32ビット仮想コンソール

システム仕様

第5回:サウンドチップ(SPU)

資料作成日2023.01.08年Carra著

これは何だ?

このドキュメントは、Vircon32システム仕様のパート5です。この一連のドキュメントは、Vircon32システムを定義し、その機能と動作を詳細に説明する完全な仕様を提供します。

この仕様の主な目標は、Vircon32システムとは何か、および準拠していると見なされるためにゲームシステムがどのように実装される必要があるかについての標準を定義することです。また、Vircon32は仮想システムであるため、これらの文書の重要な第2の目標は、独自のVircon32実装を作成するための知識を誰にでも提供することです。

Vircon32について

Vircon32プロジェクトはCarraによって独自に作成されました。Vircon32システムとその関連資料(ドキュメント、ソフトウェア、ソースコード、アート、およびその他の関連要素を含む)は、元の作成者が所有しています。

Vircon32は無料のオープンソースプロジェクトで、誰でもゲーム機をプレイし、そのためのソフトウェアを開発できるようにすることを目的としている。この詳細については、使用可能な各ソフトウェアに含まれているライセンステキストを参照してください。

このドキュメントについて

このドキュメントは、Creative Commons Attribution 4.0 License(CC BY 4.0)に基づいて提供されています。ライセンスの全文は、Creative CommonsのWebサイトで読むことができます。

<https://creativecommons.org/licenses/by/4.0/>

まとめ

仕様のパート5では、コンソールのサウンドチップ(SPU)を定義しています。このドキュメントでは、このチップの動作、制御ポート、提供されるサウンドコマンド、およびオーディオ出力を生成するために使用されるプロセスについて説明します。

[1 Introduction 3](#_30j0zll)

[2 External connections 3](#_1fob9te)

[3 Working concepts 4](#_3znysh7)

[4 Audio effects 7](#_2et92p0)

[5 Sound generation 9](#_tyjcwt)

[6 Internal variables 11](#_3dy6vkm)

[7 Control ports 15](#_1t3h5sf)

[8 Execution of commands 20](#_4d34og8)

[9 Generation of audio output 22](#_2s8eyo1)

[10 Responses to control signals 23](#_17dp8vu)

## 1はじめに

SPUは、Vircon32コンソールのオーディオチップです。スピーカーから聞こえるすべてのサウンドを作成します。これを行うには、BIOSとカートリッジの両方に含まれるサウンドを再生し、それらに基本的なオーディオエフェクトを適用します。

このサウンドチップは非常にシンプルです。16のサウンドチャンネルの配列を特徴とし、それぞれが任意の時点で使用可能なサウンドの1つを再生できます。出力を生成するために、現在再生中のすべてのチャンネルからのサウンドが混合されます。

サウンドチャネルのアクティビティは、プロセッサによるサウンドコマンドによって制御されます。これ以外にも、チャネルの再生に適用できるオーディオエフェクトがいくつかあります。これについては、後のセクションで説明します。

## 外部接続×2

SPUはコンソールを構成するチップの1つであるため、単独で動作することはできません。この図は、SPUと他のコンポーネントとのすべての通信を示しています。図に示すように、SPUは、含まれるサウンドを再生に使用できるように、使用可能なすべてのオーディオROMへの接続を持っています。明確にするために、仕様のパート2のコンソール図では、意図的にこれらの接続を省略していることに注意してください。

これらの各接続については、以下のセクションで個別に説明します。

### 2.1コントロール信号

すべてのコンソールコンポーネントと同様に、SPUはリセット、新しいフレーム、新しいサイクルのための信号を受信します。これらの信号に対する応答は、このドキュメントのセクション10に詳述されています。

### 2.2制御バス

SPUは、デバイスID=3のスレーブ・デバイスとしてコントロール・バスに接続されます。これにより、バス・マスター(CPU)は、SPUによって公開されるコントロール・ポートに対する読み取りまたは書き込み操作を要求できます。SPUポートのリストとそのプロパティについては、後のセクションで詳しく説明します。

### 2.3 BIOSチップ

BIOSは常に存在し、正確に1つのサウンドを持つオーディオROMを含んでいます。SPUは、ID=-1のサウンドスロットを介してそのサウンドにアクセスできます。

### 2.4カートリッジコントローラ

BIOSと同様に、SPUはIDが0～1023のサウンドスロットを介してカートリッジ内に存在する任意のサウンドにアクセスできます。ただし、カートリッジが存在しない可能性があるため、接続ではカートリッジコントローラをプロキシとして使用する必要があることに注意してください。また、カートリッジが接続されている場合でも、各カートリッジには異なる数のサウンド(0～1024)を含めることができます。

### 2.5スピーカー

SPUは、すべての再生チャネルに対してサウンドを生成し、それをミキシングし、その結果を正しい再生周波数でスピーカーに送信して聞こえるようにする必要があります。これはオーディオ出力接続を介して行われます。セクション9では、この出力がどのように行われるかについて説明します。

## 3作業コンセプト

SPUのサウンド機能またはそれに影響を与える内部変数を説明する前に、SPUが中心となって構築される一連の基本概念を提示しなければならない。

### 3.1サウンドサンプル

SPUによって処理されるオーディオ情報の最小単位はサンプルであり、スピーカーまたはサウンドのいずれかのサウンドシーケンス内の単一の離散値を表します。すべてのサンプルは、左および右チャネルの16ビット値をエンコードする単一の32ビット値として表されます。これは、仕様のパート2ですでに文書化されています。

### 3.2 SPUサウンド

SPUサウンドはサウンドサンプルのシーケンスで、コンソールのオーディオ出力信号と同じフォーマットと再生レートで保存されます。これはステレオの16ビットサンプルで、44100サンプル/秒のレートで再生されます。サウンドは、1サンプルからそれを含むオーディオROMの全サイズまでの任意の長さにすることができます。サウンド内のサンプルには、0から始まる位置で番号が付けられます。すべてのSPUプレイバック機能は、サウンドを使用してオーディオを生成する必要があります。

SPUは、番号付きのサウンドスロットの配列を介して、使用可能なサウンドを識別し、アクセスします。カートリッジで使用できるスロットは1024個あり(サウンドIDは0～1023)、BIOSサウンド用にID=-1の追加スロットがあります。

### 3.3サウンドチャンネル

サウンドチャネルは、SPUサウンドを割り当て、そのサンプルを使用して出力再生を作成できる基本的なオーディオジェネレータです。サウンドチャネルは、基本的な再生コントロールコマンド(再生、一時停止/再開、および停止)をサポートします。SPUは16のサウンドチャネルの配列を特徴とし、それぞれが0～15の数値IDで識別されます。

サウンドチャンネルは、コンソールのオーディオ出力信号と同じ再生フォーマットを使用します。これにより、SPUは現在再生中のすべてのチャンネルの出力を1つの出力に混合することができます。新しいフレームごとに、SPUはフレームの時間に適切な数のサンプルを生成し、それらをバッファに保存します。そこから、SPUは出力オーディオサンプルのシーケンスを正しいタイミングでスピーカーに継続的に送信します。これにより、サウンドチャンネルで再生されているすべてのサウンドが聞こえます。全体として、サウンドチャンネルを使用したSPUの操作は次のようになります:

### 3.4オーディオROMへの接続

SPU自体にはサウンドが含まれていないため、BIOSおよびカートリッジに格納されているサウンドを、それぞれのオーディオROMに接続して読み取る必要があります。オーディオROMは、一連のサウンドを含む読み取り専用のメモリ領域です。これらのサウンドのそれぞれは、1つのサンプルからオーディオROM全体まで、任意の長さを持つことができます。オーディオROMのサイズにはサイズ制限があります:SPUは、カートリッジの場合は256 x 1024 x 1024サンプルまで、BIOSの場合は1024 x 1024サンプルまでのサイズのオーディオROMしか処理できません。

N個のサウンドを含むカートリッジが接続されると、ID=0からの最初のN個のスロットが順に各サウンドに割り当てられます。ID=1023までの残りのスロットは使用されず、アクセスできなくなります。ID=-1のスロットはBIOSサウンドに割り当てられます。BIOSサウンドは存在し、一意であることが保証されます。

この接続プロセスは、新しいカートリッジが挿入されるたびに発生します。ただし、Vircon32システムでは、カートリッジはコンソールがオフの間にのみ挿入できます。したがって、実装では、次のコンソールの電源がオンになるまでこの接続を確立するためにSPUに必要な手順を遅らせることが安全です。

オーディオROMへの接続を確立し、それらのサウンドを見つけて読み取る方法を決定するのは実装次第です。例えば、接続時にオーディオROM内のすべてのサウンドを事前に読み取ることができます。別のオプションは、サウンドへのポインタを保持し、SPUがオンザフライでサンプル値を読み取るようにすることです。

### 3.5選択されたエレメント

SPUは、そのオーディオ要素(サウンドおよびチャンネル)をセットに編成します。サウンド生成では、必要なすべてのチャンネルとサウンドを一度に使用できますが、他のSPU機能では、通常、各セットの1つの要素のみを使用して動作します。

#### 選択したサウンド

割り当てられたすべてのサウンドスロットは、この図に示すように、使用可能なサウンドIDの連続した範囲を形成します。サウンドコマンドを実行したり、内部変数を適用したりするときに使用するサウンドを決定するために、SPUは常にそれらのIDの1つを「選択された」ものと見なします。デフォルトで選択されるサウンドはBIOSサウンドです。これは、常に存在することが保証されている唯一のサウンドであるためです。

#### 選択されたチャンネル

同様に、使用可能なサウンドチャネルは、0～15の範囲の使用可能なチャネルIDを形成します。サウンドコマンドの実行または内部変数の適用時に使用するチャネルを決定するために、SPUは常にこれらのIDの1つを「選択された」ものと見なします。デフォルトで選択されるチャネルは、チャネルID=0に対応する最初のチャネルです。

## 4オーディオエフェクト

柔軟性を高め、より高度なサウンド機能を実現するために、さまざまなタイプのエフェクトを適用して各SPUサウンドチャネルを設定できます。これらのエフェクトを有効にすると、チャネルに割り当てられたサウンドの再生方法が変更されるため、スピーカーから聞こえる結果が変わります。

### 4.1ボリューム効果

サウンドを再生する場合、SPUチャネルは再生ボリュームを変更するように設定できます。チャネルのボリュームパラメータは線形スケーリング係数として機能します。その結果、出力される音波はそれに応じてスケーリングされます。このエフェクトの例は次のようになります。

### 4.2速度変動効果

SPUチャネルのもう1つの設定可能なパラメータは、再生速度です。チャネルの速度パラメータは、割り当てられたサウンドのサンプルに沿って前進するレートを決定するために使用され、速度とピッチの両方が変更されます。変更された再生速度の例は次のようになります:

### 4.3ループ再生

このエフェクトを有効にすると、チャンネルは割り当てられたサウンドの定義された領域にわたって連続的にループします。ループ領域の開始と終了の制限は、SPUサウンドごとに設定できます。ループを有効にしたサウンドの再生は、次に示すように3段階に分けることができます。

### 4.4再生ジャンプ

チャンネルは、サウンドに沿って連続的に進行する再生位置を維持することによって、割り当てられたサウンドを再生します。チャンネルでは、この再生位置をプログラムコマンドによって変更できるため、ジャンプが発生します。たとえば、前方へのジャンプの場合、次の図に示すようにサウンドが再生されます。

## 5音の発生

サウンド生成は、既存のソースSPUサウンドから新しい出力オーディオサンプルを生成するために使用されるプロセスです。これを達成するために、SPUは、まず、構成によって決定されるように、現在再生中の各チャネルから個々の出力サンプルを生成する。これらのチャネルサンプルは、単一のグローバル出力サンプルに混合される。最後に、混合された出力サンプルのシーケンスが格納され、44100 Hzの再生レートによって必要とされるタイミングでオーディオ信号を介してスピーカーに配信される。

グローバルなサウンド生成プロセスは、次の図に要約できます。

### 5.1一般的な手順及び時期

サウンド生成プロセスは、各フレームの先頭でトリガーされます。すべての実行に対して、このプロセスは1フレームに必要な出力サンプルを生成します。これは、44100/60=735サンプルの順序付けられたシーケンスです。これを達成するために、次の手順が735回繰り返されます。

1. 現在再生中の各サウンドチャンネルの現在のサンプルを生成します。
2. ミキシングロジックを適用して、すべてのチャンネルサンプルを1つの出力サンプルに結合します。
3. 出力サンプルを出力バッファに保存し、再生用に送信できるようにします。
4. 現在再生中のすべてのチャンネルを進めます。

これらの各手順については、以降のセクションで詳しく説明します。

フレームの先頭でのみサウンドを生成することの重要な結果は、これらの瞬間の間のサウンドおよびチャネルの状態がサウンド生成に関連しないということです。たとえば、あるフレーム内で、プログラムがチャネル2の音量を0.5に設定したとします。その後、同じフレーム内で、チャネル2の音量が0.8に変更されたとします。その場合:

* サウンドの生成が連続的である場合、チャンネル2はボリューム0.5の出力サンプルを生成し、後にボリューム0.8の他のサンプルを生成しています。
* ただし、サウンドはフレーム間でのみ生成されるため、チャンネル2は最終的なボリューム0.8を使用して、そのフレームのすべてのサンプルを生成します。

同様に、チャンネルが再生するように設定されていて、後で同じフレームで停止した場合、次の新しいフレームで停止されるため、そのフレームではサウンドはまったく生成されません。

### 5.2チャネル出力サンプルの生成

現在再生中のチャンネルは、割り当てられたサウンドから現在の再生位置に対応するサンプルを選択します。このサンプルにチャンネルの再生ボリュームの現在の値が乗算され、その結果の値がチャンネルの出力サンプルとして使用されます。

#### 補間方法

再生速度は変動する可能性があるため、チャネルの再生位置は常に整数であるとは限りません。これにより、実装は、チャネルが選択されたサンプルをどのように選択すべきかを決定するための何らかのロジックを決定する必要があります。

Vircon32 SPUにおける基準音補間法は、最も近い隣接音である。しかし、実装は、非整数の再生位置に対して他の補間法を自由に選択することができる。元の音を歪めない限り、任意の方法が許容される。これは、線形またはキュービックなどの元の波形に基づく音声補間が、Vircon32に準拠していると考えられることを意味する。

### 5.3チャンネルサンプルの混合

現在再生中のすべてのチャネルによって生成された出力は、最初にSPUのグローバルボリューム係数で乗算されます。その後、結果の値がすべて追加されます。この結果が最終的な出力サンプルとして使用されます。

操作の順序が結果に影響することに注意してください。ここで説明する順序(AFTER乗算の追加)は、複数のチャネルが同時に再生され、グローバルボリュームでそれらを変調する必要がある場合に、オーディオ信号の飽和を防ぐために選択されます。

### 5.4混合出力サンプルの保存

サウンドの生成と出力の正確なタイミングを確保するために、何らかのタイプのストレージバッファが必要になります。その機能は、オーディオ出力信号が個々のサンプルをスピーカーに送信するまで、生成された出力サンプルを保持することです。このための可能な戦略については、セクション9で説明します。

### 5.5各サンプル後にチャネルを進める

現在の出力サンプルが生成された後、チャネルは再生位置を進める必要があります。再生速度が設定できない場合、チャネル位置は出力サンプルごとに1サンプルずつ進みます(同じ再生速度)。乗算係数として機能する再生速度を含めると、各出力サンプル後の進みは単純に次のようになります:

| チャンネルの再生位置+=チャンネルの再生速度 |
| --- |

#### ループ処理

進んだ後、チャンネルはループロジックを適用する必要があります。これは、チャンネルループが有効になっていて、サウンドループ設定が有効な場合にのみ行われます。有効な設定のためには、サウンドのループ終了位置はループ開始位置よりも大きくなければなりません。

ループロジックでは、最初に、最後の前進によって再生位置が割り当てられたサウンドの現在のループ終了位置を超えたかどうかを判断する必要があります。これが発生した場合、再生位置はループに戻されるように「ラップアラウンド」されます。ただし、再生速度は1よりもはるかに大きい可能性があるため、このプロセスでは次のことを考慮する必要があります。

* ループの長さ自体よりも大きな値によって、ループの終点がオーバーショットされた可能性があります。
* シームレスループの正しい新しい位置は、ループの開始ではない場合があります。

一般的に、正しいループを作成するには、新しい再生位置を次のように設定する必要があります。

| 1. ループ長=ループ終了-ループ開始+1 2. フルループのオーバーショット=floor((再生位置-ループ終了)/ループの長さ) 3. 部分的なループオーバーショット=ループ位置-(ループ長\*完全なループオーバーショット) 4. 再生位置=ループ開始+部分ループオーバーショット |
| --- |

#### 再生を終了する

必要に応じてループを処理した後、チャネルは、再生位置が割り当てられたサウンドの最後のサンプル位置よりも大きいかどうかをチェックする必要があります。その場合、サウンドの再生は終了します。チャネルは停止し、現在のフレームのサンプルは生成されません。実装がチャネルに対して常に735のサンプルを生成する必要がある場合は、残りのサンプルをゼロで埋めることによってそうすることができる。

## 6内部変数

SPUは、その内部状態のさまざまな側面を格納する変数のセットを特徴とします。これらの変数はそれぞれ32ビット値として格納され、それらはすべて、仕様のパート2で説明されている同じデータ形式(整数、浮動小数点など)を使用して解釈されます。これに対する唯一の例外は、後述するように、「チャネル位置」変数です。ここでは、すべての内部変数をセクションに編成してリストし、詳細を示します。

### 6.1グローバルSPU変数

| グローバルボリューム | **初期値:**1.0 |
| --- | --- |
| **形式:**フロート | **有効範囲:**0.0年から2.0年まで |

SPUの現在の出力ボリュームレベルを表します。この値は線形乗数として機能し、すべてのチャンネルから受信したサンプルにミキサーによって適用されます。したがって、各チャンネルの合計ボリューム乗数は(グローバルボリューム\*チャンネルボリューム)になります。

グローバルボリュームの最も一般的な使用方法は、1より小さい値を使用して、複数のチャンネルをミキシングするときにオーディオが飽和しないようにすることです。

### 6.2選択されたエレメント

| 選択したサウンド | **初期値:**-1個 |
| --- | --- |
| **形式:**整数 | **有効範囲:**-1～1023(\*) |

この値は、現在選択されているSPUサウンドの数値IDです。選択されたサウンドは、サウンド関連のすべての操作で使用されます。また、サウンド設定の変更によって影響を受けるサウンドでもあります。

(\*)上限は、現在接続されているカートリッジによって決まります。カートリッジが存在しない場合は、BIOSサウンド(ID=-1)のみが選択可能なサウンドです。

| 選択されたチャンネル | **初期値:**0 |
| --- | --- |
| **形式:**整数 | **有効範囲:**0から15 |

この値は、現在選択されているSPUサウンドチャネルの数値IDです。選択されたチャネルは、すべてのシングルチャネルサウンドコマンドで使用されます。また、チャネル設定の変更によって影響を受けるチャネルでもあります。

### 6.3各サウンドの構成

ここにリストされている変数は特別なものです。SPUは、既存のSPUサウンドごとにこれらの変数のコピーを保存します。これは、(1024+1)個のコピーが存在する可能性があることを意味します。実装は、これらすべてが常に保存され、未使用のサウンドスロットのものはアクセスできないようにしておくか、新しいカートリッジが挿入されるたびに必要なものだけが作成されるかを決定することができます。

これらの変数の各セットは、単一のSPUサウンドの現在の設定を記述します(SPUサウンドの定義方法の概要については、セクション3を参照してください)。これらの変数の1つのセットのみがいつでもアクセス可能であるため、このセクションの各変数は、「ポインタ」プロキシを介して制御ポートによってアクセスされます。選択されたサウンドが変更されると、これらのポートはすべて、正しいサウンドの変数のコピーにリダイレクトされます。

| 音の長さ | **初期値:**(\*)は |
| --- | --- |
| **形式:**整数 | **有効範囲:**1～268435456(=256\*1024\*1024) |

関連付けられたサウンドのサンプル数を表します。

(\*):その初期値は、カートリッジが接続されたとき、またはコンソールが起動したときに、すでにサンプルのサウンド長に設定されています。カートリッジが取り外されるまで、それは変更されません。

| ループでサウンドを再生 | **初期値:**偽 |
| --- | --- |
| **形式:**ブール値 | **有効範囲:**真/偽 |

このパラメータがtrueの場合、関連付けられたサウンドはループを有効にして再生されるように設定されます。チャネルは、再生コマンドを受信すると、割り当てられたサウンドからこの変数を読み取り、「チャネルループ有効」変数を同じ値に設定します。

| サウンドループの開始 | **初期値:**0 |
| --- | --- |
| **形式:**整数 | **有効範囲:**0～268435455(\*) |

関連付けられたサウンドのループ範囲に含まれる最初のサンプルの位置をマークします。チャネルループが有効になっている場合、再生がループの終了位置を超えると、再生はこのループの開始位置に「ラップアラウンド」します。

(\*)その最大許容値は、最後の音の位置、すなわち音の長さ-1である。

| サウンドループ終了 | **初期値:**(\*)は |
| --- | --- |
| **形式:**整数 | **有効範囲:**0～268435455(\*\*) |

関連するサウンドのループされた範囲に含まれる最後のサンプルの位置をマークします。チャンネルループが有効になっている場合、再生がこのループの終了位置を超えると、再生はループの開始位置に戻ります。

(\*)その初期値は、サウンド内の最後のサンプル位置、すなわち、サウンド長-1である。

(\*\*)その最大許容値は、最後のサンプル位置でもある。すなわち、音響長-1。

### 6.4各音声チャンネルの構成

ここにリストされている変数は、サウンド変数と同じように機能します。SPUは、各SPUサウンドチャネルに対してこれらの変数のコピーを保存します。これは、16のコピーがあることを意味します。

同時に、これらの変数の各セットは、単一のSPUサウンドチャネルの現在の設定を記述します(SPUサウンドチャネルの定義方法の概要については、セクション3を参照)。これらの変数の1つのセットのみがいつでもアクセス可能であるため、このセクションの各変数は、「ポインタ」プロキシを介して制御ポートによってアクセスされます。選択されたサウンドチャネルが変更されると、これらのポートはすべて、正しいチャネルの変数のコピーにリダイレクトされます。

| チャネルの状態 | **初期値:**40h(チャンネル停止) |
| --- | --- |
| **形式:**整数 | **有効範囲:**リストされた値のみ |

この値は、関連付けられたサウンドチャネルの現在の再生状態として解釈されます。設定可能な値は次のとおりです。

40h:チャンネル停止

41h:チャネルが一時停止

42h:チャネル再生

| チャンネルアサインサウンド | **初期値:**-1個 |
| --- | --- |
| **形式:**整数 | **有効範囲:**-1:最後に使用されたサウンドID |

関連付けられたチャンネルに現在割り当てられているサウンドIDを表します。チャンネルが再生コマンドを受信すると、チャンネルに割り当てられたSPUサウンドからのサンプルを使用してサウンドが生成されます。

| チャネルボリューム | **初期値:**1.0 |
| --- | --- |
| **形式:**フロート | **有効範囲:**0.0から8.0 |

関連付けられたチャネルの現在の出力ボリュームレベルを表します。この値は、このチャネルによって生成される出力サンプルに適用される線形サンプル乗数として機能します。

| チャネル速度 | **初期値:**1.0 |
| --- | --- |
| **形式:**フロート | **有効範囲:**0.0から128.0 |

関連付けられたサウンドチャネルの現在の再生速度を表します。1.0の速度は変更されていない再生を表し、その他の値は乗数として解釈されます。たとえば、2.0の速度は、チャネル位置が各出力サンプルごとに2つのサウンドサンプルを進めることを意味します。値が0.0の場合、再生はまったく進みません。

| チャネルループ有効 | **初期値:**偽 |
| --- | --- |
| **形式:**ブール値 | **有効範囲:**真/偽 |

関連付けられたサウンドチャネルのループ再生が現在有効かどうかを判別します。trueの場合、そのチャネルは関連付けられたサウンドのループ領域を再生し続けます。

| チャネル位置 | **初期値:**0.0 |
| --- | --- |
| **形式:**倍精度浮動小数点数型 | **有効範囲:**0.0から(\*) |

関連付けられたサウンドチャネルの現在の再生位置を表します。これは、関連付けられたサウンド内のサンプル位置として解釈されます。このチャネルが次の出力サンプルを生成すると、関連付けられたサウンド内のこの位置にあるサンプルが選択されます。

この変数の内部形式は64ビットIEEE doubleですが、これは例外です。これは、非整数値に対して十分な精度を保持しながら、広い範囲内の位置を表す必要があるためです。ただし、制御ポートは32ビット値しか処理できないため、ポートからの読み取りおよび書き込みではチャネル位置が32ビット整数として処理されます。

(\*):この変数の上限はチャンネルごとに異なり、そのチャンネルに関連付けられたサウンドが変更されるたびに更新されます。この上限は、新しい関連付けられたサウンドの最後のサンプル位置(サウンドの長さ-1)に設定されます。

## 7つの制御ポート

このセクションでは、スレーブ・デバイスとしてのCPUコントロール・バスへの接続を介してSPUによって公開されるコントロール・ポートのセットについて詳しく説明します。公開されるすべてのポートとその基本プロパティを次の表に示します:

| 公開されたコントロールポートのリスト | | | |
| --- | --- | --- | --- |
| 外部アドレス | 内部アドレス | ポート名 | R/Wアクセス |
| 300時間 | 00時間 | コマンド(Command) | 書き込み専用 |
| 301時間 | 01時間 | グローバルボリューム | 読み取り/書き込み |
| 302時間 | 02時間 | 選択されたサウンド | 読み取り/書き込み |
| 303時間 | 03時間 | 選択されたチャネル | 読み取り/書き込み |
| 304時間 | 04時間 | サウンドの長さ | 読み取り専用 |
| 305時間 | 05時間 | ループでサウンドを再生 | 読み取り/書き込み |
| 306時間 | 06時間 | サウンドループの開始 | 読み取り/書き込み |
| 307時間 | 07時間 | サウンドループ終了 | 読み取り/書き込み |
| 308時間 | 08時間 | チャネルの状態 | 読み取り専用 |
| 309時間 | 09時間 | チャンネルアサインサウンド | 読み取り/書き込み |
| 30 Ah弾 | 0 Ah(ゼロ時) | チャンネルボリューム(Channel Volume) | 読み取り/書き込み |
| 30 Bh弾 | 0時 | チャネル速度 | 読み取り/書き込み |
| 30分 | 0 Ch(0チャネル) | チャネルループ有効 | 読み取り/書き込み |
| 30度Dh | 0 Eh(0時) | チャネル位置 | 読み取り/書き込み |

### 7.1ポートの読み取り/書き込み要求に対する動作

SPUコントロール・ポートは、単なるハードウェア・レジスタではありません。特定のポートに対する読み取り/書き込み要求によってトリガーされる効果は、多くの場合、値の読み取りまたは書き込みとは異なります。このセクションでは、各SPUポートの動作について説明します。

実行されるアクションに加えて、コントロール・バス通信の一部としてリクエストに対して成功/失敗レスポンスを提供する必要があることに注意してください。このレスポンスは、特に指定がないかぎり、常に成功とみなされます。提供されたレスポンスが失敗の場合、SPUはそれ以上のアクションを実行せず、CPUはHWエラーをトリガーします。

### コマンドポート

#### オン読み取る要求:

このポートは書き込み専用であるため、障害応答が制御バスに提供されます。

#### オン書く要求:

SPUは、セクション8で詳細に説明されているコマンド実行プロセスを実行します。すべての場合において、コマンドが認識されなくても、要求は正常に応答されます。

### グローバル・ボリューム・ポート

#### オン読み取る要求:

SPUは、内部変数「Global volume」の現在の値を提供します。

#### オン書く要求:

SPUは、受信した値が内部変数「Global volume」の範囲外であるかどうかをチェックします。範囲外である場合は、範囲制限にクランプされます。結果の値は、内部変数「Global volume」を上書きします。これにより、次のサウンド生成で新しいグローバルボリュームがすぐに適用されます。

### 選択したサウンドポート

#### オン読み取る要求:

SPUは、内部変数「Selected sound」の現在の値を提供する。

#### オン書く要求:

SPUは、受信した値が現在有効なサウンドIDに対応しているかどうかをチェックします。対応していない場合、要求は無視されます。有効な値の場合、SPUは内部変数「Selected sound」を受信した値で上書きします。その後、すべてのサウンド設定ポートを、新しく選択されたサウンドの変数のセットを指すようにリダイレクトします。

### 選択されたチャネルポート

#### オン読み取る要求:

SPUは、内部変数「Selected channel」の現在の値を提供する。

#### オン書く要求:

SPUは、受信した値が有効なチャネルIDに対応しているかどうかをチェックします。対応していない場合、要求は無視されます。有効な値の場合、SPUは内部変数「Selected channel」を受信した値で上書きします。その後、すべてのチャネル設定ポートを、新しい選択されたチャネルの変数のセットを指すようにリダイレクトします。

### [Sound Length]ポート

#### オン読み取る要求:

SPUは、現在選択されているサウンドIDに関連付けられた内部変数「Sound length」の現在の値を提供する。

#### オン書く要求:

このポートは読み取り専用であるため、障害応答が制御バスに提供されます。

### [Sound Play With Loop]ポート

#### オン読み取る要求:

SPUは、現在選択されているサウンドIDに関連付けられた内部変数「Sound play with loop」の現在の値を提供する。

#### オン書く要求:

SPUは受信した値を使用して、現在選択されているサウンドIDに関連付けられた内部変数「Sound play with loop」を上書きします。これにより、次のサウンド生成では、選択されたサウンドに新しいループ設定がすぐに適用されます。

### サウンドループ開始ポート

#### オン読み取る要求:

SPUは、現在選択されているサウンドIDに関連付けられた内部変数「Sound loop start」の現在の値を提供する。

#### オン書く要求:

SPUは、受信した値がこのサウンドの内部変数「Sound loop start」の範囲外であるかどうかをチェックし、範囲外である場合は、範囲の制限にクランプされます。結果の値は、現在選択されているサウンドIDに関連付けられている内部変数「Sound loop start」を上書きします。これにより、次のサウンド生成では、選択されたサウンドに新しいループ開始がすぐに適用されます。

### サウンドループエンドポート

#### オン読み取る要求:

SPUは、現在選択されているサウンドIDに関連付けられた内部変数「Sound loop end」の現在の値を提供する。

#### オン書く要求:

SPUは、受信した値がこのサウンドの内部変数「Sound loop end」の範囲外であるかどうかをチェックし、範囲外である場合は、範囲の制限にクランプされます。結果の値は、現在選択されているサウンドIDに関連付けられている内部変数「Sound loop end」を上書きします。これにより、次のサウンド生成では、選択されたサウンドに新しいループエンドがすぐに適用されます。

### [Channel State]ポート

#### オン読み取る要求:

SPUは、現在選択されているチャネルIDに関連する内部変数「チャネル状態」の現在の値を提供する。

#### オン書く要求:

このポートは読み取り専用であるため、障害応答が制御バスに提供されます。

### [Channel Assigned Sound]ポート

#### オン読み取る要求:

SPUは、現在選択されているチャネルIDに関連付けられた内部変数「Channel assigned sound」の現在の値を提供する。

#### オン書く要求:

SPUは、まず、選択されたチャネルIDの状態をチェックし、それがStoppedでない場合、リクエストは無視される。受信された値が有効なサウンドIDでない場合、リクエストも無視される。どちらの場合も、それ以上の処理は実行されない。

要求が処理されると、SPUは受信した値を使用して内部変数「Channel assigned sound」を上書きします。これにより、次の描画操作では、選択したチャネルに対して新しい割り当てサウンドが適用されます。

その後、SPUはチャネルの「Channel position」変数に対して次の処理を行います。

* 値は0に設定されます。
* 上限は、新しいサウンドのサンプル数-1に設定されます。

### [Channel Volume]ポート

#### オン読み取る要求:

SPUは、現在選択されているチャネルIDに関連付けられた内部変数「Channel volme」の現在の値を提供する。

#### オン書く要求:

SPUは、受信した値が内部変数「Channel volume」の範囲外であるかどうかをチェックします。範囲外である場合は、範囲の制限にクランプされます。結果の値は、現在選択されているチャネルIDに関連付けられている内部変数「Channel volume」を上書きします。これにより、次のサウンド生成では、選択されたチャネルに新しいボリュームが適用されます。

### [Channel Speed]ポート

#### オン読み取る要求:

SPUは、現在選択されているチャネルIDに関連する内部変数「チャネル速度」の現在の値を提供する。

#### オン書く要求:

SPUは、受信した値が内部変数「Channel speed」の範囲外であるかどうかをチェックします。範囲外である場合は、範囲の制限にクランプされます。結果の値は、現在選択されているチャネルIDに関連付けられている内部変数「Channel speed」を上書きします。これにより、次のサウンド生成では、選択されたチャネルに新しい速度が適用されます。

### Channel Loop Enabledポート

#### オン読み取る要求:

SPUは、現在選択されているチャネルIDに関連付けられた内部変数「Channel loop enabled」の現在の値を提供する。

#### オン書く要求:

SPUは受信した値を使用して、現在選択されているチャネルIDに関連付けられた内部変数「Channel loop enabled」を上書きします。これにより、次のサウンド生成では、選択されたチャネルに新しいループ設定がすぐに適用されます。

### [Channel Position]ポート

#### オン読み取る要求:

SPUは、現在選択されているチャネルIDに関連付けられた内部変数「Channel position」の現在の値を提供します。この変数はdoubleであるため、提供される値は、その値を32ビット整数に切り捨てた結果になります。

#### オン書く要求:

SPUは、選択されたチャネルIDに現在関連付けられているSPUサウンドのサンプル数をチェックします。この書き込み操作の許容範囲は、[0からサンプル数-1]のように決定され、両方とも含まれます。受信した32ビットの整数値が範囲外の場合は、範囲制限にクランプされます。その後、結果の値は倍精度に変換され、現在選択されているチャネルIDに関連付けられている内部変数「チャネル位置」を上書きするために使用されます。これにより、次のサウンド生成では、選択されたチャネルの新しい再生位置がすぐに適用されます。

## 8コマンドの実行

SPUコマンドは、CPUがSPUコマンドポートに値を送信することによって要求できるチャネル再生制御命令です。このセクションでは、このような要求を受信したときのSPUの動作について説明します。

SPUが実行できるコマンドは6種類あります。最初の3つは単一のサウンドチャネルで動作し、他の3つはすべてのチャネルに影響します。この表は、それらの数値と、各コマンドに使用されるグラフィック効果を示しています。

| コマンド名 | 数値 |
| --- | --- |
| 選択したチャンネルを再生 | 30時間 |
| 選択したチャンネルを一時停止 | 31時間 |
| 選択されたチャネルを停止 | 32時間 |
| すべてのチャネルを一時停止 | 33時間 |
| すべてのチャネルを再開 | 34時間 |
| すべてのチャネルを停止 | 35時間 |

### 8.1共通処理

ここで説明する一般的な操作は、すべてのコマンドの実行に共通です。

最初のステップとして、SPUは、要求された書き込み値が前述の有効なコマンドのいずれかに対応するかどうかをチェックします。対応しない場合、要求は無視され、それ以上の処理は行われません。

### 8.2[Play Selected Channel]コマンド

SPUは、現在選択されているサウンドチャネルの状態を[Playing]に設定します。チャネルの以前の状態が[Stopped]であった場合は、新しい再生を開始するために次のアクションが実行されます。

1. チャンネルの位置は0に設定されます。
2. チャンネルの「Loop enabled」変数は、関連するサウンドの「Play sound with loop」変数と同じ値に設定されます。これにより、サウンドのループ設定が、再生に使用されるチャンネルに自動的に適用されます。

チャネルの以前の状態がすでに[再生中]であった場合は、前述の同じ2つのアクションが適用され、サウンド再生のリトリガーが発生します。以前に[一時停止]状態であったチャネルの場合は、位置は変更されず、サウンド再生が再開されます。

### 8.3[Pause Selected Channel]コマンド

現在選択されているサウンドチャネルの状態が[再生中]の場合、SPUはその状態を[一時停止]に設定します。その他の以前の状態に対してはアクションは実行されません。

### 8.4[Stop Selected Channel]コマンド

SPUは、現在選択されているサウンドチャネルを停止状態に設定します。すでに停止しているチャネルを停止しても効果はありません。

### 8.5[Pause All Channels]コマンド

SPUは、「Pause Selected Channel」コマンドに関して説明したのと同じ処理を適用するが、アクションは、代わりに、「Playing」状態のすべてのチャネルに適用される。

### 8.6 Resume All Channelsコマンド

SPUは、「Play Selected Channel」コマンドに関して説明したのと同じ処理を適用するが、アクションは、その代わりに、「Paused」状態のすべてのチャネルに適用される。

### 8.7 Stop All Channelsコマンド

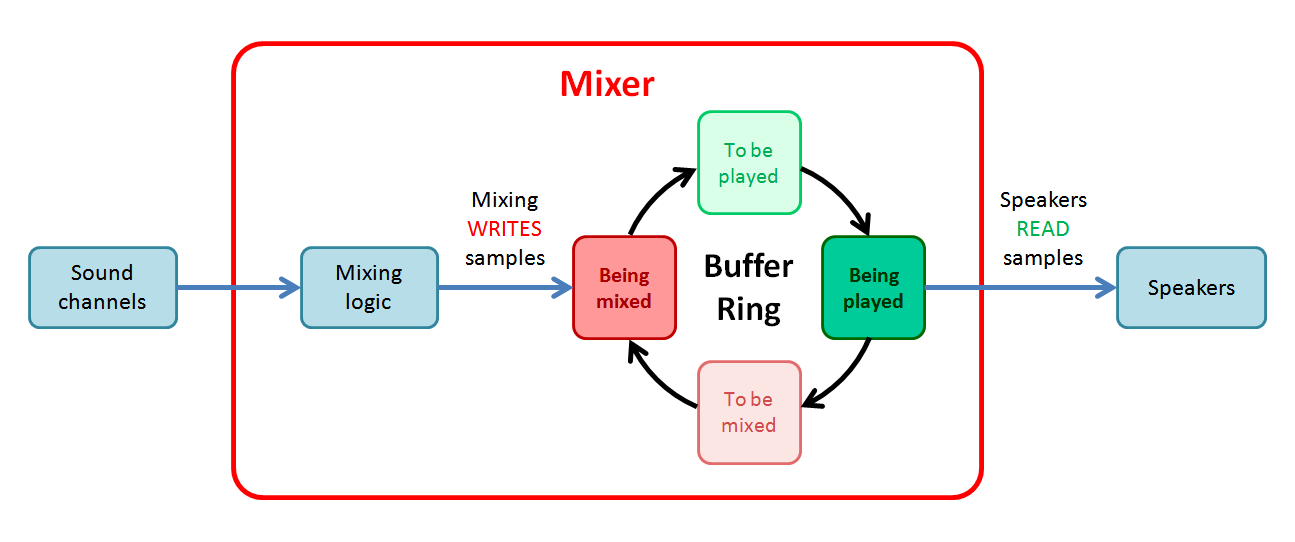
SPUは、「Stop Selected Channel」コマンドで説明したのと同じ処理を適用しますが、アクションはすべてのチャネルに適用されます。

## 9オーディオ出力の生成

セクション5で説明したように、Vircon32のSPUは、60 Hzのコンソールのグローバルタイミングに従って、フレーム単位でオーディオを生成します。しかし、サウンドカードやスピーカーなどのサウンド再生デバイスでは、通常、各単一サンプルが、44100 Hzの再生レートで、正しいタイミングでスピーカーに個別に提供される必要があります。

サウンドの生成と出力のタイミングの違いを説明するために、各フレームで生成されるサンプルは、再生のために送信される個々のSPUサンプルに適した時間になるまで、1つまたは複数の「待機バッファ」に格納する必要があります。オーディオシステムの実装とその出力信号形式に応じて、これには多くの可能な戦略があります。

ソフトウェアオーディオ処理で一般的な可能な戦略の1つは、「バッファリング」と呼ばれます。これは、周期的に再生され、新しいサンプルで補充されるいくつかの出力バッファを有することからなる。これが正しく機能するためには、各バッファが1フレームのサンプルを格納する必要があることに注意してください。したがって、バッファのサイズは44100/60=735フレームになります。次の図は、4バッファリングを実装する方法の例を示しています:



### 9.1サウンドバッファとレイテンシ

実際の実装では、コンソールのグローバルな60 Hzのタイミングが不正確になる場合があります。そのため、バッファリングのような戦略では、次のことを保証する方法でバッファを使用する必要があります。

1. スピーカーのサウンドサンプルがなくなることはありません。
2. ミキサーには、新しいサウンドを生成するための空きバッファが常にあります。

一般的に、オーディオバッファ戦略は常にトレードオフに直面する必要があります。一方で、より多くのバッファを使用することで、タイミングがより不正確であっても、これらの2つの条件を満たすことができます。bufより多くのバッファを使用すると、再生キューのサイズが大きくなるため、より多くのバッファが使用されるほど、サウンドの遅延が大きくなります。

### 9.2出力オーディオ信号

実装のオーディオ信号フォーマットに応じて、出力信号は、ステレオ出力をその2つのチャネルに分離するか(すなわち、32ビットSPUサンプルを左右について2つの別個の16ビット・サンプル・シーケンスに分割する)、または各スピーカに対して明確に識別された個々のサンプルを有するジョイント・シーケンスにそれらを配置する必要がある。

次に、SPUは、オーディオ出力信号を介してその情報をスピーカに転送する。信号フォーマット/プロトコルがそれを必要とする場合、タイミングまたはシーケンス情報が追加されて、有効なオーディオサンプル送信が形成される。

実装では、オーディオおよびビデオ信号の通信形式と物理コネクタを自由に選択できます。これらは個別に送信することも、現在のほとんどのディスプレイで一般的であるように、両方を同じディスプレイに送信するジョイントコネクタを使用することもできます。

## 10制御信号に対する応答

コンソール内のすべてのコンポーネントと同様に、制御信号がトリガーされるたびに、SPUはそれを受信し、そのイベントを処理するための応答を生成する。制御信号のそれぞれに対して、SPUは以下のアクションを実行することによって応答する。

#### リセット信号:

* オーディオの再生がまだ進行中の場合は、停止します。
* すべてのSPU内部変数は、初期値に設定されます。これには、すべてのサウンドおよびすべてのチャネルの設定変数が含まれます。
* コントロールポートの書き込み動作で説明されているように、内部変数の変更に関連付けられた追加の効果はすぐに適用されます。
* 出力バッファ内のすべてのサンプルは0に設定されます。

#### フレーム信号:

* SPUは、セクション5で説明したように、1つの新しいフレームの出力を生成する。

#### サイクル信号:

* SPUは、特定の実装詳細がそれを必要としない限り、このシグナルに反応する必要はない。

制御信号に反応することに加えて、SPUは、コンソールレベルのイベントに応答して以下の処理を実行する必要もある。

#### 新しいカートリッジが接続されたとき:

* SPUは、BIOSおよびカートリッジ(存在する場合)に含まれるサウンドを見つける。
* SPUは、セクション3で説明した接続プロセスを実行して、各サウンドにサウンドスロットを割り当て、それらのサンプル情報にアクセスする。
* 変数「Selected sound」の上限は、最後に使用されたサウンドスロットIDに調整されます。
* 使用されるすべてのサウンドスロットについて、変数「Sound length」の値は、割り当てられたサウンドのサンプル数に設定されます。
* 使用されるすべてのサウンドスロットについて、変数「Sound loop start」および「Sound loop end」の上限は、割り当てられたサウンドのサンプル数-1に設定されます。

(パート5の終わり)