32ビット仮想コンソール

システム仕様

第4回:グラフィックスチップ(GPU)

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4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4では、コンソールのグラフィックスチップ(GPU)が定義されています。このドキュメントでは、このチップの動作、制御ポート、提供される描画コマンド、およびビデオ出力の生成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 Drawing functions 8](#_2et92p0)

[5 Graphic effects 10](#_tyjcwt)

[6 Color blending 11](#_3dy6vkm)

[7 GPU performance 13](#_1t3h5sf)

[8 Internal variables 14](#_4d34og8)

[9 Control ports 18](#_2s8eyo1)

[10 Execution of commands 24](#_17dp8vu)

[11 Generation of video output 28](#_3rdcrjn)

[12 Responses to control signals 28](#_26in1rg)

## 1はじめに

GPUは、Vircon32コンソールのビデオチップです。画面上に表示されるすべてのものを描画します。これを行うには、BIOSとカートリッジの両方に含まれるテクスチャを使用し、基本的なグラフィカル効果を適用します。

このグラフィックスチップは非常にシンプルです。GPUが画面上で描画できるアクションは、次の2種類のみです。

* 一定の色で画面をクリアすることができます。
* 使用可能なテクスチャの1つから領域を画面上に描画できます。

どちらも、プロセッサから要求された描画コマンドとして実行されます。これ以外にも、これらのアクションに適用できるグラフィカル効果がいくつかあります。これらについては、後のセクションで説明します。

## 外部接続×2

GPU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、GPU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、GPUには、含まれるイメージをテクスチャとして使用できるように、使用可能なすべてのビデオROMへの接続があります。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、GPU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12に詳述されています。

### 2.2制御バス

GPUは、デバイスID=2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、GPU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。GPU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

常に存在するBIOSには、1つのイメージだけを含むビデオROMが含まれています。GPUは、ID=-1のテクスチャスロットを介してそのイメージ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、GPUは、IDが0～255のテクスチャスロットを介して、カートリッジ内に存在する任意のイメージ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、それぞれに異なる数のイメージ(0～256)を含めることができます。

### 2.5表示

フレームごとに、描画が完了すると、GPUは結果をディスプレイに送信して表示できるようにする必要があります。これはビデオ出力接続を介して行われます。セクション11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

GPUのグラフィック機能またはグラフィック機能に影響を与える内部変数を説明する前に、GPUの基礎となる一連の基本概念を示す必要があります。

### 3.1ピクセルとカラー

GPUによって処理されるビデオ情報の最小単位はピクセルであり、これは画面またはテクスチャのいずれかの単一の個別の位置を表します。すべてのピクセルは、仕様のパート2ですでに文書化されているように、単一の32ビットRGBAカラーとして表されます。

### 3.2図面バッファ

描画バッファは、画面と同じサイズ(640 x 360ピクセル)のピクセルの矩形2D配列です。各ピクセルは、左上隅からの位置((0,0)から(639,359))で識別されます。

GPUは、このバッファを画面自体のプロキシとして使用します。描画バッファの内容は、GPUグラフィック操作のたびに変更され続けます。その後、新しいフレームごとに、ビデオ出力信号が描画バッファの現在の内容で更新され、最後のフレームからの変更が表示されます。全体として、画面バッファを介したGPUの操作は次のようになります。

ピクセルのアルファコンポーネントは、描画バッファにカラーをブレンドするプロセスのために、描画操作中にのみ使用される必要があることに注意してください。この後、描画バッファ自体は、そのピクセルのアルファコンポーネントを格納する必要はありません。描画フォーマットのピクセルフォーマットが32ビットを使用するか、アルファを省略した24ビットのみを使用するかを決定するのは、実装次第です。いずれの場合も、描画バッファ内のピクセルは常に完全な不透明度(アルファ=255)であると想定できます。

### 3.3 GPUテクスチャ

GPUテクスチャは、1024 x 1024ピクセルの固定サイズを持つ正方形の2Dピクセル配列です。テクスチャ内のピクセルの座標は、左上隅のピクセル(0,0)から始まります。画面をクリアする以外に、すべてのGPU描画操作では、描画にテクスチャを使用する必要があります。

GPUは、番号付きのテクスチャスロットの配列を使用して、使用可能なテクスチャを識別し、アクセスします。カートリッジで使用できるスロットは256個あり(テクスチャIDは0～255)、BIOSテクスチャ用にID=-1の追加スロットがあります。

### 3.4ビデオROMへの接続

GPU自体にはテクスチャ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イメージを、それぞれのビデオROMに接続して読み取る必要があります。ビデオROMは、一連のイメージを含む読み取り専用のメモリ領域です。これらの各イメージは、1 x 1ピクセルからGPUテクスチャサイズまで、任意の幅と高さを持つことができます。

N個のイメージ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個のスロットが各イメージに順番に割り当てられます。ID=255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イメージに割り当てられます。BIOSイメージは存在し、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、GPU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。

ビデオROMへの接続を確立し、それらのイメージを見つけて読み取る方法を決定するのは実装次第です。たとえば、接続時にビデオROM内のすべてのイメージを事前に読み取ることができます。もう1つのオプションは、イメージへのポインタを保持し、GPUがその場でピクセル値を読み取るようにすることです。

#### イメージを拡張する

前述したように、GPUテクスチャは固定サイズです。ただし、スロットはそれぞれ異なるサイズのイメージからピクセルを読み取ります。この問題を解決するために、各テクスチャスロットは、サイズを1024 x 1024に自動的に拡張し、残りのピクセルを空(つまり、4つのコンポーネントが0)と見なすことによって、割り当てられたイメージをGPUテクスチャとして使用できるように適応させます。拡張は右下で行われるため、すべてのピクセルの座標が保持されます。

この図は、ビデオROMに格納されているイメージが空のピクセルでどのように拡張されるかを示しています(わかりやすくするために灰色で示しています)。

繰り返しになりますが、イメージを読み取って拡張するためにテクスチャスロットによって使用される方法(ピクセルマッピング、内部コピー、ハードウェア接続など)も、実装次第です。

### 3.5テクスチャ領域

GPUテクスチャは画面よりも大きいため、GPUはテクスチャを直接操作しません。代わりに、各テクスチャ内に一連の区切り領域を定義して、GPU描画関数がテクスチャを操作できるようにすることができます。

各テクスチャに領域を定義すると、Vircon32プログラムでは、複数のイメージを1つのテクスチャの一部としてグループ化することで、テクスチャをより効率的に使用できるようになります。この方法の例を次の例に示します。

テクスチャ領域はテクスチャの長方形の部分で、次の図に示すように、3つのピクセルの座標によって定義されます。ピクセル1と2は領域の境界を定義し(両方とも領域に含まれます)、一般にホットスポットと呼ばれるピクセル3は、描画時に領域を画面に配置するための参照として使用されます。ホットスポットは領域の境界の外側に配置できます。

### 3.6選択されたエレメント

GPUは、グラフィックエレメント(テクスチャおよび領域)をセットに編成しますが、通常は、各セットの1つのエレメントのみを使用して動作します。

#### 選択されたテクスチャ

割り当てられたすべてのテクスチャスロットは、このイメージに示されているように、使用可能なテクスチャIDの連続した範囲を形成します。ただし、使用可能なテクスチャのうちアクティブになるのは常に1つだけです。描画時またはその他の機能の使用時に使用するテクスチャを決定するために、GPUは常にそれらのIDの1つを「選択済み」と見なします。デフォルトで選択されるテクスチャはBIOSテクスチャです。これは、常に存在することが保証されている唯一のテクスチャであるためです。

#### 選択された領域

割り当てられた各テクスチャスロットには、4096の設定可能な領域があり、領域ID 0～4095を使用してアクセスできます。これらはすべて、プログラムで値が定義されていない場合でも、常に存在し、使用可能であることに注意してください。テクスチャと同様に、GPUは常に領域IDの1つを「選択された」と見なし、GPU機能で使用される領域を決定します。ただし、このパラメータはグローバルであり、各テクスチャに対して異なる選択領域はありません。

## 4図形描画機能

導入セクションで説明したように、GPUが画面に描画できる方法は2つしかありません。画面をクリアする方法とテクスチャ領域を描画する方法です。このセクションでは、これらの機能がどのように実行されるかについて基本的な説明を行います。

ただし、グラフィック機能の実行方法の詳細については、コマンドの実行に関するセクションで後述します。また、特定の修正効果を適用することは可能ですが、この説明では基本的な修正されていないバージョンについてのみ説明します。

### 4.1画面のクリア

画面をクリアすると、描画バッファのすべてのピクセルに単色(クリアカラー)が適用されます。画面をクリアすることは最も簡単な機能です。テクスチャにアクセスせずに動作し、グラフィック効果を使用して修正することはできません。

### 4.2テクスチャ領域の描画

GPUは、描画バッファ上の現在使用可能なテクスチャ領域を描画できます。その領域を配置する場所を決定するために、ピクセル座標で指定された描画位置を使用します。領域は、そのホットスポットまたは参照ポイントがその描画位置に配置されるように描画されます。

例えば、描画位置が画面の中心(画素320,180)である場合、先に定義したサンプル領域の基準点は、以下の例のようにこれらの座標に一致するように作成されます。

## 5グラフィック効果

柔軟性を高め、より高度なグラフィック機能を実現するために、さまざまな種類の効果をGPU描画機能に適用できます。これらの効果を有効にすると、描画機能の動作方法が変更されるため、画面に表示される結果が変わります。

### 5.1スケーリング効果

テクスチャ領域を描画するときに、領域のX軸とY軸に沿ってスケーリング係数を適用するようにGPUに指示することもできます。その結果、出力矩形はそれに応じてスケーリングされ、ホットスポットの位置が保持されます。このエフェクトの例は次のようになります。

### 5.2回転エフェクト

テクスチャ領域を描画するときに使用できるもう1つのオプションの効果は、回転角度を適用することです。描画された領域はホットスポットを基準に回転され、画面上の位置が保持されます。回転された領域の例を次に示します:

### 5.3色の乗算エフェクト

この効果は、領域が描画されるときに常に適用されます。描画されるすべてのピクセルのカラーコンポーネントは、最初に単一の指定されたカラー(乗算カラー)で乗算されます。乗算カラーの効果はフィルタの効果です。フィルタは、画面に描画される前に、すべての領域ピクセルの4つのカラーコンポーネントを調整します。この効果は、OpenGLでglColor関数を使用する場合と同じです。

この効果の例として、(0,255,255,255)の乗算カラーを使用すると、描画領域の赤コンポーネントが削除されますが、その他のコンポーネントは保持されます。画面に表示される視覚効果は次のようになります。

描画される色を変更する方法の式は次のとおりです。

描かれたR=(描かれたR\*乗算R)/255

描かれたG=(描かれたG\*乗算G)/255

描画されたB=(描画されたB\*乗算B)/255

描画されたA=(描画されたA\*乗算A)/255

各成分の変調効果は、これらの式が比率として記述されている場合、よりよく視覚化できます。作図X\*(X/255を乗算)ただし、一部の実装では、この操作の順序によって結果が正しくない場合があることに注意してください。

乗算カラーが完全に不透明な白の場合、描画されたカラーには影響しません。これはデフォルトであるため、実際にコンソールが起動すると、この効果は「無効」と見なされます。

## 6カラーブレンディング

カラーブレンディングは、描画バッファからいくつかのピクセルに新しい色を描画するために使用されるプロセスです。Vircon32グラフィックスは基本的な透明度と照明効果をサポートしているため、このプロセスを定義する必要があります。このようなグラフィックシステムでは、新しい色を描画することは、描画された色が単に前のピクセル色を置き換えることを意味するとは限りません。代わりに、異なる方法で前のピクセル色に依存する場合もあります。

GPUは3種類のカラーブレンディングモードをサポートしています。任意の時点で3つのモードのいずれかがアクティブになり、実行されたグラフィック機能はそのモードを使用して描画バッファに描画されます。次の図は、3つのモードがどのようなものかを簡単に示しています。

### 6.1アルファブレンディング

これがデフォルトの描画モードです。アルファチャンネルを考慮して背景の透明度のレベルを決定しながら、カラーをそのまま描画します。最も基本的な形式(完全な不透明度、透明度なし)では、その効果は次のようになります。バッファカラー=描画カラーアルファを考慮に入れると、効果は次のように調整されます。

バッファR=(描画されたR\*描画されたA+バッファR\*(255-描画されたA))/255

バッファG=(描画G\*描画A+バッファG\*(255-描画A))/255

バッファB=(描画されたB\*描画されたA+バッファB\*(255-描画されたA))/255

結果はすでに[0-255]の範囲内にあるため、クリッピングは必要ありません。完全に透明な描画色は変更されません。

### 6.2添加配合

このモードは、カラーを明るくし、ライトエフェクトを生成するために使用されます。最も基本的な形式(完全な不透明度)では、その効果は次のようになります。バッファカラー+=描画カラーをクリックし、[0-255]にクリップします。アルファチャンネルを追加すると、エフェクトは次のように調整されます。

バッファR=バッファR+(作図R\*作図A)/255

バッファG=バッファG+(描かれたG\*描かれたA)/255

バッファB=バッファB+(描画されたB\*描画されたA)/255

結果として得られる3つのRGBコンポーネントは、最大255に制限されます。描画される色が黒または完全に透明の場合、変更は生じません。この描画モードは、多くの描画プログラムで「加算」または「リニアドッジ」と呼ばれるレイヤーモードに類似しています。

### 6.3減算ブレンディング

このブレンドは、色を暗くし、シャドウ効果を生成するために使用されます。最も基本的な形式(完全な不透明度)では、その効果は次のようになります:バッファカラー(Buffer Color)**-**=描画色をクリックし、[0-255]にクリップします。アルファチャンネルを追加すると、エフェクトは次のように調整されます。

スクリーンR=スクリーンR-(描かれたR\*描かれたA)/255

スクリーンG=スクリーンG-(描かれたG\*描かれたA)/255

スクリーンB=スクリーンB-(描かれたB\*描かれたA)/255

結果として得られる3つのRGBコンポーネントは、最小0に制限されます。描画される色が黒または完全に透明の場合、変更は生じません。この描画モードは、多くの描画プログラムで「減算」または「差分」と呼ばれるレイヤーモードに類似しています。

## 7 GPUパフォーマンス

GPUは、そのパフォーマンス制限を確立しない限り、完全には定義されません。このGPUの場合、パフォーマンスは描画されるピクセルの量の概算に基づいています。GPUの各フレームには、全画面の9倍の描画容量が許可されます。これは、1920 x 1080ピクセルの1つの全画面と同じピクセル量です。

### 7.1描画画素の計算

有効な描画操作要求を受け取った後にGPUが最初に実行する必要があるのは、描画されるピクセル数の簡易推定を実行することです。

clear screenコマンドの場合、描画されるピクセルの量は常に1フルスクリーン=640 x 360ピクセルです。領域を描画する場合、計算プロセスは次の手順に従います。

1. 次の値を計算します。「有効幅」次のようになります。まず、テクスチャピクセルの領域幅を取得します。スケーリングが適用されている場合は、それにXのスケーリング係数を掛けます。次に、絶対値を取得し、結果が画面幅より大きい場合は、640ピクセルに制限します。
2. 次の値を計算します。「有効高さ」:次のようになります。最初にテクスチャピクセルの領域の高さを取得します。スケーリングが適用されている場合は、それにYのスケーリング係数を乗算します。次に絶対値を取得し、結果が画面の高さより大きい場合は、360ピクセルに制限します。
3. 最後に計算"描画されたピクセル"「有効幅」x「有効高さ」

この計算では、領域の描画位置と回転角度が無視されることに注意してください。これは、画面の外側にある領域の部分も描画されたピクセルとしてカウントされることを意味します。ただし、この不正確さにより、計算プロセスが大幅に簡素化される場合があります。

### 7.2異なる運用のパフォーマンス要因

すべての操作がGPUにとって同じようにコストがかかるわけではありません。これをモデル化するには、実行される操作に応じて、描画されるピクセルの数が次の要因によって変更されます:

* + 画面をクリアする場合:-50%
  + 尺度を適用して領域を描画する場合:+15%
  + 回転を適用して領域を描画する場合:+25%  
    (スケーリングと回転の両方が適用される場合、組み合わされた係数は+40%です)

乗算カラーやカラーブレンディングの適用などのカラー処理操作は、パフォーマンスに影響せず、関連する要因もないことに注意してください。

### 7.3コマンドのパフォーマンスをチェックする

受信したコマンド要求に対して描画するピクセル数を決定した後、GPUはその量を現在の残りのピクセルと比較します。

* このフレームに描画操作に十分な残りのピクセルがある場合は、コマンドが実行され、計算に従って残りのピクセルが削減されます。
* 逆に、現在の残りのピクセルが操作に十分でない場合、GPUは代わりに残りのピクセルを-1に設定し、コマンドを実行しません。現在のフレーム内で受信されたそれ以降の要求も無視されます。

この最後の点のために、実装は、このプロセスを単純化することができ、残りの画素が0以下であるときに、いかなる計算もなしに、操作を自動的に拒絶することに留意されたい。

GPU描画容量はフレーム間で転送できません。新しいフレームでは、前のフレームに残っているピクセルに関係なく、同じピクセル容量が復元されます。

## 8内部変数

GPUは、内部状態のさまざまな側面を格納する一連の変数を備えています。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、浮動小数点など)を使用してすべて解釈されます。ここでは、すべての変数をセクションに分けてリストし、詳細を説明します。

### 8.1 GPUコントロールの変数

| 残りのピクセル | **初期値:**2073600(=9\*640\*360) |
| --- | --- |
| **形式:**整数 | **有効範囲:**-1～2073600 |

このフレームのGPUの残りの描画容量を格納します。描画要求を実行できない場合(つまり、コマンドを実行するのに残りのピクセルが不十分な場合)、GPUは次のフレームが開始されるまでブロックされ、現在のフレームではコマンドを実行できなくなります。値-1は、このフレームの描画容量が使い果たされたことを示すために使用されます。

### 8.2空間図面パラメータ

| 図面ポイントX | **初期値:**0 |
| --- | --- |
| **形式:**整数 | **有効範囲:**-1000から1639まで |

これは、次に描画される領域のホットスポットが画面上に配置されるX座標をピクセル単位でマークします。これが画面の幅から外れると、描画される領域の一部または全体の可視性が失われる可能性があることに注意してください。

| 図面ポイントY | **初期値:**0 |
| --- | --- |
| **形式:**整数 | **有効範囲:**-1000から1359まで |

これは、次に描画される領域のホットスポットが画面上に配置されるY座標をピクセル単位でマークします。これが画面の高さから外れると、描画される領域の一部または全体の可視性が失われる可能性があることに注意してください。

| 図面縮尺X | **初期値:**1.0 |
| --- | --- |
| **形式:**フロート | **有効範囲:**-1024.0から1024.0 |

この描画パラメータは、スケーリングを有効にするコマンドでのみ適用されます。このパラメータを使用すると、描画される領域は、この係数によってテクスチャのX次元に沿ってスケーリングされます。Xに沿ったミラー効果の場合は、負の値にすることもできます。

1ピクセル未満の出力幅を生成する縮小スケールで描画する場合、結果は実装依存であると見なされます。つまり、領域を1ピクセルの垂直線に圧縮するか、まったく描画しないかのいずれかです。

| 図面尺度Y | **初期値:**1.0 |
| --- | --- |
| **形式:**フロート | **有効範囲:**-1024.0から1024.0 |

この描画パラメータは、スケーリングを有効にするコマンドでのみ適用されます。このパラメータを使用すると、描画される領域は、この係数によってテクスチャのY次元に沿ってスケーリングされます。Yに沿ったミラー効果の場合は、負の値にすることもできます。

1ピクセル未満の出力高さを生成する縮小スケールで描画する場合、結果は実装に依存すると考えられる:領域を1ピクセルの水平線に圧縮するか、何も描画しないかのいずれかである。

| 描画角度 | **初期値:**0.0 |
| --- | --- |
| **形式:**フロート | **有効範囲:**-1024.0から1024.0 |

この描画パラメータは、回転を有効にするコマンドでのみ適用されます。これは、画面に描画される領域に適用される回転角度です。値はラジアンで解釈されます(1完全な円=2\*piラジアン=360度)。角度0は回転がないことを意味し、正の値の場合、領域は時計回りに回転します。

### 8.3色処理変数

| クリアカラー | **初期値:**R=0、G=0、B=0、A=255の場合 |
| --- | --- |
| **形式:**GPUカラー | **有効範囲:**完全なRGBA範囲 |

現在のクリアカラーは、画面のクリアコマンドが実行されるたびに適用されるカラーです。アルファコンポーネントを調整することによって透明度をサポートします。

| カラーの乗算(Multiply color | **初期値:**R=255、G=255、B=255、A=255の場合 |
| --- | --- |
| **形式:**GPUカラー | **有効範囲:**完全なRGBA範囲 |

このカラーは、領域描画コマンドを実行するときに常に適用されるカラー乗算エフェクトを制御します。初期値はニュートラル乗算カラー(エフェクトなし)です。

| アクティブな描画モード | **初期値:**20h(アルファブレンディング) |
| --- | --- |
| **形式:**整数 | **有効範囲:**リストされた値のみ |

この値は、現在アクティブなカラーブレンディングモードとして解釈され、すべての描画機能でのカラーの描画方法を制御します。各モードの動作の詳細については、カラーブレンディングのセクションを参照してください。設定可能な値は次のとおりです。

20h:アルファブレンディング

21h:添加混合

22h:サブトラクションブレンディング

### 8.4選択されたエレメント

| 選択されたテクスチャ | **初期値:**-1個 |
| --- | --- |
| **形式:**整数 | **有効範囲:**-1～255(\*) |

この値は、現在選択されているGPUテクスチャの数値IDです。選択されたテクスチャは、すべての領域描画コマンドで使用されるテクスチャです。また、領域設定の変更によって領域が影響を受けるテクスチャでもあります。

(\*)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テクスチャ(ID=-1)のみが選択可能なテクスチャです。

| 選択された領域 | **初期値:**0 |
| --- | --- |
| **形式:**整数 | **有効範囲:**0～4095 |

この値は、選択されたテクスチャ内で現在選択されているGPU領域の数値IDです。選択されたテクスチャを変更しても、選択された領域IDは変更されません。これは、選択された領域IDがグローバルパラメータであるためです(各テクスチャに対して選択された領域はありません)。選択された領域(現在選択されているテクスチャから)は、すべての領域描画コマンドで使用される領域です。また、領域設定の変更によって影響を受ける領域でもあります。

### 8.5各テクスチャ領域の構成

ここにリストされている変数は特別なものです。GPUは、可能なテクスチャ領域ごとにこれらの変数のコピーを保存します。これは、(256+1)テクスチャ\*4096領域/テクスチャと同じ数が存在する可能性があることを意味します。実装では、未使用のテクスチャスロットの変数をアクセスできないようにして、これらすべてを常に保存するか、新しいカートリッジが挿入されるたびに必要な変数のみを作成するかを決定できます。

これらの変数の各セットは、1つのGPU領域の現在の設定を示します。領域の定義方法については、第3章のGPUテクスチャ領域のセクションを参照してください。GPUでは、領域の拡張を定義する最小値と最大値が逆になってもかまいません。これにより、イメージのフリップ効果が生成されます。

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。選択したテクスチャまたは領域が変更されると、これらのポートはすべて、正しい領域の変数のコピーにリダイレクトされます。

| 領域の最小X | **初期値:**0 |
| --- | --- |
| **形式:**整数 | **有効範囲:**0～1023 |

領域の一部と見なされるテクスチャの左端のX座標を表します。テクスチャ座標でピクセル単位で指定します。

| 領域最小Y | **初期値:**0 |
| --- | --- |
| **形式:**整数 | **有効範囲:**0～1023 |

領域の一部とみなされるテクスチャの一番上のY座標を表します。テクスチャ座標でピクセル単位で指定します。

| 領域の最大X | **初期値:**0 |
| --- | --- |
| **形式:**整数 | **有効範囲:**0～1023 |

領域の一部と見なされるテクスチャの右端のX座標を表します。テクスチャ座標でピクセル単位で指定します。

| 領域の最大Y | **初期値:**0 |
| --- | --- |
| **形式:**整数 | **有効範囲:**0～1023 |

領域の一部とみなされるテクスチャの最下部のY座標を表します。テクスチャ座標でピクセル単位で指定します。

| 領域のホットスポットX | **初期値:**0 |
| --- | --- |
| **形式:**整数 | **有効範囲:**-1024から2047 |

領域を描画するときに参照として使用されるのは、テクスチャ座標のX座標です。GPUは、ホットスポットが描画ポイントに描画されるように領域の配置を計算します。ホットスポットは、指定された範囲の制限内で、テクスチャ境界の外側に配置できます。

| 領域のホットスポットY | **初期値:**0 |
| --- | --- |
| **形式:**整数 | **有効範囲:**-1024から2047 |

領域を描画するときに参照として使用されるのは、テクスチャ座標のY座標です。GPUは、ホットスポットが描画ポイントに描画されるように領域の配置を計算します。ホットスポットは、指定された範囲の制限内で、テクスチャ境界の外側に配置できます。

## 9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GPU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 |
| 200時間 | 00時間 | コマンド(Command) | 書き込み専用 |
| 201時間 | 01時間 | 残りのピクセル | 読み取り専用 |
| 202時間 | 02時間 | クリアカラー | 読み取り/書き込み |
| 203時間 | 03時間 | カラーの乗算(Multiply Color) | 読み取り/書き込み |
| 204時間 | 04時間 | アクティブブレンディング(Active Blending) | 読み取り/書き込み |
| 205時間 | 05時間 | 選択されたテクスチャ | 読み取り/書き込み |
| 206時間 | 06時間 | 選択された領域 | 読み取り/書き込み |
| 207時間 | 07時間 | 図面ポイントX | 読み取り/書き込み |
| 208時間 | 08時間 | 図面ポイントY | 読み取り/書き込み |
| 209時間 | 09時間 | 図面縮尺X | 読み取り/書き込み |
| 20 Ah弾 | 0 Ah(ゼロ時) | 図面尺度Y | 読み取り/書き込み |
| 20 Bh弾 | 0時 | 作図角度 | 読み取り/書き込み |
| 20時 | 0 Ch(0チャネル) | 領域最小X(Region Min X) | 読み取り/書き込み |
| 20 Dh弾 | 0度Dh | 領域最小Y(Region Min Y) | 読み取り/書き込み |
| 20時 | 0 Eh(0時) | 領域最大X(Region Max X) | 読み取り/書き込み |
| 20 Fh弾 | 0時Fh | 領域最大Y | 読み取り/書き込み |
| 210時間 | 10時間 | リージョンホットスポットX | 読み取り/書き込み |
| 211時間 | 11時間 | リージョンホットスポットY | 読み取り/書き込み |

### 9.1ポートの読み取り/書き込み要求に対する動作

GPU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GPU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、GPU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:

GPUは、セクション10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、コマンドが実行されなくても、リクエストは成功して応答されます。

### [Remaining Pixels]ポート

#### オン読み取る要求:

GPUは、内部変数「Remaining pixels」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。

### [Clear Color]ポート

#### オン読み取る要求:

GPUは、内部変数「Clear color」の現在の値を提供します。

#### オン書く要求:

GPUは、内部変数「Clear color」を受信した値で上書きします。これにより、次の画面クリア操作で新しいクリアカラーがすぐに適用されます。この変数を設定しても、画面はクリアされません。

### [Multiply Color]ポート

#### オン読み取る要求:

GPUは、内部変数「Multiply color」の現在の値を提供します。

#### オン書く要求:

GPUは、内部変数「Multiply color」を受信した値で上書きします。これにより、次の領域描画操作で新しい乗算カラーがすぐに適用されます。

### [Active Blending]ポート

#### オン読み取る要求:

GPUは、内部変数「Active blending mode」の現在の値を提供します。

#### オン書く要求:

GPUは、受信した値が現在有効な描画モード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、GPUは内部変数「アクティブ描画モード」を受信した値で上書きします。これにより、次の描画操作ですぐに新しい描画モードが適用されます。

### [Selected Texture]ポート

#### オン読み取る要求:

GPUは、内部変数「Selected texture」の現在の値を提供します。

#### オン書く要求:

GPUは、受信した値が現在有効なテクスチャ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、GPUは内部変数「Selected texture」を受信した値で上書きします。その後、すべての領域設定ポートを、新しく選択された領域(つまり、同じ領域IDですが、新しく選択されたテクスチャから取得されたもの)の変数のセットを指すようにリダイレクトします。

### [Selected Region]ポート

#### オン読み取る要求:

GPUは、内部変数「Selected region」の現在の値を提供します。

#### オン書く要求:

GPUは、受信した値が有効な領域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、GPUは内部変数「Selected region」を受信した値で上書きします。その後、すべての領域設定ポートを、新しく選択された領域(つまり、新しい領域IDですが、選択された同じテクスチャから取得されます)の変数のセットを指すようにリダイレクトします。

### 図面ポイントXポート

#### オン読み取る要求:

GPUは、内部変数「Drawing point X」の現在の値を提供します。

#### オン書く要求:

GPUは、受け取った値が内部変数"Drawing point X"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。その後、結果の値は内部変数"Drawing point X"を上書きします。これにより、次の描画操作で新しい描画位置がすぐに適用されます。

### 図面ポイントYポート

#### オン読み取る要求:

GPUは、内部変数「Drawing point Y」の現在の値を提供します。

#### オン書く要求:

GPUは、受け取った値が内部変数"Drawing point Y"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。その後、結果の値は内部変数"Drawing point Y"を上書きします。これにより、次の描画操作で新しい描画位置がすぐに適用されます。

### 図面縮尺Xポート

#### オン読み取る要求:

GPUは、内部変数「Drawing scale X」の現在の値を提供します。

#### オン書く要求:

GPUは、受信した値が内部変数「図面スケールX」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、範囲制限にクランプされます。結果の値は、内部変数「図面スケールX」を上書きします。これにより、スケーリングが有効になっている次の図面操作で、Xの新しい図面スケールがすぐに適用されます。

### 図面縮尺Yポート

#### オン読み取る要求:

GPUは、内部変数「Drawing scale Y」の現在の値を提供します。

#### オン書く要求:

GPUは、受信した値が内部変数「図面スケールY」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、範囲制限にクランプされます。結果の値は、内部変数「図面スケールY」を上書きします。これにより、スケールが有効になっている次の図面操作で、Yの新しい図面スケールがすぐに適用されます。

### 領域の最小Xポート

#### オン読み取る要求:

GPUは、現在選択されているテクスチャIDに対して、現在選択されている領域IDに関連付けられている内部変数「Region minimum X」の現在の値を提供します。

#### オン書く要求:

GPUは、受信した値が内部変数「Region minimum X」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、現在選択されているテクスチャIDに対して、現在選択されている領域IDに関連付けられている内部変数「Region minimum X」を上書きします。これにより、次の領域描画操作では、選択されている領域に新しい左制限が適用されます。

### 領域の最小Yポート

#### オン読み取る要求:

GPUは、現在選択されているテクスチャIDに対して、現在選択されている領域IDに関連付けられている内部変数「Region minimum Y」の現在の値を提供します。

#### オン書く要求:

GPUは、受信した値が内部変数「Region minimum Y」の範囲外であるかどうかをチェックします。範囲外の場合は、範囲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テクスチャIDに対して、現在選択されている領域IDに関連付けられている内部変数「Region minimum Y」を上書きします。これにより、次の領域描画操作では、選択されている領域に新しい上限が適用されます。

### Region Max Xポート

#### オン読み取る要求:

GPUは、現在選択されているテクスチャIDに対して、現在選択されている領域IDに関連付けられている内部変数「Region maximum X」の現在の値を提供します。

#### オン書く要求:

GPUは、受け取った値が内部変数「Region maximum X」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、現在選択されているテクスチャIDに対して、現在選択されている領域IDに関連付けられている内部変数「Region maximum X」を上書きします。これにより、次の領域描画操作では、選択されている領域に新しい右制限が適用されます。

### [Region Max Y]ポート

#### オン読み取る要求:

GPUは、現在選択されているテクスチャIDに対して、現在選択されている領域IDに関連付けられている内部変数「領域最大Y」の現在の値を提供します。

#### オン書く要求:

GPUは、受け取った値が内部変数「領域最大Y」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テクスチャIDに対して、現在選択されている領域IDに関連付けられている内部変数「領域最大Y」を上書きします。これにより、次の領域描画操作では、選択された領域に新しい下限が適用されます。

### リージョンホットスポットXポート

#### オン読み取る要求:

GPUは、現在選択されているテクスチャIDに対して、現在選択されている領域IDに関連付けられている内部変数「Region hotspot X」の現在の値を提供します。

#### オン書く要求:

GPUは、受信した値が内部変数「Region hotspot X」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、現在選択されているテクスチャIDに対して、現在選択されている領域IDに関連付けられている内部変数「Region hotspot X」を上書きします。これにより、次の領域描画操作では、選択されている領域に新しいホットスポット位置がすぐに適用されます。

### リージョンホットスポットYポート

#### オン読み取る要求:

GPUは、現在選択されているテクスチャIDに対して、現在選択されている領域IDに関連付けられている内部変数「Region hotspot Y」の現在の値を提供します。

#### オン書く要求:

GPUは、受信した値が内部変数「Region hotspot Y」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、現在選択されているテクスチャIDに対して、現在選択されている領域IDに関連付けられている内部変数「Region hotspot Y」を上書きします。これにより、次の領域描画操作で、選択されている領域に新しいホットスポット位置がすぐに適用されます。

## 10コマンドの実行

GPUコマンドは、CPUがGPU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描画操作です。このセクションでは、このような要求を受信したときのGPUの動作について説明します。

GPUで実行できるコマンドは5種類あります。この表には、コマンドの数値と、各コマンドで使用されるグラフィック効果が示されています。

| コマンド名 | 数値 | スケーリング | 回転 | カラーの乗算(Multiply color |
| --- | --- | --- | --- | --- |
| 画面をクリア | 10時間 | いいえ | いいえ | いいえ |
| 領域を描画 | 11時間 | いいえ | いいえ | はい |
| ズームした領域を描画 | 12時間 | はい | いいえ | はい |
| 回転した領域を描画 | 13時間 | いいえ | はい | はい |
| [作成][リージョン][Rotozoomed] | 14時間 | はい | はい | はい |

### 10.1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、GPUは、要求された書き込み値が上記の有効なコマンドのいずれか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、それ以上の処理は行われません。

この後、GPUは、セクション7(GPUパフォーマンス)で説明されているように、現在の残りのピクセルが要求された操作を実行するのに十分であるかどうかをチェックします。

次に、残りのピクセルが不足しているためにコマンドが中止されなかった場合、GPUは描画バッファに描画します。この部分はコマンド固有であり、次のサブセクションで説明します。ただし、すべてのコマンドについて、カラーブレンディングプロセスは、「アクティブブレンディングモード」変数の現在の値についてセクション6で説明されているように実行されます。

### 10.2 Clear Screenコマンド

GPUは、内部変数「Clear color」に現在格納されているカラーを使用して、描画バッファ描画内のすべてのピクセルを描画します。ブレンド以外には、エフェクトは適用されません。

### 10.3[Draw Region]コマンド

GPUは、現在選択されているテクスチャの矩形部分を描画バッファ上に描画します。この矩形は、そのテクスチャに対して現在選択されている領域IDに対応する領域設定変数によって区切られます。

領域は、ホットスポットとしてマークされた領域のピクセルが、描画バッファ内の現在のGPU描画位置によってマークされた座標に描画されるように配置されます。

領域を描画するとき、GPUは"Multiply color"変数に格納されている現在のGPUカラーを使用してカラーの乗算を適用します。カラー操作の順序は次のとおりです。

1. 領域のピクセルと乗算カラーの間でカラー乗算を実行します。
2. (1)で得られたカラーを描画バッファの内容にブレンドします。

このコマンドは、描画スケールおよび描画角度変数の現在の値を無視します。領域は常に空間変換なしで描画されます。

### 10.4[Draw Region Zoomed]コマンド

処理はコマンド"Draw Region"と同じですが、描画された領域もスケールされます。XおよびYに沿ったスケール係数は、それぞれ"Drawing scale X"および"Drawing scale Y"変数の現在の値になります。XまたはYに沿った負のスケール係数は、その次元のミラー効果をもたらします。

スケール係数にかかわらず、領域のホットスポットが描画バッファ内のGPU描画位置に描画され続けるように、スケーリングが行われます。

### 10.5[Draw Region Rotated]コマンド

処理はコマンド"Draw Region"と同じですが、描画された領域も回転されます。回転角度は変数"Drawing angle"の現在の値になります。これはラジアンで解釈され、正の角度を指定すると領域は時計回りに回転します。

回転は、描画された領域のホットスポットを中心として行われます。このようにして、領域のホットスポットは、描画バッファ内のGPU描画位置に描画され続けます。

### 10.6[Draw Region Rotozoomed]コマンド

処理はコマンド"Draw Region"と同じですが、描画された領域も拡大縮小および回転されます。これらの効果は、コマンド"Draw Region Zoomed"および"Draw Region Rotated"で説明されているのと同じ方法で適用されます。

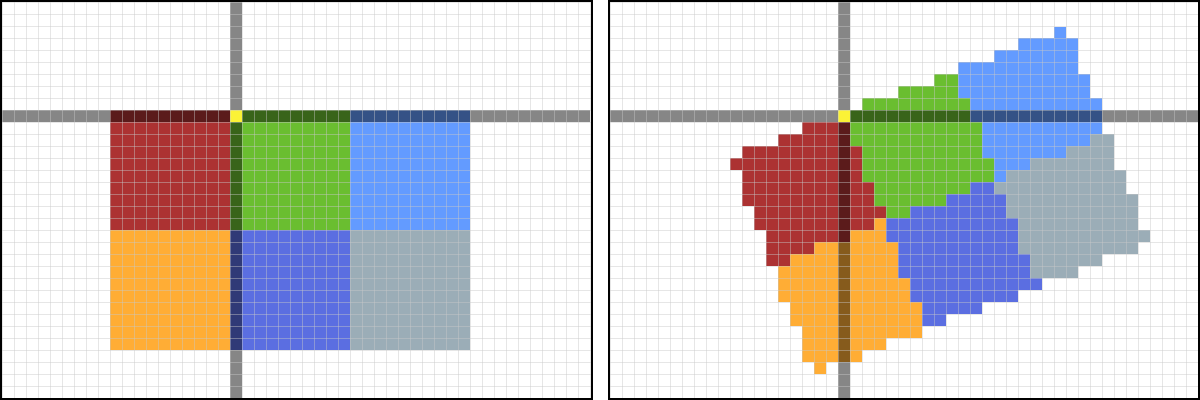
スケーリングと回転の効果が組み合わされ、領域のホットスポットが描画バッファ内のGPU描画位置に描画され続けるようになります。また、両方の効果の組み合わせは、常にテクスチャのX軸とY軸に対してスケーリングを実行する必要があり、画面座標に対してではありません。つまり、変換の順序は、最初にスケーリングしてから回転する必要があります。

### 10.7 2D変換に関する注意

#### 大きなスケール係数のための領域配置

大きなスケールで描画する場合、テクスチャピクセルを配置するための基準は、ピクセルの中心ではなく、その左上隅になります。つまり、係数X=640.0およびY=360.0で1つのピクセルをスケールして、位置(0,0)に描画すると、スクリーン全体が正確にカバーされます。

例として、画像(3 x 2ピクセル)に表示される領域を大きなスケールで描画します。この領域のホットスポットは、その緑色のピクセルに配置されます。暗くなった交差線によって与えられる描画位置を使用すると、画面には次のような結果が表示されます。



領域が回転なしで描画される場合(左のイメージ)、緑のピクセルの左上が描画位置と重なります。回転も有効にすると(右のイメージ)、同じ画面ピクセルが回転の中心として使用されるため、緑のピクセルの元の左上隅が描画位置に描画され続けます。

#### 補間方法

Vircon32 GPUの参照補間法は最も近いものです。ただし、実装では、イメージのスケーリングと回転に対して他の補間法を自由に選択できます。ただし、これらのアルゴリズムは元の色を保持するために必要です。

つまり、リニア補間やキュービック補間などのカラースムージング方法は、Vircon32に準拠していません。ただし、Scale2XやRotSpriteなどのピクセルアートアルゴリズムは、ピクセルカラーを保持するように設計されているため、受け入れられます。

### 10.8コマンド実行時間

この仕様では、描画コマンドを実際に実行するのにかかる時間については言及していません。代わりに、Vircon32システムはこれを抽象化し、すべての描画操作を常に即座に完了したかのように扱います。この結果、CPUは前のコマンドが完了する前に、より多くのコマンドを要求できます。

この抽象化の理由は、Vircon32システムは、コンポーネント間の操作を同期させる必要がないように設計されているためです。多くの場合、これは有効な近似です:最新のGPUはVircon32よりもはるかに高速です。ただし、実際の描画操作の中には、完了までに数CPUサイクルを要するものもあります。このため、実装には、描画操作を管理するための何らかの戦略が必要になります。

戦略にはいくつかのタイプがあります。OpenGLなどの一部のグラフィックシステムでは、タスクはバックグラウンドで実行され、メインプログラムにとって問題にならない場合があります。ハードウェア実装の場合、並列に管理されるコマンドキューを実装するだけで十分な場合があります。一方、ソフトウェア実装では、この点を考慮してコンソール操作全体を計画し、最後に要求された操作が完了するまで次のサイクルを延期する場合があります。

それでも、グラフィカルな正確性のためには、いかなる実装も次のことを保証しなければならない。

1. 受け入れられた描画操作が実行されないままになることはありません。
2. 図面操作は、要求された順序を維持して実行されます。
3. すべての描画操作は、現在のフレームが終了する前に完了する必要があります。

## 11ビデオ出力の生成

このGPUのビデオ出力は、コンソールのグローバルタイミングによって制御されるため、フレームに基づいています。60 Hzのレートでは、新しいフレーム信号が受信されるたびに、GPUは新しいビデオフレームの生成と送信をトリガーします。

GPUは、その時点で描画バッファ内のすべてのピクセルに必要なすべてのカラー情報を収集することによって、ビデオフレームを生成します。次に、GPUはその情報をビデオ出力信号を介してディスプレイに転送します。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、有効なビデオフレーム送信が形成される。

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。

描画バッファの内容は、新しいフレームが開始されたとき、またはビデオ出力プロセスの一部としてクリアまたは変更されないことに注意してください。描画バッファの内容は永続的であり、リセットまたは電源オンイベント時にのみ自動的にクリアされます。

## 12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、GPU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、GPU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:

* 図面操作がまだ実行中の場合は、中止されます。
* すべてのGPU内部変数が初期値に設定されます。これには、すべてのテクスチャ内のすべての領域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。
* 描画バッファ内のすべてのピクセルが黒(R=0、G=0、B=0)に設定されます。

#### フレーム信号:

* 図面操作がまだ実行中の場合は、中止されます。
* GPUは、セクション11で説明するように、新しいビデオフレームの出力を生成します。
* 「Remaining pixels」内部変数は、初期値にリセットされます。

#### サイクル信号:

* GPUは、特定の実装の詳細で必要とされない限り、この信号に反応する必要はありません。

GPU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答して次の処理を実行する必要があります。

#### 新しいカートリッジが接続されたとき:

* GPUは、BIOSおよびカートリッジ(存在する場合)に含まれるイメージを検出します。
* GPUは、セクション3で説明した接続プロセスを実行して、各イメージにテクスチャスロットを割り当て、そのピクセル情報にアクセスします。
* 変数「Selected texture」の上限は、最後に使用されたテクスチャスロットIDに調整されます。

(パート4の終わり)