡＭ２０３上の変数ａにセットする（ステップＳ１７０４）。

【０１４１】

そして、ＣＰＵ２０１は、波形メモリ２０６上の、変数ｗｐの値と変数ｃの値を加算して得たアドレスに、変数ａに格納されているデータを書き込む（ステップＳ１７０５）。

【０１４２】

以上の繰返し処理の結果、ステップＳ１７０７の判定がＹｅｓになると、ＣＰＵ２０１は、図１７のフローチャートで示される図１６のステップＳ１６０９の波形転送ルーチンの処理を終了する。

【０１４３】

図１８は、図９のステップＳ９０８の押鍵処理の詳細例を示すフローチャートである。

【０１４４】

まずＣＰＵ２０１は、ＲＡＭ２０３上の変数ＣＵＲ＿ＴＯＮＥに音色が選択されていないことを示す無効値「－１」が格納されているか否かを判定する（ステップ１８０１）。

【０１４５】

ステップＳ１８０１の判定がＹｅｓならば、ＣＰＵ２０１は、図１８のフローチャートで示される図９のステップＳ９０８の押鍵処理を終了し、発音処理を実行しない。

【０１４６】

ステップＳ１８０１の判定がＮｏならば、ＣＰＵ２０１は、ＲＡＭ２０３上の変数ＲＥＡＤＩＮＧ＿ＷＡＶＥの値が波形転送を完了した事を示す値「－１」になっているか否かを判定する（ステップＳ１８０２）。

【０１４７】

ステップＳ１８０２の判定がＮｏならば、ＣＰＵ２０１は、図１８のフローチャートで示される図９のステップＳ９０８の押鍵処理を終了し、発音処理を実行しない。即ち、ＣＰＵ２０１は、波形転送が完了して変数ＲＥＡＤＩＮＧ＿ＷＡＶＥの値が「－１」になるまで、発音処理を実行しない。

【０１４８】
ステップＳ１８０２の判定がＹｅｓならば、ＣＰＵ２０１は、スプリット波形検索ルーチンを実行する（ステップＳ１８０３）。この場合、ＣＰＵ２０１は、引数として、図９のステップＳ９０２で取得しているキー番号とベロシティをそれぞれ、ＲＡＭ２０３上の変数Ｋ及びＶに格納する。このルーチンの処理により、例えば図６のスプリットエリア内の１つが判別され、ＲＡＭ２０３上の変数ＴＷに、現在の音色内の波形番号が戻り値として得られる。なお、変数ＴＷの戻り値として「－１」がセットされている場合は、該当するスプリットエリアが無い場合である。
【０１４９】
次に、ＣＰＵ２０１は、変数ＴＷの戻り値が「－１」であるか否かを判定する（ステップＳ１８０４）。
【０１５０】
ステップＳ１８０４の判定がＹｅｓならば、ＣＰＵ２０１は、図１８のフローチャートで示される図９のステップＳ９０８の押鍵処理を終了し、発音処理を実行しない。
【０１５１】
ステップＳ１８０４の判定がＮｏならば、ＣＰＵ２０１は、セグメント内波形検索ルーチンを実行する（ステップＳ１８０５）。このとき、ＣＰＵ２０１は、引数として、変数ＴＷに格納されている音色内の波形番号を引き渡す。このルーチンの処理により、ＲＡＭ２０３上の変数Ｓに、セグメント番号が戻り値として得られる。この変数Ｓの戻り値が６４の場合は該当する波形データが存在しない場合である。また、ＲＡＭ２０３上の変数ＳＷに、セグメント内の波形番号が戻り値として得られる。
【０１５２】
ステップＳ１８０５のセグメント内波形検索ルーチンの処理の後、ＣＰＵ２０１は、変数Ｓの戻り値が６４であるか否かを判定する（ステップＳ１８０６）。
【０１５３】
ステップＳ１８０６の判定がＹｅｓならば、ＣＰＵ２０１は、図１８のフローチャートで示される図９のステップＳ９０８の押鍵処理を終了し、発音処理を実行しない。
【０１５４】
ステップＳ１８０６の判定がＮｏならば、ＣＰＵ２０１は、音源ＬＳＩ２０５に通知する波形データのアドレスとして、ＲＡＭ２０３上の変数ａに、次式により計算されるアドレスをセットする（ステップＳ１８０７）。
【０１５５】
ａ＝Ｓ×１００００Ｈ
＋ＳＥＧ＿ＩＮＦ［Ｓ］．ＷＡＶＥ＿ＩＮＦ［ＳＷ］．ＳＥＧ＿ＡＤＲＳ
【０１５６】
図１７のステップＳ１７０２の場合と同様に、変数Ｓが示すセグメント番号のセグメントの先頭のバイトアドレスは、波形メモリ２０６上において、１０００００Ｈ×Ｓとなる。この値に、定数ＳＥＧ＿ＩＮＦ［Ｓ］．ＷＡＶＥ＿ＩＮＦ［ＳＷ］．ＳＥＧ＿ＡＤＲＳに格納されている変数Ｓが示すセグメントの変数ＳＷが示す波形番号の波形データのセグメント内オフセットアドレスの値を加算した値が、音源ＬＳＩ２０５が読み出すべき波形メモリ２０６中の先頭アドレスとなる。
【０１５７】
その後、ＣＰＵ２０１は、音源ＬＳＩ２０５に対して発音ルーチンを実行する（ステップＳ１８０８）。このとき、ＣＰＵ２０１は、変数Ｋに格納されたキー番号と、変数Ｖに格納されたベロシティと、変数ａに格納されたアドレス値を、引数として音源ＬＳＩ２０５に引き渡す。また、ＣＰＵ２０１は、その他波形情報として、定数ＴＯＮＥ＿ＩＮＦ［ＣＵＲ＿ＴＯＮＥ］．ＷＡＶＥ［ＳＷ］．ＴＧ＿ＩＮＦによって示されるパラメータ群（図７参照）を、引数として音源ＬＳＩ２０５に引き渡す。音源ＬＳＩ２０５の処理は、既存の処理であるためその詳細については省略するが、上記引数に基づいて、波形メモリ２０６から波形データを読み出して発音処理を実行する。

【０１５８】
図１９は、図１８のステップＳ１８０３のスプリット波形検索ルーチンの詳細例を示すフローチャートである。
【０１５９】
ＣＰＵ２０１は、音色内の波形番号を示すＲＡＭ２０３上の変数ｗの値を、ステップＳ１９０１で０にリセットした後、ステップＳ１９０６で＋１ずつインクリメントしながら、ステップＳ１９０７で６４に達したと判定されるまで、ステップＳ１９０２からＳ１９０５の一連の判定処理を繰返し実行する。
【０１６０】
上述の繰返し処理において、ＣＰＵ２０１はまず、変数Ｖとして引き渡されたベロシティが、変数ＣＵＲ＿ＴＯＮＥと変数ｗとで参照されるＲＯＭ２０２上の定数ＴＯＮＥ＿ＩＮＦ［ＣＵＲ＿ＴＯＮＥ］．ＷＡＶＥ［ｗ］．ＶＥＬ＿ＬＯによる最低ベロシティ値以上であるか否かを判定する（ステップＳ１９０２）。
【０１６１】
ステップＳ１９０２の判定がＮｏならば、ステップＳ１９０６の処理に移行して次の波形番号の処理に移る。
【０１６２】
ステップＳ１９０２の判定がＹｅｓならば、次にＣＰＵ２０１は、変数Ｖとして引き渡されたベロシティが、変数ＣＵＲ＿ＴＯＮＥと変数ｗとで参照されるＲＯＭ２０２上の定数ＴＯＮＥ＿ＩＮＦ［ＣＵＲ＿ＴＯＮＥ］．ＷＡＶＥ［ｗ］．ＶＥＬ＿ＨＩによる最高ベロシティ値以下であるか否かを判定する（ステップＳ１９０３）。
【０１６３】
ステップＳ１９０３の判定がＮｏならば、ステップＳ１９０６の処理に移行して次の波形番号の処理に移る。
【０１６４】
ステップＳ１９０３の判定がＹｅｓならば、次にＣＰＵ２０１は、変数Ｋとして引き渡されたキー番号が、変数ＣＵＲ＿ＴＯＮＥと変数ｗとで参照されるＲＯＭ２０２上の定数ＴＯＮＥ＿ＩＮＦ［ＣＵＲ＿ＴＯＮＥ］．ＷＡＶＥ［ｗ］．ＫＥＹ＿ＬＯによる最低キー番号以上であるか否かを判定する（ステップＳ１９０４）。
【０１６５】
ステップＳ１９０３の判定がＮｏならば、ステップＳ１９０６の処理に移行して次の波形番号の処理に移る。
【０１６６】
ステップＳ１９０４の判定がＹｅｓならば、次にＣＰＵ２０１は、変数Ｋとして引き渡されたキー番号が、変数ＣＵＲ＿ＴＯＮＥと変数ｗとで参照されるＲＯＭ２０２上の定数ＴＯＮＥ＿ＩＮＦ［ＣＵＲ＿ＴＯＮＥ］．ＷＡＶＥ［ｗ］．ＫＥＹ＿ＨＩによる最高キー番号以下であるか否かを判定する（ステップＳ１９０５）。
【０１６７】
ステップＳ１９０４の判定がＮｏならば、ステップＳ１９０６の処理に移行して次の波形番号の処理に移る。
【０１６８】
ステップＳ１９０５の判定がＹｅｓならば、ＣＰＵ２０１は、変数ｗの値を、音色内波形番号の戻り値として変数ＴＷにセットする。その後、ＣＰＵ２０１は、図１９のフローチャートで示される図１８のステップＳ１８０３のスプリット波形検索ルーチンを終了する。
【０１６９】
ステップＳ１９０５の判定がＮｏならば、ステップＳ１９０６の処理に移行して次の波形番号の処理に移る。
【０１７０】
ステップＳ１９０２からＳ１９０７の繰返し処理の結果、ステップＳ１９０７の判定が

Ｙｅｓになると、ＣＰＵ２０１は、変数ＴＷに、該当無しを示す値「－１」をセットする。その後、ＣＰＵ２０１は、図１９のフローチャートで示される図１８のステップＳ１８０３のスプリット波形検索ルーチンを終了する。

【０１７１】

図２０は、図１８のステップＳ１８０５のセグメント内波形検索ルーチンの詳細例を示すフローチャートである。

【０１７２】

ＣＰＵ２０１は、セグメントをカウントするＲＡＭ２０３上の変数ｓの値を、ステップＳ２００１で０にリセットした後、ステップＳ２００９で＋１ずつインクリメントしながら、ステップＳ２０１０で６４に達したと判定されるまで、ステップＳ２００２からＳ２００８の一連の処理を繰返し実行する。

【０１７３】

上述の繰返し処理において、ＣＰＵ２０１はまず、ＲＡＭ２０３上の変数ＳＥＧ＿ＩＮＦ［Ｓ］．ＴＯＮＥ（図８参照）の値として変数Ｓの値に対応するセグメントに設定されている音色番号が、ＲＡＭ２０３上の変数ＣＵＲ＿ＴＯＮＥ（図８参照）の値として現在選択されている音色番号に等しいか否かを判定する（ステップＳ２００２）。

【０１７４】

ステップＳ２００２の判定がＮｏならば、ＣＰＵ２０１は、ステップＳ２００９の処理に移行して次のセグメンに対する処理に移る。

【０１７５】

ステップＳ２００２の判定がＹｅｓならば、ＣＰＵ２０１は、セグメント内の波形番号を保持するＲＡＭ２０３上の変数ＳＷの値を、ステップＳ２００３で０にリセットした後、ステップＳ２００７で＋１ずつインクリメントしながら、ステップＳ２００８で６４に達したと判定されるまで、ステップＳ２００４からＳ２００７の一連の処理を繰返し実行する。

【０１７６】

上述の繰返し処理において、ＣＰＵ２０１はまず、変数Ｓ及び変数ＳＷに基づいて変数ＳＥＧ＿ＩＮＦ［Ｓ］．ＷＡＶＥ＿ＩＮＦ［ＳＷ］．ＷＡＶＥ＿ＮＵＭの値として参照される変数Ｓの値に対応するセグメント内のＳＷ番目（＝変数ＳＷの値）の波形データの音色内での波形番号を、ＲＡＭ２０３上の変数ｗに格納する（ステップＳ２００４）。

【０１７７】

次に、ＣＰＵ２０１は、変数ｗの値が、引数として引き渡されたＲＡＭ２０３上の変数ＴＷが示す音色内の波形番号に等しいか否かを判定する（ステップＳ２００５）。

【０１７８】

ステップＳ２００５の判定がＮｏならば、変数ＴＷの値が「－１」であるか否かを判定する（ステップＳ２００６）。

【０１７９】

ステップＳ２００６の判定がＮｏならば、ＣＰＵ２０１は、ステップＳ２００７に移行して、次のセグメント内波形番号の波形データに対する処理に移る。

【０１８０】

ステップＳ２００６の判定がＹｅｓならば、ＣＰＵ２０１は、ステップＳ２００９に移行して、次のセグメントに対する処理に移る。

【０１８１】

上述の繰返し処理において、ステップＳ２００５の判定がＹｅｓになると、ＣＰＵ２０１は、そのときの変数Ｓの値と変数ＳＷの値をそれぞれ、セグメントの戻り値及びセグメント内波形番号の戻り値としてセットし、図２０のフローチャートで示される図１８のステップＳ１８０５のセグメント内波形検索ルーチンの処理を終了する。

【０１８２】

上述の繰返し処理において、ステップＳ２０１０の判定がＹｅｓになると、ＣＰＵ２０１は、変数Ｓの戻り値が該当無しを示す値にセットして、図２０のフローチャートで示さ

れる図１８のステップＳ１８０５のセグメント内波形検索ルーチンの処理を終了する。

【０１８３】

以上説明した実施形態により、波形メモリ上に波形が存在しない音色が選択されて新たな音色波形データを波形メモリに読み込む際に、セグメントが同じ使用数の音色を開放すれば、確実に新しい波形を読み込むことが可能となる。例えば、３セグメント使用している音色波形を選択した際には、３セグメント以上使用している音色に上書きを行えば良い。

これにより読み込みたい波形の容量よりもはるかに大きな空きエリアがありながら読み込めないという非効率な状態を回避できるので、演奏に支障をあたえるような時間のかかるガベージコレクションなどのメモリ整理をする必要から開放される。

また、波形メモリ上に読み込まれた音色の組み合わせが同じであるなら、選択された音色の波形を読み込むための条件、例えばその状態で読み込める、ある音色を上書きすれば読み込める等は、過去に波形を読み込んだ順序に関係なく予想ができるので、管理がし易い電子楽器を実現することが可能となる。

【０１８４】

以上の実施形態に関して、更に以下の付記を開示する。

（付記１）

複数のグループに分割された複数の波形データを有する音色波形データを複数種記憶した一次記憶装置から、前記複数種の音色波形データのいずれかを指定する指定処理と、

夫々が予め定められたサイズの記憶領域からなる複数のセグメントを有し、前記セグメントのそれぞれに前記波形データが記憶可能な二次記憶装置に、前記指定された音色波形データが記憶されているか否かを判別する判別処理と、

前記指定された音色波形データが記憶されていないと判別された場合に、前記指定された音色波形データ内の複数のグループの数に対応する数のセグメントを前記二次記憶装置内で確保するセグメント確保処理と、

前記確保された数のセグメントの領域に、前記一次記憶装置に記憶されている前記指定された音色波形データを読み込む読込み処理と、

を実行する処理部を備えた波形読込み装置。

（付記２）

前記音色波形データ内のグループそれぞれに含まれる波形データのサイズは、前記二次記憶装置の前記各セグメントのサイズを越えない、付記１記載の波形読込み装置。

（付記３）

前記処理装置はさらに、前記二次記憶装置が有する複数のセグメントのうち、空きのセグメントの数を取得する取得処理を実行し、

前記セグメント確保処理は、前記取得された空きのセグメント数が前記指定された音色波形データのグループ数と同じあるいはそれ以上の場合は、前記指定された音色波形データのグループ数と同じ数の空きのセグメントを確保し、前記取得された空きのセグメント数が前記指定された音色波形データのグループ数より少ない場合は、前記空きのセグメントと、前記空いていないセグメントの中で予め定められた条件を満たすセグメントとを確保する処理を実行する、付記１又は２記載の波形読込み装置。

（付記４）

前記予め定められた条件を満たすセグメントは、前記二次記憶装置に記憶されている音色波形データのうち、記憶されている期間が最も長い音色波形データを記憶しているセグメントである、付記１乃至３の何れかに記載の波形読込み装置。

（付記５）

前記一次記憶装置は、フラッシュメモリを備え、二次記憶装置は、一次記憶装置より容量の小さいランダムアクセスメモリを備えた、付記１乃至４のいずれかに記載の波形読込み装置。

（付記６）

波形読込み装置に用いられる波形読込み方法であって、前記波形読込み装置が、

複数のグループに分割された複数の波形データを有する音色波形データを複数種記憶した一次記憶装置から、前記複数種の音色波形データのいずれかを指定し、
夫々が予め定められたサイズの記憶領域からなる複数のセグメントを有し、前記セグメントのそれぞれに前記波形データが記憶可能な二次記憶装置に、前記指定された音色波形データが記憶されているか否かを判別し、
前記指定された音色波形データが記憶されていないと判別された場合に、前記指定された音色波形データ内の複数のグループの数に対応する数のセグメントを前記二次記憶装置内で確保し、
前記確保された数のセグメントの領域に、前記一次記憶装置に記憶されている前記指定された音色波形データを読み込む、波形読み込み方法。

（付記７）
波形読込み装置として用いられるコンピュータに、
複数のグループに分割された複数の波形データを有する音色波形データを複数種記憶した一次記憶装置から、前記複数種の音色波形データのいずれかを指定する指定ステップと、
夫々が予め定められたサイズの記憶領域からなる複数のセグメントを有し、前記セグメントのそれぞれに前記波形データが記憶可能な二次記憶装置に、前記指定された音色波形データが記憶されているか否かを判別する判別ステップと、
前記指定された音色波形データが記憶されていないと判別された場合に、前記指定された音色波形データ内の複数のグループの数に対応する数のセグメントを前記二次記憶装置内で確保するセグメント確保ステップと、
前記確保された数のセグメントの領域に、前記一次記憶装置に記憶されている前記指定された音色波形データを読み込む読込みステップと、
を実行させるプログラム。

（付記８）
付記１に記載の波形読込み装置と、
前記一次記憶装置と、
前記二次記憶装置と、
演奏情報を供給する演奏操作子と、
音色を指定する音色指定操作子と、
前記演奏情報の供給に応答して、前記演奏情報と前記指定された音色とに基づいて、前記一次記憶装置から読み出された前記波形データに対応する楽音を生成する音源と、
を備えた電子楽器。

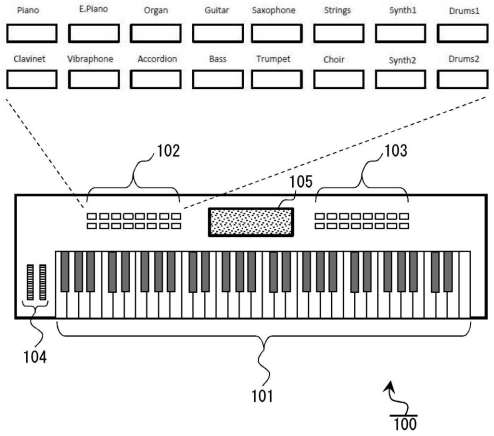
（付記９）
前記音色波形データ内のグループそれぞれに含まれる波形データのサイズは、前記二次記憶装置の前記各セグメントのサイズを越えない、付記８に記載の電子楽器。

【符号の説明】

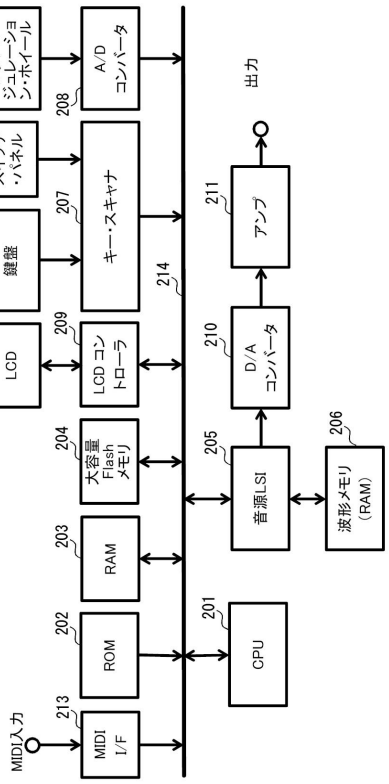
【０１８５】
１００　電子鍵盤楽器
１０１　鍵盤
１０２　音色選択ボタン
１０３　機能選択ボタン
１０４　ベンダ／モジュレーション・ホイール
１０５　ＬＣＤ
２０１　ＣＰＵ
２０２　ＲＯＭ
２０３　ＲＡＭ
２０４　大容量フラッシュメモリ
２０５　音源ＬＳＩ
２０６　波形メモリ

２０７　キー・スキャナ
２０８　Ａ／Ｄコンバータ
２０９　ＬＣＤコントローラ
２１０　Ｄ／Ａコンバータ
２１１　アンプ
２１３　ＭＩＤＩ　Ｉ／Ｆ
２１４　システムバス

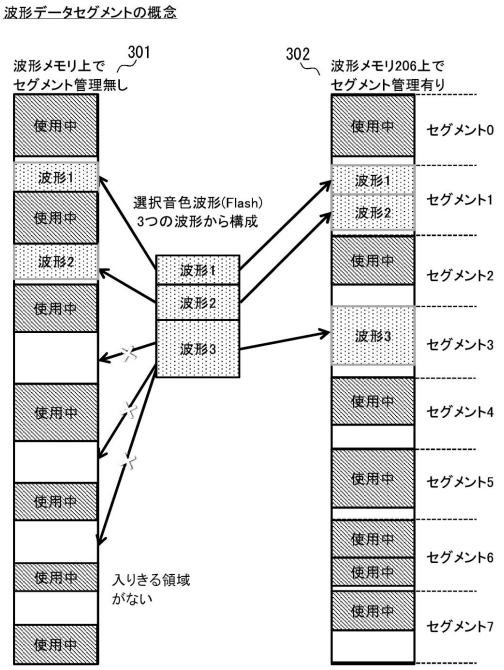
【図１】



【図２】



【図３】



【図４】

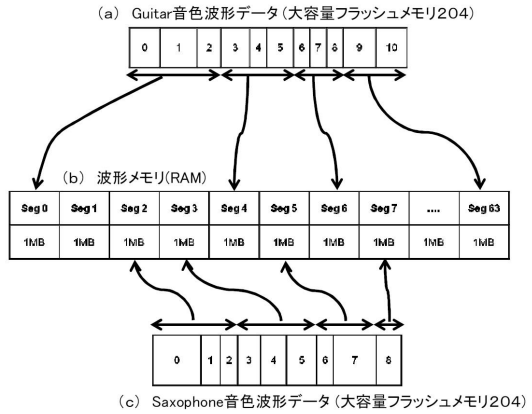
番号	音色名	波形アドレスオフセット（16進数）	波形サイズ（16進数）	使用セグメント数
0	Piano	00000000	00100000	9
1	E.Piano	00100000	00100000	6
2	Organ	00200000	00040000	2
		:		
8	Clavinet	00800000	00080000	5
9	Vibraphone	00900000	000A0000	3
		:		
15	Drums 2	00F00000	00100000	7

（a） フラッシュメモリ音色情報テーブル

波形番号	最低ベロシティ	最大ベロシティ	最低キー番号	最高キー番号	セグメント先頭からのアドレス	波形サイズ	セグメントグループ
0（1A）	0	60	0	40	00000H	0A00H	0
1（2A）	61	127	0	40	00A00H	1100H	0
			:				
10（5B）	80	127	100	127	40000H	0800H	3
			:				
63（不使用）	0	0	0	0	00000H	0	0

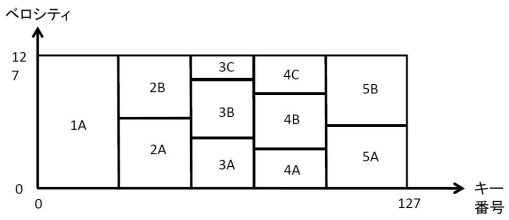
（b） 音色波形情報テーブル

【図５】



（c） Saxophone音色波形データ（大容量フラッシュメモリ204）

【図６】



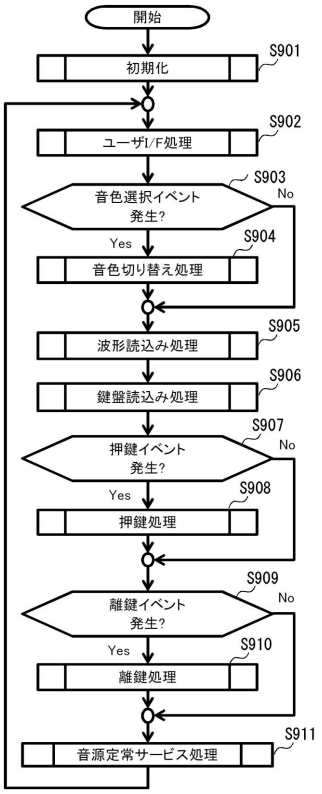
【図７】

上位構造体定数[配列数]	中位構造体定数[配列数]	定数[配列数]	サイズ	値域（****Hは16進数）	説明
TONE_INF[16]	-	NAME[16]	7bit	0-7FH	音色表示名
		WAVE_TOTAL_SIZE	32bit	0-FFFFFFFFH	音色内波形データ合計サイズ
		NUM_SEG	8bit	1-64	音色内波形データ使用セグメント数
	WAVE[64]	VEL_LO	8bit	0-127	波形データが発音する範囲の本来の最小ベロシティ
		VEL_HI	8bit	0-127	波形データが発音する範囲の本来の最大ベロシティ
		KEY_LO	8bit	0-127	波形データが発音する範囲の本来の最低キー番号
		KEY_HI	8bit	0-127	波形データが発音する範囲の本来の最高キー番号
		FLASH_ADRS	32bit	0-FFFFFFFFH	波形データ先頭のフラッシュメモリ上での先頭アドレス
		SEG_ADRS	32bit	0-100000H	音色の持つ波形データの読み込まれるべきセグメントの先頭からのアドレスオフセット
		SIZE	32bit	0-100000H	波形データのサイズ(バイト)0の時は波形無し
		SEG_GROUP	8ibt	0-63	セグメントのグループ番号同じ番号を持つ波形データは同じセグメントに読み込まれる
		TG_INF	構造体	-	音源による波形データの発音のための情報（スタート、ループ、エンドアドレス、音量、チューニング情報等）

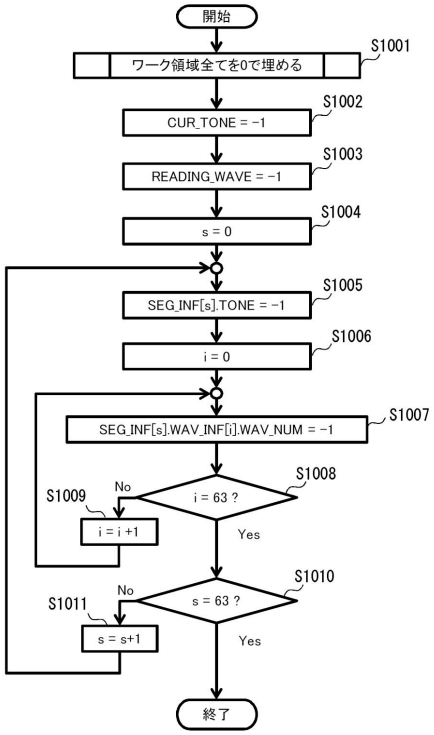
【図８】

上位構造体変数[配列数]	中位構造体変数[配列数]	変数[配列数]	サイズ	値域	初期値	説明
–	–	TIMESTAMP	32bit	0–FFFFFFFFH	0	音色選択イベント毎に1インクリメントする履歴番号。セグメントが割り当てられる際にSEG_INF[].TIMESATMPに格納される。
–	–	CUR_TONE	8bit	0–15, –1	–1	現在鍵盤演奏のために選択されている音色番号。–1は該当なし。
SEG_INF[64]		READING_WAVE	8bit	0–63	–1	波形転送中の波形データの波形番号。–1は全て読込み済み
	–	TONE	8bit	0–15, –1	–1	当該セグメントが割り当てられている音色番号。–1は空状態(未使用)
	–	SEG_GROUP	8bit	0–63	0	当該セグメントに割り当てられている音色のセグメントグループ値。
	–	TIMESTAMP	32bit	0–FFFFFFFFH	0	当該セグメントが使用された履歴。音色選択ごとに1インクリメントする。同時に割り当てられたセグメントは同じTIME_STAMPを持つ。
	WAV_INF[64]	WAVE	1bit	0–1	0	当該セグメント中の当該波形データの有無。0…無し、1…有り(読込み済み)
		WAV_NUM	8bit	0–31, –1	–1	当該セグメント中の当該波形データの音色中の波形番号。–1であれば割当てなし
		SEG_ADRS	32bit	0–FFFFFFFFH	0	当該セグメント中での当該波形データのアドレスオフセット
		SIZE	32bit	0–FFFFFFFFH	0	当該セグメント中の当該波形データのサイズ(バイト)

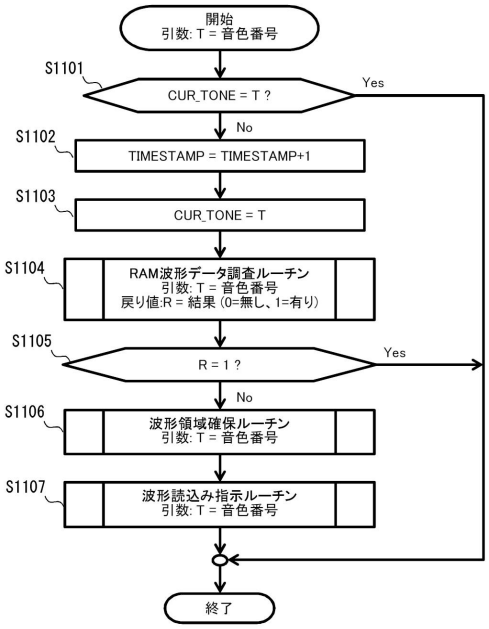
【図９】



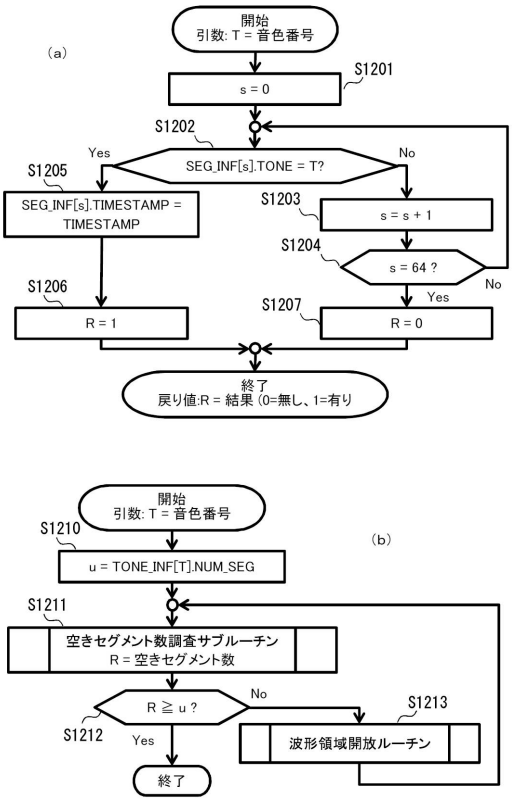
【図１０】



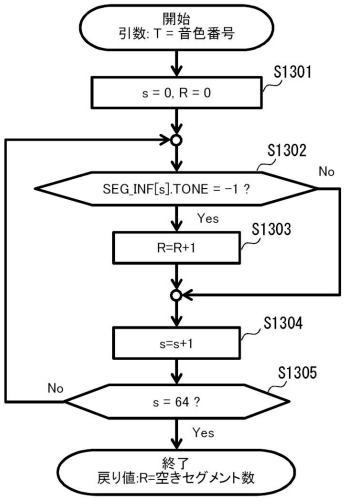
【図１１】



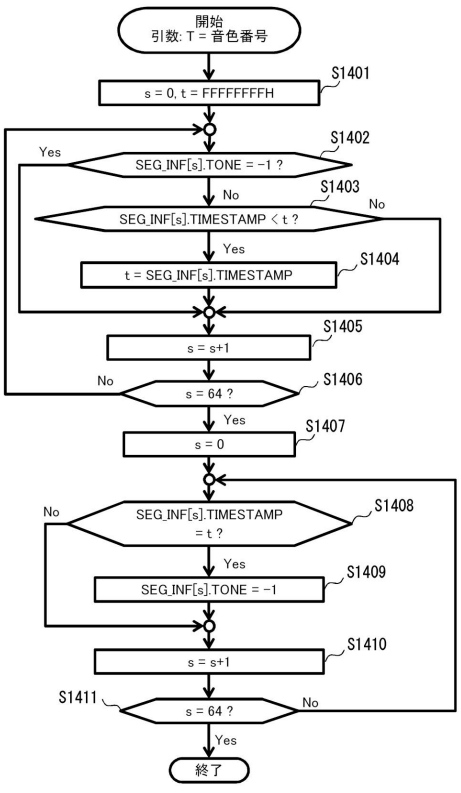
【図１２】



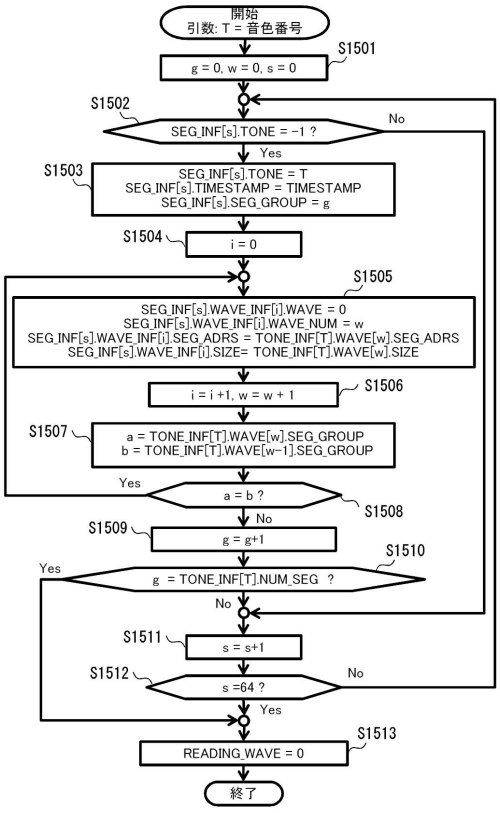
【図１３】



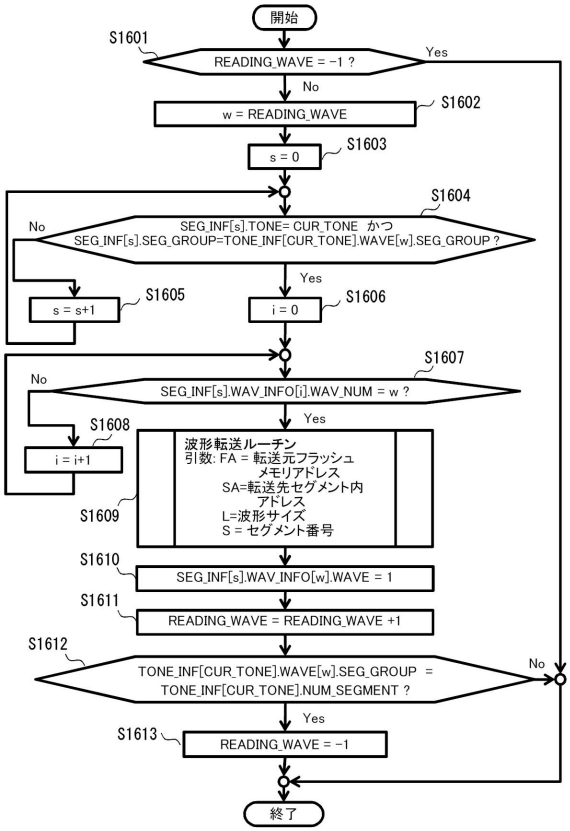
【図１４】



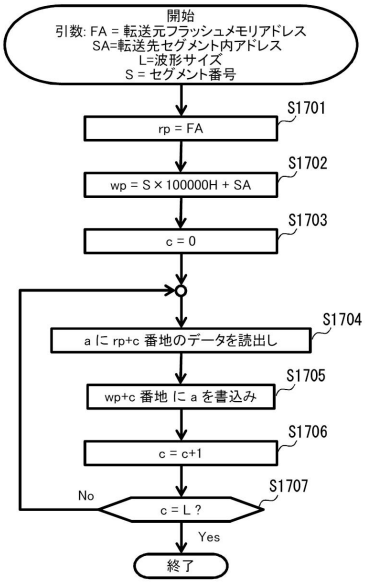
【図１５】



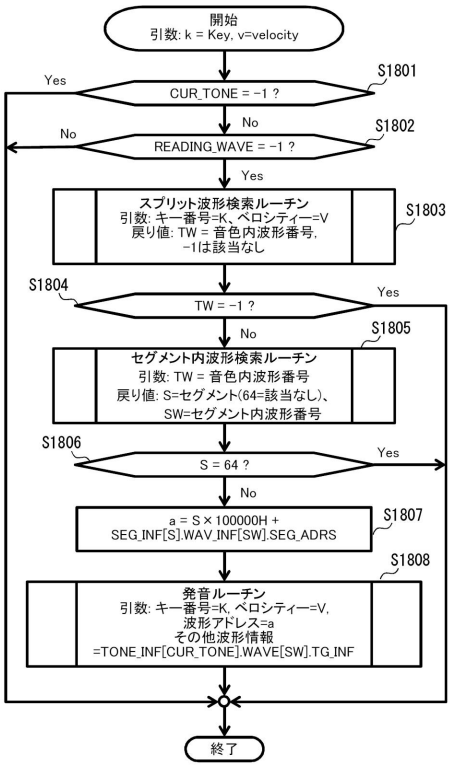
【図１６】



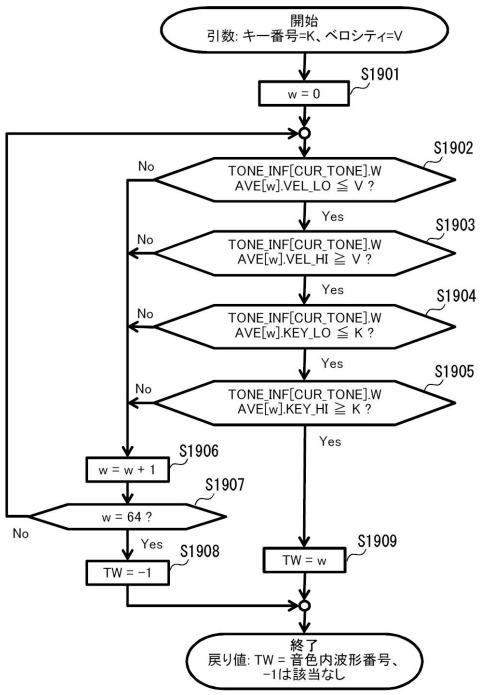
【図１７】



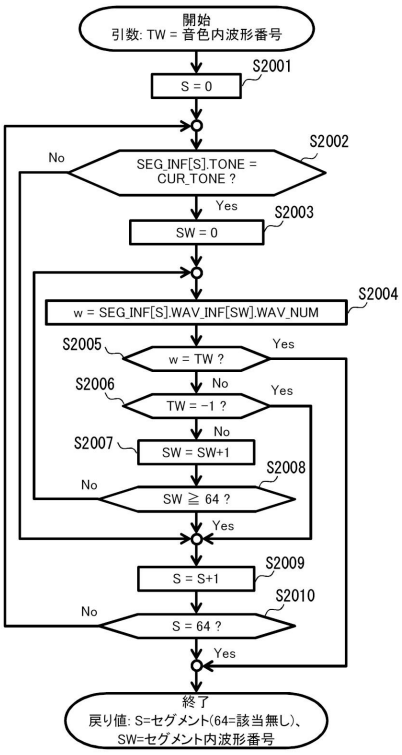
【図１８】



【図１９】



【図２０】


開始
引数: TW = 音色内波形番号
S2001
S = 0
S2002
No
SEG_INF[S].TONE =
CUR_TONE ?
Yes
S2003
SW = 0
S2004
w = SEG_INF[S].WAV_INF[SW].WAV_NUM
S2005
w = TW ?
Yes
No
S2006
TW = −1 ?
Yes
No
S2007
SW = SW+1
S2008
No
SW ≧ 64 ?
Yes
S2009
S = S+1
S2010
No
S = 64 ?
Yes
終了
戻り値: S=セグメント(64=該当無し)、
SW=セグメント内波形番号

フロントページの続き

(56)参考文献 特開２００７－２７１８２７（ＪＰ，Ａ）
特開２０００－２２１９７８（ＪＰ，Ａ）
特開平０８－１４６９６４（ＪＰ，Ａ）
特開平０６－０３５４７３（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ１０Ｈ　　１／００－７／１２