32ビット仮想コンソール

システム仕様

パート6:コントローラチップ

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6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6では、コンソールの3つのコントローラチップが定義されています。このドキュメントでは、それぞれの動作、制御ポート、内部変数、および動作の一般的なプロセスについて説明します。

[1 About controller chips 3](#_30j0zll)

[2 Input controller 4](#_1fob9te)

[3 Cartridge controller 12](#_3znysh7)

[4 Memory card controller 18](#_2et92p0)

## 1コントローラチップについて

Vircon32コンソールは、取り外し可能ないくつかの外部要素(ゲームパッド、カートリッジ、メモリカード)と相互作用する必要があり、いつでも存在または不在にすることができます。このため、内部コンソールコンポーネントからこれらの取り外し可能な要素に直接接続することはできません。代わりに、通信のプロキシとして機能する「コントローラ」チップが必要です。

取り外し可能な要素は、それらがいつ存在するかをコンソールが判断できるようにする、ある種の論理的または物理的メカニズムを使用してコンソールに接続されます。

コントローラチップの基本機能は、CPUがこれらの可能な外部要素と通信するための安全な方法を提供することであり、それを可能にすることによって:

1. 指定された外部要素がいつ存在するかを判別します。
2. 存在しない要素にアクセスしようとしたときにエラー応答を受け取ります。
3. 要素が存在する場合に、メモリまたは状態にアクセスします。

入力コントローラ、カートリッジコントローラ、メモリカードコントローラの3つのコントローラチップがあります。次のセクションでは、これらのチップの1つについて詳しく説明します。この3つのチップを1つの文書にまとめるために、特定のチップの各セクションは、前のセクションとは異なり、この文書ではサブセクションとして作成されます。

## 2入力コントローラ

入力コントローラーは、コンソールの4つのゲームパッドポートを制御するチップです。すべてのVircon32ゲームパッドは同一ですが、ゲームパッドポートには番号が付けられています。例えば、1つのゲームパッドのみが接続されている場合でも、2番目のポートに接続されている場合は、ゲームパッド2として扱われます。これにより、合計16の組み合わせ(24個)をクリックします。

### 2.1外部接続

入力コントローラはコンソールを構成するチップの1つにすぎないため、単独で動作することはできません。この図は、他のコンポーネントとの通信をすべて示しています。前述したように、ゲームパッドへの4つの接続には番号が付けられており、交換できません。

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。

#### 2.1.1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、入力コントローラはリセット、新しいフレーム、新しいサイクルの信号を受信します。これらの信号に対する応答については、このドキュメントのセクション2.6に詳述されています。

#### 2.1.2制御バス

入力コントローラは、デバイスID=4のスレーブ・デバイスとしてコントロール・バスに接続されています。これにより、バス・マスター(CPU)は、入力コントローラによって公開されたコントロール・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。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。

#### 2.1.3ゲームパッド

入力コントローラは、4つの番号付きゲームパッドポートを管理します。各ポートは、CPUによって照会され、現在ゲームパッドが接続されているかどうかを判断できます。ポートにゲームパッドが接続されている場合、入力コントローラは現在の状態を読み取ることができます。

### 2.2作業コンセプト

入力コントローラの機能またはそれらに影響を与える内部変数を説明する前に、このチップが構築されている基本的な概念を提示しなければならない。

#### 2.2.1選択されたゲームパッド

使用可能なすべてのゲームパッドポートは、0から3までの使用可能なゲームパッドIDの範囲を形成します。制御状態を提供するとき、または内部変数を適用するときに使用するゲームパッドを決定するために、入力コントローラは常にそれらのIDの1つを「選択された」とみなします。デフォルトで選択されるゲームパッドは、ゲームパッドID=0に対応する最初のものです。

#### 2.2.2ゲームパッドのコントロール

Vircon32ゲームパッドはすべて同じです。各ゲームパッドには、11のコントロール(4つの方向と7つのボタン)があります。これらのリストは、それらをすべて列挙しています:

| 方向:  左、右、上、下 | ボタン:  A、B、X、Y、L、R、開始 |
| --- | --- |

これらのコントロールはすべてデジタルであり、ゲームパッドはその状態を1ビットで表します。押されたときは1(True)、押されていないときは0(False)です。

#### 2.2.3ゲームパッド状態フォーマット

すべてのゲームパッドコントロールは1ビットで表すことができますが、入力コントローラは実際には整数形式を使用してそれらの状態を保存します。これは、入力コントローラがゲームパッドから読み取られたコントロール状態を保存するだけでなく、これらの値が時間の経過とともに処理され、各コントロールの現在の状態だけでなく、より多くの情報を提供するためです。

入力コントローラは、コントロールが現在の状態にあった時間(フレーム単位)を報告するために、過去の状態を追跡します。これは整数の符号と値としてエンコードされ、押された場合は正の符号、押されていない場合は負の符号になります。たとえば、ボタンAの現在の状態が20の場合、その解釈は次のようになります。

多くのゲームでは、ボタンが押された瞬間に1回だけアクションを実行する必要があります。この追加情報を提供することによって、入力コントローラは、押されたボタンが押されたばかりであるかどうかを即座に知ることを可能にします。これを決定するために以前の状態を保存する必要はありません。

制御状態値は、符号を常に決定できるように、ゼロにならないことが保証されています。これを保証するために、次の2つのルールが実装されています。

1. 現在のフレームで状態を変更したばかりのコントロールは、値+/-1に設定され、状態が維持されている場合は、そこからフレームカウントが開始されます。
2. 特定のゲームパッドポートにゲームパッドが接続されていない場合、入力コントローラはすべてのコントロール状態を-3600の下限に設定します。この値は、コントロールが長時間押されていないと解釈される可能性があります。

### 2.3ゲームパッドの読み取りプロセス

ゲームパッド読み取りプロセスは、ゲームパッドの現在の状態を読み取り、その結果として内部変数を更新するために入力コントローラによって使用されるプロセスです。CPUは、現在のゲームパッド入力を決定するために、これらの内部変数の読み取りを要求できます。

ほとんどのコンソールプロセスと同様に、ゲームパッドを読み取るタイミングはフレームに基づいています。ゲームパッドの読み取りプロセスは、各フレームの最初にトリガーされ、実行ごとに4つのゲームパッドポートのそれぞれに対してループを実行します。次に、各ゲームパッドポートに対して、各ゲームパッドの11のコントロールのそれぞれに対して次の処理を実行し、それぞれの更新された状態(それぞれの内部変数に格納されている)を決定します。

範囲の制限により、コントロール状態は3600を超えてインクリメントすることも、-3600を超えてデクリメントすることもできないため、それに応じてクランプされることに注意してください。このアルゴリズムによるコントロール状態の更新に加えて、入力コントローラは、各ゲームパッドが現在接続されているかどうかを表すブール変数も更新します。

#### 2.3.1方向の一貫性

Vircon32では、ゲームパッドが同じ軸の反対方向に同時に押されることを物理的に防ぐ必要があります。これは、上+下、または左+右を押すことができないことを意味します。ただし、実装によっては、すべての可能なゲームパッドに対してこれを保証することができない場合があります。

このゲームパッドの機能が保証されない場合、入力コントローラは、プログラムが方向性パッドで一貫性のない方向に遭遇しないことを保証するために、方向を押すイベントまたは状態をフィルタリングする必要があります。これを行う一般的な方法は、方向コントロールが押されると、反対の方向が自動的に解除されると考えることです。つまり、各軸で最後に押された方向が優先されます。

#### 2.3.2 Gamepad接続イベント

ゲームパッドが接続または取り外された場合、入力コントローラは、このイベントをオンザフライで報告するための信号を提供しません。コンソールは、ゲームパッドの状態が次のフレームで読み取られた後、指定されたゲームパッドポートの接続された変数に変更があった場合にのみ、このことを認識します。実装は、必要に応じてこれらのイベントを使用して内部的に動作できます。

### 2.4内部変数

入力コントローラは、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ています。これらの変数はそれぞれ32ビット値として格納され、仕様のパート2で説明されている同じデータ形式(整数、ブールなど)を使用してすべて解釈されます。ここでは、すべての内部変数をセクションに分けてリストし、詳細を説明します。

#### 2.4.1グローバル変数

| 選択されたゲームパッド | **初期値:**0 |
| --- | --- |
| **形式:**整数 | **有効範囲:**0から3 |

この値は、現在選択されているゲームパッドの数値IDです。選択されたゲームパッドは、ゲームパッドに関連するすべての操作で使用されます。

4つのゲームパッドIDは、対応するゲームパッドが現在接続されていない場合でも、常に選択可能であることに注意してください。

#### 2.4.2各ゲームパッドの状態

ここにリストされている変数は特別なものです。入力コントローラは、既存のゲームパッドポートごとにこれらの変数のコピーを保存します。これは、4つのコピーがあることを意味します。現在接続されているゲームパッドとは無関係に、これらの4つのすべてが常に存在します。

同時に、これらの変数の各セットは、単一のゲームパッドの現在の入力を記述します。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。選択されたゲームパッドが変更されると、これらのポートはすべて、正しいゲームパッドの変数のコピーにリダイレクトされます。

| ゲームパッド接続済み | **初期値:**(\*)は |
| --- | --- |
| **形式:**ブール値 | **有効範囲:**真/偽 |

この値は、関連付けられたゲームパッドが接続されているかどうかを示します。

(\*)その初期値は、コンソールの起動時に関連するゲームパッドポートにゲームパッドが存在するかどうかによって決定されます。

| ゲームパッド{*支配*}をクリックし | **初期値:**-3600人 |
| --- | --- |
| **形式:**整数 | **有効範囲:**-3600～3600(0を除く) |

これらは11の変数で、セクション2.2.2にリストされている11のゲームパッドコントロールのそれぞれに対応しています。これらのそれぞれは、関連するゲームパッドのコントロールの現在の状態を表します。この状態は、ブール状態だけでなく、すでに処理された状態です。したがって、セクション2.2.3で説明されているように、{sign,value}として解釈されます。

### 2.5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入力コントローラ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 |
| 400時間 | 00時間 | 選択されたゲームパッド | 読み取り/書き込み |
| 401時間 | 01時間 | ゲームパッド接続 | 読み取り専用 |
| 402時間 | 02時間 | ゲームパッド左 | 読み取り専用 |
| 403時間 | 03時間 | ゲームパッド右 | 読み取り専用 |
| 404時間 | 04時間 | ゲームパッドアップ | 読み取り専用 |
| 405時間 | 05時間 | ゲームパッドダウン | 読み取り専用 |
| 406時間 | 06時間 | ゲームパッドボタン開始 | 読み取り専用 |
| 407時間 | 07時間 | ゲームパッドボタンA | 読み取り専用 |
| 408時間 | 08時間 | ゲームパッドボタンB | 読み取り専用 |
| 409時間 | 09時間 | ゲームパッドボタンX | 読み取り専用 |
| 40 Ah弾 | 0 Ah(ゼロ時) | ゲームパッドボタンY | 読み取り専用 |
| 40 Bh弾 | 0時 | ゲームパッドボタンL | 読み取り専用 |
| 40 Ch(英語の可能性あり | 0 Ch(0チャネル) | ゲームパッドボタンR | 読み取り専用 |

#### 2.5.1ポートの読み取り/書き込み要求に対する動作

入力コントローラの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。特定のポートへの読み取り/書き込み要求によってトリガされる効果は、値の読み取りまたは書き込みとは異なる場合があります。このセクションでは、各入力コントローラ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、制御バス通信の一部として要求に対して成功/失敗応答を提供する必要があることに注意してください。この応答は、特に指定がない限り、常に成功と見なされます。提供された応答が失敗の場合、入力コントローラ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。

選択されたゲームパッドポート

オン読み取る要求:

入力コントローラは、内部変数「Selected gamepad」の現在の値を提供します。

オン書く要求:

入力コントローラは、受信した値が有効なゲームパッド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、入力コントローラは、内部変数「Selected gamepad」を受信した値で上書きします。その後、すべてのゲームパッド設定ポートを、新しく選択されたゲームパッドの変数セットを指すようにリダイレクトします。

Gamepad接続ポート

オン読み取る要求:

入力コントローラは、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「Gamepad connected」の現在の値を提供します。

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。

ゲームパッド{*支配*}ポート

これと同じ動作は、セクション2.2.2にリストされている11のゲームパッドコントロールに対応する11のポート(アドレス02h～0 Ch)にも適用されます。

オン読み取る要求:

入力コントローラは、現在選択されているゲームパッドIDに関連付けられた対応する内部変数「Gamepad{control}」の現在の値を提供する。

オン書く要求:

これら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。

### 2.6操縦信号への応答

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、入力コントローラはそれを受信し、そのイベントを処理するための応答を生成します。各制御信号に対して、入力コントローラは次のアクションを実行することによって応答します。

#### リセット信号:

* すべての入力コントローラ内部変数は、初期値に設定されます。これには、すべてのゲームパッドの設定変数が含まれます。
* ゲームパッドポートごとに、入力コントローラは、接続されているゲームパッドがあるかどうかをチェックし、対応する「Gamepad connected」変数を更新します。
* 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。

#### フレーム信号:

* 入力コントローラは、セクション2.3で説明されているように、ゲームパッド読み取りプロセスをトリガする。

#### サイクル信号:

* 入力コントローラは、特定の実装詳細がそれを必要としない限り、この信号に反応する必要はない。

## 3カートリッジコントローラ

カートリッジコントローラは、コンソールのカートリッジスロットの制御を担当するチップです。ゲームパッドとは異なり、カートリッジは1つしか存在できません。ただし、各カートリッジの内容は異なります。3つのカートリッジROM(プログラム、ビデオ、およびオーディオ)のうち、最初のものだけが常に存在し、すべてのカートリッジには毎回異なる数の要素が含まれる可能性があります。

### 3.1外部接続

カートリッジコントローラは、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、他のコンポーネントとの通信をすべて示しています。前述のように、接続されているカートリッジ(存在する場合)は、毎回異なるものになる可能性があります。

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。

#### 3.1.1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、カートリッジコントローラは、リセット、新しいフレーム、および新しいサイクル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3.5に詳述されています。

#### 3.1.2制御バス

カートリッジ・コントローラは、デバイスID=5のスレーブ・デバイスとしてコントロール・バスに接続されています。これにより、バス・マスター(CPU)は、カートリッジ・コントローラによって公開されるコントロール・ポートでの読取りまたは書込み操作を要求できます。カートリッジ・コントローラ・ポートおよびそのプロパティのリストについては、後の項で詳しく説明します。

#### 3.1.3メモリバス

カートリッジコントローラは、デバイスID=2のスレーブデバイスとしてメモリバスに接続されています。これにより、バスマスタ(CPU)は、カートリッジによって公開されるメモリアドレスに対して読み取りまたは書き込み操作を要求できます。カートリッジメモリアドレスの範囲とプロパティについては、後のセクションで説明します。

#### 3.1.4カートリッジ

カートリッジコントローラは、カートリッジスロットを管理します。このスロットは、現在カートリッジが接続されているかどうかを判断するためにCPUによって照会されます。カートリッジが存在する場合、カートリッジコントローラはその内容を読み取ることができます。

カートリッジスロットには、コンソールの電源が入っている限りスロットをふさぐ安全機構があります。このため、コンソールの操作中にカートリッジの接続や内容が変更されることはありません。

#### 3.1.5 GPUおよびSPU

GPUとSPUは、カートリッジのビデオROMとオーディオROMにそれぞれアクセスする必要があります。これらのROMは存在しない可能性があるため、これら2つの接続はカートリッジコントローラを介して行う必要があります。これは、各チップがターゲットROMが存在することと、それに含まれる要素の数を判断するために必要です。

これらの初期情報がわかると、GPUとSPUからの接続は単純なパススルーとして考えられ、各チップはこれらのコンテンツを自由に読み取ることができる。それでも、このアクセスを設定および管理するための実際のメカニズムは、実装によって定義される必要があります。

### 3.2カートリッジメモリ

コンソール自体には、BIOSルーチン以外のCPU用プログラムは含まれていません。そのため、プログラムROMに接続してカートリッジに格納されているメモリワードを読み取る必要があります。プログラムROMは、32ビットワードのシーケンスを含む読み取り専用のメモリ領域です。

カートリッジコントローラはメモリバスに接続され、接続されたカートリッジがそのプログラムROM(常に存在する)の内容をCPUに公開できるようにします。カートリッジコントローラはメモリバス内でデバイスID=2を使用するため、Nワードを含むカートリッジプログラムROMの場合、カートリッジメモリのアドレス範囲は次のようになります。

0からN-1までの内部アドレス🡪

外部アドレス🡪20000000hから(20000000h+N-1)。

カートリッジプログラムROMのサイズは、カートリッジごとに異なります。1からカートリッジプログラムROMのサイズ制限である128 x 1024 x 1024までの任意の数のワードを含めることができます。

#### 3.2.1プログラムROMへの接続

Nワードのカートリッジを接続した場合、カートリッジコントローラの内部メモリアドレスは、0から順にN個ずつ割り当てられ、(128 x 1024 x 1024-1)までは使用されず、アクセスできなくなります。

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の電源がオフのときにのみ挿入できます。したがって、実装では、カートリッジコントローラがこの接続を確立するために必要な手順を、次にコンソールの電源がオンになるまで遅らせることが安全です。

プログラムROMへの接続を確立し、それらのワー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、接続時に事前にプログラムROM全体を読み取ることができます。別のオプションは、カートリッジメモリへのポインタを保持し、カートリッジコントローラがオンザフライでワードを読み取るようにすることです。

#### 3.2.2メモリの読み取り/書き込み要求に対する動作

現在使用されている範囲内のすべてのメモリアドレスは自由に読み取ることができるため、その範囲の読み取り要求は常に格納されたワード値を提供し、成功して応答されます。現在使用されている範囲外を読み取ろうとすると、応答は失敗し、ハードウェアエラーが発生します。

カートリッジプログラムROMは常に読み取り専用メモリであるため、書き込み操作は失敗して応答され、ハードウェアエラーが発生します。

### 3.3内部変数

カートリッジコントローラは、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ています。これらの変数はそれぞれ32ビット値として格納され、仕様のパート2で説明されている同じデータ形式(整数、ブールなど)を使用してすべて解釈されます。ここでは、すべての内部変数をリストし、詳細を示します。

| カートリッジが接続されました | **初期値:**(\*)は |
| --- | --- |
| **形式:**ブール値 | **有効範囲:**真/偽 |

この値は、カートリッジがコンソールに接続されているかどうかを示します。

(\*)初期値は、コンソール起動時のカートリッジスロット内のカートリッジの有無によって決定されます。この値は、コンソール操作中に変更することはできません。

| プログラムROMサイズ | **初期値:**(\*)は |
| --- | --- |
| **形式:**整数 | **有効範囲:**1～134217728(=128\*1024\*1024) |

現在接続されているカートリッジのプログラムROMのサイズを32ビットワードで表します。カートリッジが存在しない場合、この値は0です。

(\*)その初期値は、コンソールの起動時にカートリッジスロットに存在するカートリッジ(存在する場合)によって決定されます。この値は、コンソールの動作中には変更できません。

| テクスチャの数 | **初期値:**(\*)は |
| --- | --- |
| **形式:**整数 | **有効範囲:**0～256 |

現在接続されているカートリッジのビデオROMに含まれているテクスチャの数を表します。カートリッジが存在しない場合、またはカートリッジにビデオROMが含まれていない場合、この値は0です。

(\*)その初期値は、コンソールの起動時にカートリッジスロットに存在するカートリッジ(存在する場合)によって決定されます。この値は、コンソールの動作中には変更できません。

| サウンドの数 | **初期値:**(\*)は |
| --- | --- |
| **形式:**整数 | **有効範囲:**0～1024 |

現在接続されているカートリッジのオーディオROMに含まれているサウンドの数を表します。カートリッジが存在しない場合、またはカートリッジにオーディオROMが含まれていない場合、この値は0です。

(\*)その初期値は、コンソールの起動時にカートリッジスロットに存在するカートリッジ(存在する場合)によって決定されます。この値は、コンソールの動作中には変更できません。

### 3.4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カートリッジ・コントローラ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説明します。公開されるすべてのポートとその基本的な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 |
| 500時間 | 00時間 | カートリッジが接続されました | 読み取り専用 |
| 501時間 | 01時間 | プログラムROMサイズ | 読み取り専用 |
| 502時間 | 02時間 | テクスチャの数(Number Of Textures) | 読み取り専用 |
| 503時間 | 03時間 | サウンドの数 | 読み取り専用 |

#### 3.4.1ポートの読み取り/書き込み要求に対する動作

他のチップとは異なり、カートリッジコントローラポートは実際には読み取り専用レジスタとしてモデル化できます。これらのポートを読み取ると、このセクションで説明するように、関連する値のみが取得されます。

実行されるアクションに加えて、制御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。このレスポンスは常に、読取りリクエストでは成功、書込みリクエストでは失敗になります。この最後の場合、カートリッジ・コントローラ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。

カートリッジ接続ポート

オン読み取る要求:

カートリッジコントローラは、内部変数「Cartridge connected」の現在の値を提供します。

プログラムROMサイズポート

オン読み取る要求:

カートリッジコントローラは、内部変数「プログラムROMサイズ」の現在の値を提供します。

[Number Of Textures]ポート

オン読み取る要求:

カートリッジコントローラは、内部変数「テクスチャの数」の現在の値を提供します。

[Number Of Sounds]ポート

オン読み取る要求:

カートリッジコントローラは、内部変数「Number of sounds」の現在の値を提供します。

### 3.5操縦信号への応答

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、カートリッジコントローラはそれを受信し、そのイベントを処理するための応答を生成します。各制御信号に対して、カートリッジコントローラは次のアクションを実行することによって応答します:

#### リセット信号:

* カートリッジがない状態でコンソールの電源が入っている場合(つまり、カートリッジイベントがまだ処理されていない場合)、カートリッジコントローラはカートリッジが取り外されたときと同じ処理を実行します。

#### フレームおよびサイクルシグナル:

* カートリッジコントローラは、特定の実装詳細がそれを必要としない限り、これらの信号に反応する必要はない。

カートリッジコントローラは、制御信号に反応するだけでなく、コンソールレベルのイベントに応じて次の処理を実行する必要があります。

#### カートリッジを取り外したら、次の手順に従います。

* カートリッジコントローラの「Cartridge connected」変数はFalseになります。
* 内部変数「プログラムROMサイズ」、「テクスチャ数」、「サウンド数」はすべて0に設定されます。
* すべてのカートリッジコントローラのメモリアドレスが割り当てられません。

#### 新しいカートリッジが接続されたとき:

* カートリッジコントローラは、「Cartridge connected」変数をTrueに設定します。
* カートリッジコントローラは、セクション3.2.1で説明した接続プロセスを実行して、メモリアドレスをそのプログラムROMに割り当て、それに含まれるワードにアクセスする。
* 内部変数「プログラムROMサイズ」は、接続されたカートリッジのプログラムROMに含まれるワード数に設定されます。
* 接続されたカートリッジにビデオROMが含まれている場合、カートリッジコントローラは内部変数「テクスチャの数」を、含まれているテクスチャの数に設定します。
* 接続されたカートリッジにオーディオROMが含まれている場合、カートリッジコントローラは内部変数「Number of sounds」を、含まれているサウンドの数に設定します。

## 4メモリカードコントローラ

メモリカードコントローラは、本体のメモリカードスロットを制御するためのチップです。カートリッジと同じように、カードは1つしか存在できません。すべてのカードのメモリは同じサイズであり、保存されている内容のみが異なります。ただし、カートリッジとは異なり、メモリカードはいつでも接続および取り外しができます。

### 4.1外部接続

メモリカードコントローラ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、他のコンポーネントとの通信をすべて示しています。前述のように、接続されているメモリカード(存在する場合)は、毎回異なるものになる可能性があります。

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。

#### 4.1.1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、メモリカードコントローラはリセット、新しいフレーム、新しいサイクル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4.5に詳述されています。

#### 4.1.2制御バス

メモリ・カード・コントローラは、デバイスID=3のスレーブ・デバイスとしてコントロール・バスに接続されています。これにより、バス・マスタ(CPU)は、メモリ・カード・コントローラによって公開されるコントロール・ポートでの読み取りまたは書き込み操作を要求できます。メモリ・カード・コントローラ・ポートのリストとそのプロパティについては、後のセクションで説明します。

#### 4.1.3メモリバス

メモリカードコントローラは、デバイスID=3のスレーブデバイスとしてメモリバスに接続されています。これにより、バスマスタ(CPU)は、メモリカードによって公開されるメモリアドレスに対して読み取りまたは書き込み操作を要求できます。カードメモリアドレスの範囲とプロパティについては、後のセクションで説明します。

#### 4.1.4メモリカード

メモリカードコントローラは、メモリカードスロットを管理します。このスロットは、メモリカードが現在接続されているかどうかを判断するためにCPUによって照会されます。カードが存在する場合、メモリカードコントローラはその内容を読み取り、変更できます。

メモリカードスロットには、カートリッジスロットのようなブロッキングメカニズムはありません。このため、プログラムは、メモリカードの読み取りまたは書き込みを開始する前に、毎回メモリカードがあるかどうかを確認する必要があります。メモリカードがないときにカードにアクセスしようとすると、ハードウェアエラーが発生し、プログラムが停止します。

### 4.2カードメモリ

コンソール自体には、プログラムが使用できる永続的なストレージは含まれていません。セッション間でデータを保存および取得するには、RAMメモリに接続してカード内の単語を読み書きする必要があります。RAMメモリは、32ビットワードのシーケンスを含む読み取りおよび書き込みメモリ領域です。

メモリカードコントローラはメモリバスに接続され、接続されたカードがRAMの内容をCPUに公開できるようにします。メモリカードコントローラはメモリバス内でデバイスID=3を使用します。メモリカードRAMには正確に(256 x 1024)ワードが含まれるため、カードメモリのアドレス範囲は次のようになります。

0～3 FFFhの内部アドレス🡪(=256 x 1024-1)

外部アドレス🡪30000000hから30003 FFFh

#### 4.2.1カードRAMへの接続

メモリカードが接続されている場合、メモリカードコントローラはすべての内部メモリアドレス(256 x 1024)をカードのRAMワードに0から順に割り当てます。

この接続プロセスは、新しいメモリカードが挿入されるたびに発生します。これはいつでも発生する可能性があるため、メモリカードコントローラはその時点で接続を実行する必要があります。コンソールがオフの間にカードが接続された場合、実装では、次のコンソール電源がオンになるまでこの接続プロセスを遅延させることができます。

カードRAMへの接続を確立する方法と、そのワードにアクセスする方法を決定するのは、実装次第です。カードアクセスはサイクルごとに発生する可能性があるため、実装が十分に高速でない場合に可能なオプションは、システムRAM内のコピーを使用して作業し、特定の瞬間に(たとえば、新しいフレーム信号を使用して)実際のカードに変更を書き込むことです。

#### 4.2.2メモリの読み取り/書き込み要求に対する動作

指定された範囲内のメモリアドレスは自由に読み書きできます。したがって、その範囲に対する要求は常に格納されたワード値を提供し、成功して応答されます。現在使用されている範囲を超えて読み書きしようとすると、失敗して応答され、ハードウェアエラーが発生します。

#### 4.2.3カード接続イベント

カードが接続または取り外された場合、メモリカードコントローラは、このイベントをオンザフライで報告するための信号を提供しません。それでも、コンソールは、カードが接続または取り外された後、「カード接続」変数がその値を変更したときに、このことを知ることができる。それでも、実装は、必要に応じてこれらのイベントを使用して内部的に動作することができる。

### 4.3内部変数

メモリカードコントローラには、内部状態を格納する単一の変数があります。この変数は32ビット値として格納され、仕様のパート2で説明されているのと同じデータ形式(整数、ブールなど)を使用して解釈されます。ここでは、この内部変数のリストと詳細を示します。

| カードが接続されました | **初期値:**(\*)は |
| --- | --- |
| **形式:**ブール値 | **有効範囲:**真/偽 |

この値は、メモリカードが現在コンソールに接続されているかどうかを示します。

(\*)その初期値は、コンソール起動時のメモリカードスロット内のカードの有無によって決定されます。

### 4.4制御ポート

この項では、スレーブ・デバイスとしてのCPU制御バスへの接続を介してメモリー・コントローラによって公開される制御ポートのセットについて説明します。公開される単一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 |
| 600時間 | 00時間 | カード接続 | 読み取り専用 |

#### 4.4.1ポートの読み取り/書き込み要求に対する動作

他のチップとは異なり、メモリカードコントローラポートは実際には読み取り専用レジスタとしてモデル化できます。このセクションで詳細に説明するように、このポートを読み取ると、関連する値のみが取得されます。

カード接続ポート

オン読み取る要求:

メモリカードコントローラは、内部変数「Card connected」の現在の値を提供し、成功応答が制御バスに提供される。

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。

### 4.5操縦信号への応答

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、メモリカードコントローラはそれを受信し、そのイベントを処理するための応答を生成します。各制御信号に対して、メモリカードコントローラは次のアクションを実行することによって応答します。

#### リセット信号:

* カードがない状態でコンソールの電源がオンになっている場合(つまり、カードイベントがまだ処理されていない場合)、メモリカードコントローラは、メモリカードが取り外されたときと同じ処理を実行します。

#### フレームおよびサイクルシグナル:

* メモリカードコントローラは、特定の実装詳細がそれを必要としない限り、これらの信号に反応する必要はない。

メモリカードコントローラは、制御信号に反応することに加えて、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がある。

#### メモリーカードを接続した場合

* メモリカードコントローラは、「Card connected」変数をTrueに設定します。
* メモリカードコントローラは、カードRAMにメモリアドレスを割り当て、それに含まれるワードにアクセスするために、セクション4.2.1で説明した接続プロセスを実行する。

#### メモリカードを取り外すと、次のようになります。

* メモリカードコントローラは、「Card connected」変数をFalseに設定します。
* メモリカードコントローラのすべてのメモリアドレスが割り当て解除されます。

(パート6の終わり)