Power Movie PCI2 Development Kit

Programmer's Manual

canopus

ご使用の前に

■絵表示について

本製品を安全に正しくお使いいただくために、以下の内容をよく理解してから本文をお読みください。

警告

人が死亡または重傷を負う恐れのある内容を示しています。

注意

けがをしたり財産に損害を受ける恐れのある内容を示しています。

■絵表示の意味

この記号はしてはいけないことを表しています。

この記号はしなければならないことを表しています。

この記号は気をつける必要があることを表しています。

■ご購入製品を使用される際の注意事項

ここでは、ご購入製品を使用されるときにご注意いただきたい事柄について説明しています。

警告

●健康上のご注意

ごくまれに、コンピュータのモニタに表示される強い光の刺激や点滅によって、一時的にてんかん・意識の喪失などが引き起こされる場合があります。こうした経験をこれまでにされたことがない方でも、それが起こる体質をもっていることも考えられます。こうした経験をお持ちの方や、経験をお持ちの方の血縁にあたられる方は、本製品を使用される前に必ず医師と相談してください。

警告

●製品のご利用についての注意事項

医療機器や人命に関わるシステムでは、絶対にご利用にならないでください。製品の性質上、これらのシステムへの導入は適しません。

●製品の取り付けおよび取り外しに関する注意事項

製品の取り付けおよび取り外しを行う場合必ずパソコン本体および周辺機器の電源を切り、さらに電源ケーブルをコンセントから抜いた状態で行ってください。

パソコン本体および周辺機器の電源を入れたまま製品を取り付けたり取り外したりした場合、製品やパソコン本体、周辺機器および周辺機器に接続されている機器の一部が破壊される恐れがあります。また、パソコン本体および周辺機器の電源ケーブルをコンセントから抜かずにパソコン本体や周辺機器の筐体（電源ユニットなど）、機器の金属部分に触れた場合には感電する恐れがあります。

●静電気に関する注意事項

製品に静電気が流れると製品上の部品が破壊される恐れがあります。各コネクタや部品面には直接手を触れないでください。

静電気は衣服や人体からも発生します。製品に触れる前に、一旦接地された金属製のものに触れてください（体内の静電気を放電することになります）。

注意

●消費電流に関する注意事項

複数の拡張ボードをパソコンに取り付けるときは、ご購入製品を含めたすべての製品の消費電流の合計がパソコンの最大供給電流を超えていないことを必ず確認してください。全ボードの消費電流の合計がパソコンの最大供給電流を超えたりするなどの動作条件を満たさない環境で使用し続けると、システムが正常に動作しない場合やシステムに負荷がかかり、パソコンが故障する原因となる恐れがあります。

消費電流のわからない製品については、その製品の取扱説明書をご覧いただくか、メーカーに直接お問い合わせいただいてお確かめください。

●他社製品と併用されるときの注意事項

他社製品と併用されるとご購入製品が正常に動作しないことがあり、そのためにシステムが本来の目的を達成することができないこともあります。あらかじめ、製品単体の環境で購入製品が正常に動作することをご確認ください。また、他社製品との併用によって購入製品が正常に動作しないのであれば、その他社製品と購入製品との併用はお止めください。

●その他の注意事項

製品は指定された位置に指示通り取り付けてください。指示通りに取り付けられていない場合、製品の金属部分とパソコンの金属部分が接触してショートするなどの要因で、製品やパソコン本体・周辺機器が破壊される恐れがあります。

製品を取り扱うときは手など皮膚を傷つけないよう十分にご注意ください。ハードウェアの仕様上、製品のパネル、コネクタ、エッジ、裏面は金属のピンが、突出していることがあります。製品を取り付けたり取り外したりするときは、製品全体を軽く包み込むようにお持ちください。

動作中の製品は熱により非常に熱くなります。長時間使用した製品に手を触れる際には、十分にご注意ください。

ご注意

(1) 本製品の一部または全部を無断で複製することを禁止します。
(2) 本製品の内容や仕様は将来予告無しに変更することがあります。
(3) 本製品は内容について万全を期して作成いたしましたが、万一ご不審な点や誤り、記載漏れなどお気付きの事がございましたら、当社までご連絡ください。
(4) 運用した結果については、(3) 項にかかわらず責任を負いかねますので、ご了承ください。
(5) ご使用上の過失の有無を問わず、本製品の運用において発生した逸失利益を含む特別、付随的、または派生的損害に対するいかなる請求があったとしても、当社はその責任を負わないものとします。
(6) 本製品付属のソフトウェア、ハードウェア、マニュアル、その他添付物を含めたすべての関連製品に関して、解析、リバースエンジニアリング、デコンパイル、ディスアッセンブリを禁じます。
(7) カノープス、CANOPUS/ カノープスおよびそのロゴは、カノープス株式会社の登録商標です。
(8) Microsoft、Windows は米国マイクロソフト・コーポレーションの登録商標です。また、その他の商品名やそれに類するものは各社の商標または登録商標です。
(9) Adobe、 Adobe ロゴ、 Adobe Reader は Adobe Systems Incorporated(アドビシステムズ社) の商標または登録商標です。

表記について

■本書は Power Movie PCI2 Development Kit のセットアップおよび使用方法について記載したものです。
■ Power Movie PCI2 Development Kit で作成したプログラムは Power Movie PCI2 および Power Movie PCI でご使用いただけます。
本書では Power Movie PCI2 および Power Movie PCI を Power Movie PCI と記します。
■本書に記載されていない情報が記載される場合がありますので、ディスクに添付のテキストファイル・オンラインマニュアルも必ずお読みください。
■本書での説明と実際の運用方法とで相違点がある場合には、実際の運用方法を優先するものとします。
■本書はパソコンの基本的な操作を行うことができる方を対象に書れています。特に記載の無い操作については、一般的なパソコンの操作と同じように行ってください。
■本書では Microsoft® Windows® 2000 operating system、Microsoft® Windows® XP operating system を Windows 2000、Windows XP と表記します。
■説明の便宜上、実際の製品とイラスト及び画面写真が異なる場合があります。

個人情報の取扱いについて

当社では、原則として①ご記入いただいたお客様の個人情報は下記の目的以外では使用せず、②下記以外の目的で使用する場合は事前に当該サービス上にてお知らせいたします。
当社ではご記入いただいた情報を適切に管理し、特段の事情がない限りお客様の承諾なく第三者に開示・提供することはございません。

1．ご利用の当社製品のサポートの実施
2．当社製品の使用状況調査、製品改良、製品開発、サービス向上を目的としたアンケートの実施
　＊調査結果を当社のビジネスパートナーに参考資料として提供することがありますが、匿名性を確保した状態で提供いたします。
3．銀行口座やクレジットカードの正当性、有効性の確認
4．ソフトウェアのバージョンアップや新製品の案内等の情報提供
5．懸賞企画等で当選された方お客様への賞品の発送
　＊お客様の個人情報の取扱いに関するご意見、お問い合わせは http://www.canopus.co.jp/info/ までご連絡ください。

Power Movie PCI2 Development Kit
Programmer´s Manual
Version 1.0J
September 26,2005

目次

Chapter 0 - Preliminaries

Chapter 1 - Tutorial

Chapter 2 - Quick Reference

Chapter 3 - Generic Functions

Chapter 4 - Information Functions

Chapter 5 - Surface Functions

Chapter 6 – Video Decoder Functions

Chapter 7 – Video Encoder Functions

Chapter 8 – Video Stream Functions

APPENDIX

INDEX

Chapter 0

Preliminaries

開発キットの内容

Power Movie PCI2 Development Kit(以下開発キットと記載します）は、ビデオの映像をパソコンのディスプレイにオーバーレイ表示させたり、表示されている映像を静止画として取り込んだりする Power Movie PCI の機能をアプリケーションに提供するコンポーネントです。

本開発キットのオーバーレイ表示は２種類用意されていますので使用目的に応じて指定してください。

Super Overlay	:	グラフィックカードのオンスクリーンメモリに直接データを書き込むことでオーバーレイ表示します。 ★ウィンドウサイズは160×120～640×480ピクセルとなります。
DirectDraw Overlay	:	オフスクリーンメモリにデータを転送し、グラフィックカードの DirectDraw 機能を用いてオーバーレイ表示します。

この開発キットには以下に示すように、Power Movie PCI に準じる画像オーバーレイや静止画キャプチャ、ビデオ出力デバイス用のソフトウェア開発キット(MGEAPI）が含まれています。MGEAPI はインクルードファイル、ライブラリ、およびサンプルプログラムで構成されています。

(2005 年 8 月現在、MGEAPI に対応しているデバイスは Power Movie PCI のみです。)

Visual Basic 用コードモジュール

VB¥include¥movgear.bas

VB¥include¥mgeerror.bas

C/C++ 用インクルードファイル

VC¥include¥movgear.h

VC¥include¥mgeerror.h

C/C++ 用ライブラリファイル

VC¥lib¥mge.lib

開発キットのインストール

以下の手順に従って、開発キットのインストールを行ってください。

ドライバのインストールを行うには、システム設定の変更を行える権限を持つ ID（Administrator、コンピュータの管理者など）でログオンする必要があります。あらかじめシステム設定の変更を行える権限を持つ ID でログオンしてからインストール作業を行ってください。ここでは Windows XP 環境での手順を例に説明します。

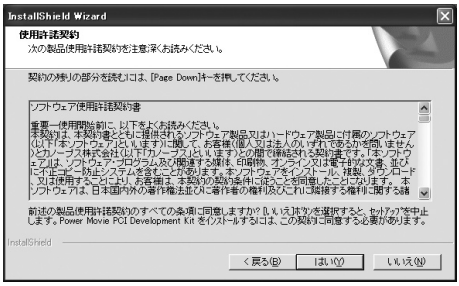
1 『Power Movie PCI2 Development Kit CD』を CD-ROM ドライブにセットします。

2 [スタート] メニューから [すべてのプログラム]（Windows 2000 環境の場合は [プログラム]）→ [アクセサリ] → [エクスプローラ] と進んで『エクスプローラ』を起動し、CD-ROM を挿入したドライブを選択して開き、SDK フォルダ内の [Setup.exe] をダブルクリックします。

3 [次へ] をクリックします。

4 使用許諾契約が表示されますので内容をよくお読みの上、同意される場合のみ[はい]をクリックしてください。使用許諾契約に同意されない場合は、[いいえ]をクリックし、インストール作業を中断して当社カスタマーサポートまで書面にてご連絡ください。

※使用許諾契約に同意されない場合は、本ソフトウェアはお使いいただけません。



5 インストール先のフォルダを選択し、[次へ]をクリックします。

※インストール先のフォルダを変更する「場合には、[参照]をクリックし、インストールするフォルダを選択するか、フォルダ名をキー入力します。

6 [完了]をクリックします。

Visual Basic でのアプリケーション開発

Visual Basic のプロジェクトに次のコードモジュールを追加してください。

movgear.bas（各種定義が書かれているファイルです。）
mgeerror.bas（エラーが定義されているファイルです。）

上記のコードモジュールでは、戻り値を持たない関数でも Function プロシージャとして宣言されています。

Visual C++ でのアプリケーション開発

●開発環境

プロジェクトで参照されるインクルードファイルおよびライブラリファイルのフォルダに以下のパスを追加してください。

（開発キットが組み込まれているフォルダを C:¥Program files¥Canopus¥Power Movie PCI Development kit¥ であると想定しています。）

インクルードファイル

C:¥Program files¥Canopus¥Power Movie PCI Development kit¥vc¥include

ライブラリ

C:¥Program files¥Canopus¥Power Movie PCI Development kit¥vc¥lib

●コンパイル

次のファイルをインクルードしてください。

movgear.h（各種定義が書かれているファイルです。）
mgeerror.h（エラーが定義されているファイルです。）

●リンク

次のライブラリをリンクしてください。

mge.lib

サンプルプログラム

開発キットにはこのドキュメントに記載されているファンクションの具体的な使用例としてVisual BasicおよびVisual C++に対応したサンプルが含まれています。作成しようとしているアプリケーションに組み込む機能に応じて各サンプルを参照してください。

サンプルプログラムの内容はC:¥Program files¥Canopus¥Power Movie PCI Development kit¥SAMPLES.TXTを参照してください。

（開発キットが組み込まれているフォルダをC:¥Program files¥Canopus¥Power Movie PCI Development kit¥であると想定しています。）

Overlay	**2画面Overlay**
Videoout	**Video出力プログラム**

Visual C++用のサンプルは C 言語で記述されています。

開発キットに含まれるソースコードの取り扱いについて

この開発キットに含まれるサンプルプログラムはドキュメント（本書）を補うための資料となっています。サンプルプログラムのソースコード（以下、「ソースコード」といいます）自体を改変したり、その一部をお客様のアプリケーションに組み込んで利用してください。ただし、お客様のアプリケーションとそこに組み込まれたソースコードの一部との適合性に関してサポートの範囲を限定させていただくこともございますのでご了承ください。

カノープス株式会社はソースコードの使用と変更に関して完全に自由な権利をお客様に許諾いたします。ただし、お客様の営利を目的としたソフトウェア製品にこれらのコードをソースコードもしくはこれを変更したものの形態のままで含めることはご遠慮願います。

また、最終的に作成されたアプリケーションの運用結果および目的への適合性につき、カノープス株式会社では一切の責任を負いかねますので予めご了承ください。

プログラム作成時の注意事項

この開発キットでプログラムを作成される場合は、次の事項にご注意ください。

1．本開発キットの対応言語は次の通りです。
   Visual Basic　　Version 4.0 以降 (32bit 版のみ)
   Visual C++　　Version 4.0 以降

2．本開発キットの開発環境対応 OS は次の通りです。
   Microsoft Windows® 2000 Professional 日本語版 +ServicePack4 以上
   Microsoft Windows® XP Home Edition 日本語版 +ServicePack2 以上
   Microsoft Windows® XP Professional 日本語版 +ServicePack2 以上
   ★作成されたプログラムを Windows 2000/XP 環境で動作させる場合の当社サポートは、提供製品の使用可能な範囲内のみとなります。

3．プログレッシブ JPEG でエンコードされたファイルは扱えません。

4．本開発キットを使用して複数枚の Power Movie PCI を制御することはできません。

5．複数のアプリケーションからの同時使用はできません。

6．MultiThread には対応していません。

7．ハードウェアの制限により、オーバーレイ表示の上部 2 ラインに同じ内容が繰り返し表示されます。

8．本開発キットは従来当社から販売しております Power Movie MP/V 開発キットおよび Power Movie 開発キットとの互換性はありません。

9．Unicode には、対応していません。

10. Power Movie PCI によるオーバーレイ表示中に、直接 Windows の画面モードを変更する操作を行ったり、画面モードを変更するユーティリティソフトウェアを使用しないでください。

11. 本開発キットに関するご質問は E-mail にて承っております。お電話、Fax でのお問い合わせについては受付できませんのであらかじめご了承ください。

インターネット E-mail SDK@canopus.co.jp

お問い合わせの際には発生現象と共に次の内容を必ず記載してください。

1．使用しているモジュールのタイムスタンプ
MGE.DLL　　/ PVJPEG.DLL / PVMJPEG.DLL
MGEHOOK.DLL / PAVAPI.DLL / PMPKRNL.SYS

2．使用しておられる開発環境
・Windows のバージョン
・コンパイラのメーカー、バージョン
・使用されているその他の開発キット

3．ハードウェア環境
・PC 本体メーカーおよび機種名
・周辺機器メーカーおよび型番

● PCI Express のビデオカードと組み合わせて使用した場合にオーバーレイ表示されない場合は Windows フォルダにある canopus.ini に下記の内容を追加してください。

```
[PowerMoviePCI.pmpnt]
OverlayMethodDD=1
OverlayMethodSO=1
```

●ビデオカードの組み合わせによりオーバーレイの内容が適切に更新されない場合があります。
Windows フォルダにある canopus.ini に下記の内容を追加してください。

```
[PowerMoviePCI.mge]
OverlayUpdateMethod=1
```

上記の設定でも問題が解決しない場合には下記の内容に変更してください。

```
[PowerMoviePCI.mge]
OverlayUpdateMethod=0
```

Chapter 1

Tutorial

1．ビデオ表示アプリケーションの作成

ビデオ表示アプリケーションでは、

Power Movie PCI の使用開始
ビデオウィンドウの生成
ビデオ入力の確認 ＊
ビデオ映像の表示開始
入力画像のフィールド選択 ＊
入力画質の調整 ＊
ウィンドウの移動およびサイズの変更
ビデオ映像の取り込みおよび静止 ＊
ビデオ映像の表示終了
ビデオウィンドウの破棄
Power Movie PCI の使用終了

などの機能が必要となります。＊は必須項目ではありません。必要に応じて組み込みください。

次ページから DirectDraw Overlay を使用し、ビデオ映像をウィンドウ内に表示するアプリケーションの作成について説明します。本文中ではＣ言語を用いて説明していますが、Visual Basic でも同様のシーケンスでコーディングを行います。また、エラーチェックは省いてあります。

Power Movie PCI の使用開始

Power Movie PCI を使用するアプリケーションは、その使用開始をドライバに許可してもらう必要があります。アプリケーションの初期化時に MGE_Initialize および MGE_SetCooperativeLevel ファンクションを呼び出して Power Movie PCI の使用を開始します。

```
HWND g_hWndApp;      // Applicationのメインウィンドウハンドル(グローバル変数)
HMGE g_hmge;         // Power Movie PCIの使用権を保持するハンドル(グローバル変数)
MGERESULT mr;

    mr = MGE_Initialize(0, g_hWndApp, &g_hmge);
    if(mr != MGE_S_OK) {
        // Power Movie PCIが使用できない

        エラー処理
    }
    mr = MGE_SetCooperativeLevel(g_hmge, MGE_SCL_EXCLUSIVE);
    if(mr != MGE_S_OK) {
        // Power Movie PCIが使用できない

        エラー処理
    }
```

ビデオウィンドウの生成

アプリケーションのウィンドウ生成終了後（CreateWindow 実行後）、クライアント領域を Video Window として使用します。
ただし、DirectDraw Overlay の制限に触れるような場合は事前にウィンドウのサイズを変更しておく必要があります。

```
HMGEVSRC    g_hmgevsDDraw;          // Video Decoderのハンドル(グローバル変数)
HMGESURFACE g_hmgesrfSurfaceDDraw; // DDraw Overlay Surfaceのハンドル(グローバル変数)
HMGESTREAM  g_hmgestmStreamDDraw;  // DDraw Overlay Streamのハンドル(グローバル変数)

    RECT     rcCrop, rcFrame, rcWindow, rcView;
    RECT     rcClient;
    DWORD    dwWindowWidth;
    DWORD    dwViewWidth;
    DWORD    dwWindowHeight;
    DWORD    dwViewHeight;
    DWORD    dwCropWidth;
    DWORD    dwCropHeight;

        // 現在のClient領域の大きさを取得
    GetClientRect(g_hWndApp, &rcClient);
    ClientToScreen(g_hWndApp, (LPPOINT)&rcClient);
    ClientToScreen(g_hWndApp, (LPPOINT)&rcClient + 1);

        // Crop用の領域初期化
    rcCrop.left = 0;
    rcCrop.top = 0;
    rcCrop.right = 640;
    rcCrop.bottom = 480;

        // Frame用の領域初期化
    rcFrame.left = 0;
    rcFrame.top = 0;
    rcFrame.right = 640;
    rcFrame.bottom = 480;

        // View用の領域初期化
    rcView.left = 0;
    rcView.top = 0;
    rcView.right = 640;
    rcView.bottom = 480;

        // Window用の領域初期化, Client領域を使用
    rcWindow = rcClient;
```

```
    // DirectDraw Overlayの制限を回避するために
    // 各矩形を補正する。
dwWindowWidth = rcWindow.right - rcWindow.left;
dwViewWidth = rcView.right - rcView.left;
dwCropWidth = rcCrop.right - rcCrop.left;

while(dwViewWidth > dwWindowWidth) {
    dwViewWidth /= 2;
    if(dwViewWidth * 4 < dwCropWidth) {
        /* この状況では実現できない */
        return;
    }
}
rcView.right = rcView.left + dwViewWidth;

dwWindowHeight = rcWindow.bottom - rcWindow.top;
dwViewHeight = rcView.bottom - rcView.top;
dwCropHeight = rcCrop.bottom - rcCrop.top;

while(dwViewHeight > dwWindowHeight) {
    dwViewHeight = dwViewHeight / 2;
    if(dwViewHeight * 8 < dwCropHeight) {
        /* この状況では実現できない */
        return;
    }
}
rcView.bottom = rcView.top + dwViewHeight;

//
// DirectDraw Overlay
//
MGE_CreateVideoDecoder(g_hmge, MGEChannel2, &g_hmgevsDDraw);
MGE_CreateVideoInStream(g_hmgevsDDraw, &g_hmgestmStreamDDraw);
MGE_CreateDDrawOvlSurface(g_hmge, &g_hmgesrfSurfaceDDraw, g_hWndApp, 640, 480);
MGE_VStreamConnect(g_hmgestmStreamDDraw, g_hmgesrfSurfaceDDraw);
MGE_VStreamSetRect(g_hmgestmStreamDDraw, MGEStreamRectCrop, &rcCrop, FALSE);
MGE_VStreamSetRect(g_hmgestmStreamDDraw, MGEStreamRectFrame, &rcFrame, FALSE);
MGE_VStreamSetRect(g_hmgestmStreamDDraw, MGEStreamRectView, &rcView, FALSE);
MGE_VStreamSetRect(g_hmgestmStreamDDraw, MGEStreamRectWindow, &rcWindow, TRUE);
```

ビデオ入力の確認

ビデオ入力が正しく行われているかどうかを確認します。
本サンプルプログラムでは入力されていない場合に確認を促すメッセージを表示するのみで、再度の確認は行いません。

```
MGELINE      mgl;
MGEVDSTATUS  mgs;

MGE_VDGetLine(g_hmgevsDDraw, &mgl);
mgs.dwSize = sizeof(mgs);
MGE_VDGetStatus(g_hmgevsDDraw, mgl, &mgs);
if((mgs.dwSignalState & MGE_SS_M_STABLE) == MGE_SS_NOTSTABLE)
    MessageBox(NULL, "ビデオが入力されていません", "注意", MB_OK);
```

ビデオ映像の表示開始

ビデオ映像をアプリケーションウィンドウのクライアント領域に表示させます。他のウィンドウが上に重なった場合に、オーバーレイされるビデオ映像がそのウィンドウの上に表示されないようにキーカラー（Color Key）を有効にします。

キーカラーを有効にするためには、ウィンドウのクライアント領域をキーカラーで塗りつぶす必要があります。これはウィンドウのバックグラウンドカラーをキーカラーと同じ色にすることで簡単に実現できます。

ビデオオーバーレイの表示開始後にウィンドウ内のキーカラーで塗りつぶすようにすれば、キーカラーそのものの色を見えないようにすることもできます。

```
MGECOLORKEY   ck;
HBRUSH        hBrush;

    // キーカラーをマゼンタにして表示を開始する
ck.dwColorLow = 0x00ff00ff;    // マゼンタ
ck.dwColorHigh = 0x00ff00ff;

MGE_SurfaceSetColorKey(g_hmgesrfSurfaceDDraw, &ck);
MGE_VStreamStart(g_hmgestmStreamDDraw);
MGE_SurfaceStartOverlay(g_hmgesrfSurfaceDDraw);

hBrush = CreateSolidBrush(RGB(255, 0, 255));
SetClassLong(g_hWndApp, GCL_HBRBACKGROUND, (LONG)hBrush);
InvalidateRect(g_hWndApp, NULL, TRUE);
UpdateWindow(g_hWndApp);
```

入力画像のフィールド選択

入力画像のフィールドを選択します。

偶数フィールドを選択する場合

```
MGE_VStreamSetVideoMode(g_hmgestmStreamDDraw, MGEVideoModeEvenField, TRUE);
```

奇数フィールドを選択する場合

```
MGE_VStreamSetVideoMode(g_hmgestmStreamDDraw, MGEVideoModeOddField, TRUE);
```

両フィールド（フレーム）を選択する場合

```
MGE_VStreamSetVideoMode(g_hmgestmStreamDDraw, MGEVideoModeFrame, TRUE);
```

入力画質の調整

明るさなど入力画質の調整を行います。

下記のコード片では現在入力されている明るさに 10 を加えています。
新しい明るさがシステムに対する設定可能値を超えている場合は、設定可能な最小の値を使用します。
このコード片自体には意味はありませんが、この応用によりアプリケーションから入力画質の調整ダイアログなどを作成することができます。

```
MGELINE  mgl;
DWORD    dwMin, dwMax, dwDefault, dwVal;

MGE_VDGetLine(g_hmgevsDDraw, &mgl);
MGE_VDGetParamRange(g_hmgevsDDraw, mgl, MGEVideoAmpBrightness,
                &dwMin, &dwMax, &dwDefault);
MGE_VDGetParam(g_hmgevsDDraw, mgl, MGEVideoAmpBrightness, &dwVal);
dwVal += 10;
if(dwVal >= dwMax)
    dwVal = dwMin;
MGE_VDSetParam(g_hmgevsDDraw, mgl, MGEVideoAmpBrightness, dwVal);
```

また、アプリケーション中に " 規定値に戻す " ボタンなどを実装する場合は、下記の様に MGE_VDGetParamRange で返されるデフォルト値を使用することで実現できます。

```
MGELINE  mgl;
DWORD    dwMin, dwMax, dwDefault;

MGE_VDGetLine(g_hmgevsDDraw, &mgl);
MGE_VDGetParamRange(g_hmgevsDDraw, mgl, MGEVideoAmpBrightness,
                &dwMin, &dwMax, &dwDefault);
MGE_VDSetParam(g_hmgevsDDraw, mgl, MGEVideoAmpBrightness, dwDefault);
```

ウィンドウの移動およびサイズ変更

ユーザーの操作により、アプリケーションのウィンドウを移動させた時や、ウィンドウのサイズを変更した場合には、ビデオウィンドウを新しい位置に移動させたり、サイズを変更します。

```
case WM_WINDOWPOSCHANGED:
    {
    RECT        rcWindow;
    RECT        rcView;
    RECT        rcCrop;
    DWORD    dwWindowWidth;
    DWORD    dwViewWidth;
    DWORD    dwWindowHeight;
    DWORD    dwViewHeight;
    DWORD    dwCropWidth;
    DWORD    dwCropHeight;

    GetClientRect(g_hWndApp, &rcWindow);
    ClientToScreen(g_hWndApp, (LPPOINT)&rcWindow);
    ClientToScreen(g_hWndApp, (LPPOINT)&rcWindow + 1);

    MGE_VStreamGetRect(g_hmgestmStreamDDraw, MGEStreamRectCrop, &rcCrop);
    rcView = rcCrop;

        // DirectDraw Overlayの制限を回避するために
        // 各矩形を補正する。
    dwWindowWidth = rcWindow.right - rcWindow.left;
    dwViewWidth = rcView.right - rcView.left;
    dwCropWidth = rcCrop.right - rcCrop.left;

    while(dwViewWidth > dwWindowWidth) {
        dwViewWidth /= 2;
        if(dwViewWidth * 4 < dwCropWidth) {
            /* この状況では実現できない */
            goto do_other_things;
        }
    }
    rcView.right = rcView.left + dwViewWidth;
```

```
dwWindowHeight = rcWindow.bottom - rcWindow.top;
dwViewHeight = rcView.bottom - rcView.top;
dwCropHeight = rcCrop.bottom - rcCrop.top;

while(dwViewHeight 〉 dwWindowHeight) {
    dwViewHeight = dwViewHeight / 2;
    if(dwViewHeight * 8 〈 dwCropHeight) {
    if(dwViewWidth * 4 〈 dwCropWidth) {
        /* この状況では実現できない */
        goto do_other_things;
    }
}
rcView.bottom = rcView.top + dwViewHeight;

    // View と Frame は同じ
MGE_VStreamSetRect(g_hmgestmStreamDDraw, MGEStreamRectFrame, &rcView, FALSE);
MGE_VStreamSetRect(g_hmgestmStreamDDraw, MGEStreamRectView, &rcView, FALSE);
MGE_VStreamSetRect(g_hmgestmStreamDDraw, MGEStreamRectWindow, &rcWindow, TRUE);
}
do_other_things:;
```

ビデオ映像の取り込みおよび静止

ビデオ映像の取り込みおよび静止を行ないます。
この例では現在の状態を取得して、取り込みと静止を交互に切り替えます。

```
MGESTREAMSTATE ss;

MGE_VStreamGetState(g_hmgestmStreamDDraw, &ss);
if(ss == MGEStreamStatePause) {
    MGE_VStreamRestart(g_hmgestmStreamDDraw);
}
else if(ss == MGEStreamStateStarted) {
    MGE_VStreamPause(g_hmgestmStreamDDraw);
}
else {
    // ビデオがスタートしていない。
}
```

ビデオ映像の表示終了

ビデオ映像の表示を終了させます。

ビデオオーバーレイの表示終了前にウィンドウ内を適当な色で塗りつぶすようにすれば、キーカラーそのものの色を見えないようにすることができます。

```
HBRUSH    hBrush;

    // バックグラウンドカラーを黒にする
hBrush = GetStockObject(BLACK_BRUSH);
hBrush = (HBRUSH)SetClassLong(g_hWndApp, GCL_HBRBACKGROUND, (LONG)hBrush);
InvalidateRect(g_hWndApp, NULL, TRUE);
UpdateWindow(g_hWndApp);

    // 表示を終了させる。
MGE_SurfaceStopOverlay(g_hmgesrfSurfaceDDraw);
MGE_VStreamStop(g_hmgestmStreamDDraw);

    // キーカラーのブラシを削除する
if(hBrush)
    DeleteObject(hBrush);
```

ビデオウィンドウの破棄

ビデオウィンドウを破棄します。WM_DESTROY などで実行します。

```
MGE_VStreamDisconnect(g_hmgestmStreamDDraw, g_hmgesrfSurfaceDDraw);
MGE_DestroyVideoStream(g_hmgestmStreamDDraw);
MGE_DestroySurface(g_hmgesrfSurfaceDDraw);
MGE_DestroyVideoDecoder(g_hmgevsDDraw);
```

Power Movie PCI の使用終了

Power Movie PCI を使用するアプリケーションは、その使用終了をドライバに通知する必要があります。アプリケーションの終了時などに MGE_Uninitialize ファンクションを呼び出して Power Movie PCI の使用を終了します。

```
MGE_Uninitialize(g_hmge);
```

2．ビデオ出力アプリケーションの作成

ビデオ出力アプリケーションでは

Power Movie PCI の使用開始
ビデオへの出力の準備
ビデオへ出力するものを描画する
ビデオへの出力の開始
ビデオへの出力を切り替える　＊
ビデオへの出力の終了
ビデオへの出力の後処理
Power Movie PCI の使用終了

などの機能が必要となります。＊は必須項目ではありません。必要に応じて組み込みください。

次ページから静止画像をビデオ出力するアプリケーションの作成について説明します。本文中ではＣ言語を用いて説明していますが、Visual Basic でも同様のシーケンスでコーディングを行います。
また、エラーチェックは省いてあります。

Power Movie PCI の使用開始

Power Movie PCI を使用するアプリケーションは、その使用開始をドライバに許可してもらう必要があります。アプリケーションの初期化時に MGE_Initialize および MGE_SetCooperativeLevel ファンクションを呼び出して Power Movie PCI の使用を開始します。

```
HWND g_hWndApp;      // Applicationのメインウインドウハンドル(グローバル変数)
HMGE g_hmge;         // Power Movie PCIの使用権を保持するハンドル(グローバル変数)
MGERESULT mr;

    mr = MGE_Initialize(0, g_hWndApp, &g_hmge);
    if(mr != MGE_S_OK) {
        // Power Movie PCIが使用できない

        エラー処理
    }
    mr = MGE_SetCooperativeLevel(g_hmge, MGE_SCL_EXCLUSIVE);
    if(mr != MGE_S_OK) {
        // Power Movie PCIが使用できない

        エラー処理
    }
```

ビデオへの出力の準備

静止画像をビデオ出力する準備を行ないます。

```
HMGEVDST      g_hmgevdVideoOut;         // Video Encoderへのハンドル(グローバル変数)
HMGESTREAM    g_hmgestmStreamVideoOut;  // Video out Streamへのハンドル(グローバル変数)
HMGESURFACE   g_hmgesrfSurfaceMemory;   // Memory Surfaceへのハンドル(グローバル変数)

    RECT        rcWindow, rcView;
    MGEPFINFO    pfinfo;

    rcView.left = 0;
    rcView.top = 0;
    rcView.right = 640;
    rcView.bottom = 480;

    rcWindow.left = 0;
    rcWindow.top = 0;
    rcWindow.right = 640;
    rcWindow.bottom = 480;

        //
        // Video outするための各オブジェクトの生成
        //
    MGE_CreateVideoEncoder(g_hmge, MGEChannel1, &g_hmgevdVideoOut);
    MGE_CreateVideoOutStream(g_hmgevdVideoOut, &g_hmgestmStreamVideoOut);
    pfinfo.dwSize = sizeof(pfinfo);
    MGE_GetPixelFormatInfo(MGEPixelFormatYUY2, &pfinfo);
    MGE_CreateMemorySurface(g_hmge, &g_hmgesrfSurfaceMemory,
                        640, 480, 640 * pfinfo.dwBytesPerPixel,
                                MGEPixelFormatYUY2);
    MGE_VStreamConnect(g_hmgestmStreamVideoOut, g_hmgesrfSurfaceMemory);
    MGE_VStreamSetRect(g_hmgestmStreamVideoOut,
                        MGEStreamRectView, &rcView, FALSE);
    MGE_VStreamSetRect(g_hmgestmStreamVideoOut,
                        MGEStreamRectWindow, &rcWindow, TRUE);
```

ビデオへ出力するものを描画する

ビデオへ出力するものを用意します。ここで MGE_SurfaceFillBuffer、MGE_SurfaceSetBitmapBits、 MGE_SurfaceLoadDIB や MGE_SurfaceLoadJPEG などを使用して、オブジェクト（静止画）をサーフェスに描画します。
ここでは MGE_SurfaceFillBuffer を使用します。

```
DWORD   s_ardwColor[] = {
        0x00ffffff,
        0x00ff0000,
        0x0000ff00,
        0x000000ff,
        0x00000000,
        0x00ffff00,
};
RECT  rc;
int   i;

rc.left = 0;
rc.top = 0;
rc.right = 640;
rc.bottom = 480;

for(i = 0;i < 6;i++) {
    MGE_SurfaceFillBuffer(g_hmgesrfSurfaceMemory, &rc, s_ardwColor[i]);
    rc.left += 40;
    rc.right -= 40;
    rc.top += 30;
    rc.bottom -= 30;
}
```

ビデオへの出力の開始

ビデオ出力を開始します。

```
MGE_VStreamStart(g_hmgestmStreamVideoOut);
```

ビデオへの出力を切り替える

Memory サーフェースを直接書き換えることでビデオの出力を変更することは可能ですが、この場合書き換えを行っている途中の絵が出力されてしまいます。

本サンプルプログラムでは別の Memory サーフェースを用意してビデオへの出力を切り替えています。

```
HMGESURFACE    g_hmgesrfSurfaceMemory2; // 切り替えのMemory Surfaceへのハンドル(グローバル変数)

    MGEPFINFO  pfinfo;
    RECT       rc;

        //
        // 切り替えのMemory Surfaceを作成します。
        //
    pfinfo.dwSize = sizeof(pfinfo);
    MGE_GetPixelFormatInfo(MGEPixelFormatYUY2, &pfinfo);
    MGE_CreateMemorySurface(g_hmge, &g_hmgesrfSurfaceMemory2,
                    640, 480, 640 * pfinfo.dwBytesPerPixel,
                            MGEPixelFormatYUY2);

        //
        // このメモリにデータを格納します。
        //

        //
        // 事前に黒で塗っておきます。
        // (JPEGのデータが640x480で在れば必要ありません)
        //
    rc.left = 0;
    rc.top = 0;
    rc.right = 640;
    rc.bottom = 480;
    MGE_SurfaceFillBuffer(g_hmgesrfSurfaceMemory2, &rc, 0);

        //
        // データをJPEGファイルからロードします。
        //
    rc.left = 0;
    rc.top = 0;
    rc.right = 640;
    rc.bottom = 400;
    MGE_SurfaceLoadJPEG(g_hmgesrfSurfaceMemory2, "pic.jpg", &rc);
```

```
    //
    // Video出力を切り替えます。
    //
MGE_VStreamReplace(g_hmgestmStreamVideoOut, g_hmgesrfSurfaceMemory2);

    //
    // 何らかの処理をする。
    // 例えば所定の時間だけ待つ
    //       :
    //       :
    //       :
    //

    //
    // Videoの出力を元に戻します。
    //
MGE_VStreamReplace(g_hmgestmStreamVideoOut, g_hmgesrfSurfaceMemory);

    //
    // 作成したMemory surfaceを削除します。
    //
MGE_DestroySurface(g_hmgesrfSurfaceMemory2);
```

ビデオへの出力の終了

ビデオ出力を終了します。

```
MGE_VStreamStop(g_hmgestmStreamVideoOut);
```

ビデオへの出力の後処理

ビデオ出力を行なったオブジェクト（静止画）を削除する。

```
MGE_VStreamDisconnect(g_hmgestmStreamVideoOut, g_hmgesrfSurfaceMemory);
MGE_DestroyVideoStream(g_hmgestmStreamVideoOut);
MGE_DestroySurface(g_hmgesrfSurfaceMemory);
MGE_DestroyVideoEncoder(g_hmgevdVideoOut);
```

Power Movie PCI の使用終了

Power Movie PCI を使用するアプリケーションは、その使用終了をドライバに通知する必要があります。アプリケーションの終了時などに MGE_Uninitialize ファンクションを呼び出して Power Movie PCI の使用を終了します。

```
MGE_Uninitialize(g_hmge);
```

Chapter 2

Quick Reference

MGE API Quick Reference

§ 1.Generic Functions(初期化)

MGE_Initialize	ライブラリの初期化
MGE_Uninitialize	ライブラリの終了
MGE_SetCooperativeLevel	ライブラリの使用形態を指定する
MGE_SaveConfiguration	ライブラリの設定を registry に格納する
MGE_LoadConfiguration	ライブラリの設定を registry からロードする

§ 2.Generic Functions (全体)

MGE_Update	パラメータを反映させる

§ 3.Information Functions

MGE_GetDeviceInfo	ハードウェアの情報を得る
MGE_SetVideoStandard	システムのビデオスタンダード（ビデオ方式）を設定する
MGE_GetVideoStandard	現在のシステムのビデオスタンダードを取得する
MGE_GetPixelFormatInfo	ピクセルフォーマットの情報を取得する
MGE_GetImageInfo	イメージデータの情報を得る

§ 4.Surface Functions

MGE_CreateSuperOvlSurface	Super Overlay のサーフェスを作成する
MGE_CreateDDrawOvlSurface	DirectDraw Overlay のサーフェスを作成する
MGE_CreateMemorySurface	サーフェスをメモリ上に作成する
MGE_DestroySurface	サーフェスを破棄する
MGE_SurfaceSetColorKey	キーカラーを設定する
MGE_SurfaceGetColorKey	現在のキーカラーを取得する
MGE_SurfaceSetOverlayMode	オーバーレイモードを設定する
MGE_SurfaceGetOverlayMode	現在のオーバーレイモードを取得する
MGE_SurfaceStartOverlay	オーバーレイを開始する
MGE_SurfaceStopOverlay	オーバーレイを停止する
MGE_SurfaceSetBitmapBits	メモリからサーフェスへ bitmap をコピーする

MGE_SurfaceGetBitmapBits	サーフェスからメモリへ bitmap をコピーする
MGE_SurfaceFillBuffer	サーフェスを指定した色で塗る
MGE_SurfaceLoadDIB	サーフェスに DIB をロードする
MGE_SurfaceSaveDIB	サーフェスのデータを DIB としてファイルにセーブする
MGE_SurfaceLoadJPEG	サーフェスに JPEG データを展開してロードする
MGE_SurfaceSaveJPEG	サーフェスのデータを JPEG に圧縮してファイルにセーブする

§ 5. Video Decoder Functions

MGE_CreateVideoDecoder	ビデオデコーダオブジェクトを作成する
MGE_DestroyVideoDecoder	ビデオデコーダオブジェクトを破棄する
MGE_VDGetStatus	ビデオデコーダの状態を取得する
MGE_VDSetParam	ビデオデコーダのパラメータを調整する
MGE_VDGetParam	現在のビデオデコーダのパラメータを取得する
MGE_VDGetParamRange	ビデオデコーダの調整可能パラメータの範囲を取得する
MGE_VDSetLine	ビデオデコーダへの入力を選択する
MGE_VDGetLine	現在のビデオデコーダへの入力を取得する
MGE_VDSaveConfiguration	ビデオデコーダの設定を registry に格納する
MGE_VDLoadConfiguration	ビデオデコーダの設定を registry からロードする

§ 6. Video Encoder Functions

MGE_CreateVideoEncoder	ビデオエンコーダオブジェクトを作成する
MGE_DestroyVideoEncoder	ビデオエンコーダオブジェクトを破棄する
MGE_VESetLine	ビデオエンコーダの出力を選択する
MGE_VEGetLine	現在のビデオエンコーダの出力ライン番号を取得する
MGE_VESaveConfiguration	ビデオエンコーダの設定を registry に格納する
MGE_VELoadConfiguration	ビデオエンコーダの設定を registry からロードする

§ 7. Video Stream Functions

MGE_CreateVideoInStream	入力用のビデオストリームを作成する
MGE_CreateVideoOutStream	出力用のビデオストリームを作成する
MGE_DestroyVideoStream	ビデオストリームを破棄する
MGE_VStreamConnect	ビデオストリームをサーフェスに結び付ける
MGE_VStreamDisconnect	ビデオストリームとサーフェスを切り離す
MGE_VStreamReplace	ビデオストリームのサーフェスを付け替える
MGE_VStreamSetVideoMode	ビデオストリーム上のビデオ信号モードを設定する
MGE_VStreamGetVideoMode	現在のビデオストリーム上のビデオ信号モードを取得する
MGE_VStreamSetRect	各矩形を指定する
MGE_VStreamGetRect	現在の矩形情報を取得する
MGE_VStreamGetRectRange	各矩形情報を取得する
MGE_VStreamStart	ストリームを開始する
MGE_VStreamPause	ストリームを一時停止させる
MGE_VStreamRestart	ストリームを再開させる
MGE_VStreamStop	ストリームを停止させる
MGE_VStreamGetState	指定したストリームの状態を取得する
MGE_VStreamSetEffect	ストリームに特殊効果を付加する
MGE_VStreamGetEffect	ストリーム上の現在の特殊効果を取得する

Chapter 3

Generic Functions

MGE_Initialize

ライブラリの初期化を行う。

書式

〈C/C++〉

```
MGEAPI MGERESULT WINAPI
MGE_Initialize(DWORD dwReserved, HWND hWnd, LPHMGE lphmge);
```

〈Basic〉

```
Declare Function MGE_Initialize Lib "mge.dll" (ByVal dwReserved As Long, ByVal hWnd As
Long, lphmge As Long) As Long
```

引数

dwReserved	予約0をわたす
hWnd	ApplicationのWindow Handle MGE_Uninitializeを呼ぶまで *hWnd* が有効でなければならない
lphmge	ライブラリのハンドルを格納するアドレス

戻り値

正常に終了した場合、MGE_S_OKを返す。それ以外の場合はエラーコードを返す。

MGE_Uninitialize

ライブラリを終了する。

書式

〈C/C++〉

```
MGEAPI MGERESULT WINAPI
MGE_Uninitialize(HMGE hmge);
```

〈Basic〉

```
Declare Function MGE_Uninitialize Lib "mge.dll" (ByVal hmge As Long) As Long
```

引数

hmge	MGE_Initializeで返されたMGEライブラリのハンドル

戻り値

正常に終了した場合、MGE_S_OKを返す。それ以外の場合はエラーコードを返す。

MGE_SetCooperativeLevel

ライブラリの使用形態を指定する。

書式

〈C/C++〉

```
MGEAPI MGERESULT WINAPI
MGE_SetCooperativeLevel(HMGE hmge, DWORD dwLevel);
```

〈Basic〉

```
Declare Function MGE_SetCooperativeLevel Lib "mge.dll" (ByVal hmge As Long, ByVal dwLevel
As Long) As Long
```

引数

hmge	MGE_Initializeで返されたMGEライブラリのハンドル
dwLevel	現在はMGE_SCL_EXCLUSIVEを指定すること

戻り値

正常に終了した場合、MGE_S_OKを返す。それ以外の場合はエラーコードを返す。

補足

*この関数でエラーが返ってきた場合には、MGE_Uninitialize以外のライブラリは使用できない。
*エラーコード MGE_E_BUSY が返ってきた場合には他のクライアントがライブラリを使用しているので、MGE_Uninitialiを呼び出して、終了しなければならない。

MGE_SaveConfiguration

ライブラリの設定を registry に格納する。

書式

〈C/C++〉

```
MGEAPI MGERESULT WINAPI
MGE_SaveConfiguration(HMGE hmge, HKEY hKey, LPCSTR lpszBase);
```

〈Basic〉

```
Declare Function MGE_SaveConfiguration Lib "mge.dll" (ByVal hmge As Long, ByVal hKey As
Long, ByVal lpszBase As String) As Long
```

引数

hmge MGE_Initializeで返されたMGEライブラリのハンドル

hKey registryのroot key
HKEY_CURRENT_USERなど

lpszBase registryのkey、実際の値はこのkeyの下に
以下の構造で作成される。

```
├─ VD¥
│   ├─ Channel1¥
│   │   ├─ Line1¥
│   │   │   └─ NTSC¥
│   │   │       ├─ Brightness=〈brightness value〉
│   │   │       ├─ Hue=〈hue value〉
│   │   │       ├─ Contrast=〈contrast value〉
│   │   │       ├─ Saturation=〈saturation value〉
│   │   │       └─ Sharpness=〈sharpness value〉
│   │   └─ CurrentLine = 〈current line number〉 1..4
│   └─ Channel2¥
│   .
│   .
└─ VE¥ ← 今は何も書かれない
```

戻り値

正常に終了した場合、MGE_S_OKを返す。それ以外の場合はエラーコードを返す。

MGE_LoadConfiguration

ライブラリの設定を registry からロードする。

書式

〈C/C++〉

```
MGEAPI MGERESULT WINAPI
MGE_LoadConfiguration(HMGE hmge, HKEY hKey, LPCSTR lpszBase);
```

〈Basic〉

```
Declare Function MGE_LoadConfiguration Lib "mge.dll" (ByVal hmge As Long, ByVal hKey As
Long, ByVal lpszBase As String) As Long
```

引数

hmge	MGE_Initializeで返されたMGEライブラリのハンドル
hKey	registryのroot key HKEY_CURRENT_USERなど
lpszBase	registryのkey (MGE_SaveConfiguration参照)

戻り値

正常に終了した場合、MGE_S_OKを返す。それ以外の場合はエラーコードを返す。

MGE_Update

パラメータを反映させる。

書式

〈C/C++〉

```
MGEAPI MGERESULT WINAPI
MGE_Update(HMGE hmge);
```

〈Basic〉

```
Declare Function MGE_Update Lib "mge.dll" (ByVal hmge As Long) As Long
```

引数

hmge	MGE_Initializeで返されたMGEライブラリのハンドル

戻り値

正常に終了した場合、MGE_S_OKを返す。それ以外の場合はエラーコードを返す。

Chapter 4

Information Functions

MGE_GetDeviceInfo

ハードウェアの情報を得る。

書式

〈C/C++〉

```
MGEAPI MGERESULT WINAPI
MGE_GetDeviceInfo(HMGE hmge, LPMGEDEVICEINFO lpmgedi);
```

〈Basic〉

```
Declare Function MGE_GetDeviceInfo Lib "mge.dll" (ByVal hmge As Long, lpmgedi As Long) As Long
```

引数

hmge	MGE_Initializeで返されたMGEライブラリのハンドル
lpmgedi	Device Informationを格納する領域

戻り値

正常に終了した場合、MGE_S_OKを返す。それ以外の場合はエラーコードを返す。

補足

*Power Movie PCIでは下記の値が返ります。

dwNumVideoDecoderChannel	2
dwNumVideoDecoderLine	4
dwNumVideoEncoderChannel	1
dwNumVideoEncoderLine	1
szBoardName	"Power Movie PCI"

MGE_SetVideoStandard

システムのビデオスタンダード（ビデオ方式）を設定する。

書式

〈C/C++〉

MGEAPI MGERESULT WINAPI
MGE_SetVideoStandard(HMGE *hmge*, MGEVIDEOSTANDARD *mgevs*);

〈Basic〉

Declare Function MGE_SetVideoStandard Lib "mge.dll" (ByVal *hmge* As Long, ByVal *mgevs* As Long) As Long

引数

hmge　MGE_Initializeで返されたMGEライブラリのハンドル

mgevs　ビデオスタンダード

MGEVideoStandardNTSC ･･･ NTSC方式

MGEVideoStandardPAL ･･･ PAL方式

※Power Movie PCIではNTSC方式のみをサポート

戻り値

正常に終了した場合、MGE_S_OKを返す。それ以外の場合はエラーコードを返す。

補足

*Power Movie PCIはPALは非サポートなので、この関数を使用する必要はありません。

MGE_GetVideoStandard

現在のシステムのビデオスタンダード（ビデオ方式）を取得する。

書式

〈C/C++〉

```
MGEAPI MGERESULT WINAPI
MGE_GetVideoStandard(HMGE hmge, LPMGEVIDEOSTANDARD lpmgevs);
```

〈Basic〉

```
Declare Function MGE_GetVideoStandard Lib "mge.dll" (ByVal hmge As Long, lpmgevs As Long)
As Long
```

引数

hmge　　MGE_Initializeで返されたMGEライブラリのハンドル

lpmgevs　　ビデオスタンダードを格納するアドレス

MGEVideoStandardNTSC ･･･ NTSC方式

MGEVideoStandardPAL ･･･ PAL方式

※Power Movie PCIではNTSC方式のみをサポート

戻り値

正常に終了した場合、MGE_S_OKを返す。それ以外の場合はエラーコードを返す。

補足

*Power Movie PCIではPALは非サポートなので、常にMGEVideoStandardNTSCが返ります。

MGE_GetPixelFormatInfo

ピクセルフォーマットの情報を取得する。

書式

〈C/C++〉

```
MGEAPI MGERESULT WINAPI
MGE_GetPixelFormatInfo(MGEPIXELFORMAT mgepfFormat, LPMGEPFINFO lpmgpfi);
```

〈Basic〉

```
Declare Function MGE_GetPixelFormatInfo Lib "mge.dll" (ByVal mgepfFormat As Long, lpmgfi As
MGEPFINFO) As Long
```

引数

mgepfFormat	情報を取得するPixel Format
lpmgpfi	Pixel Format情報を返すアドレス lpmgpfi->dwSizeは初期化されていなければならない

戻り値

正常に終了した場合、MGE_S_OKを返す。それ以外の場合はエラーコードを返す。

補足

*lpmgpf－〉rgbMaskはdwFourCCがBI_BITFIELDSの時にのみ意味を持つ。

MGE_GetImageInfo

イメージデータの情報を取得する。

書式

〈C/C++〉

```
MGEAPI MGERESULT WINAPI
MGE_GetImageInfo(LPCSTR lpszImageFile, LPMGEIMAGEINFO lpmgii);
```

〈Basic〉

```
Declare Function MGE_GetImageInfo Lib "mge.dll"Alias "MGE_GetImageInfoA"(ByVal lpszImageFile
As String, lpmgii As MGEIMAGEINFO)As Long
```

引数

lpszImageFile	情報を取得するイメージのファイル名
lpmgii	Image Info情報を返すアドレス lpmgii->dwSizeは初期化されていなければならない

戻り値

正常に終了した場合、MGE_S_OKを返す。それ以外の場合はエラーコードを返す。

補足

*取得できるのはDIBファイルまたはJPEGファイルのみです。

Chapter 5

Surface Functions

MGE_CreateSuperOvlSurface

Super Overlayのサーフェスを作成する。

書式

〈C/C++〉

```
MGEAPI MGERESULT WINAPI
MGE_CreateSuperOvlSurface(HMGE hmge, LPHMGESURFACE lphmgesrf, HWND hWnd,
                          DWORD dwWidth, DWORD dwHeight);
```

〈Basic〉

```
Declare Function MGE_CreateSuperOvlSurface Lib "mge.dll" (ByVal hmge As Long, lphmgesrf As
Long, ByVal hWnd As Long, ByVal dwWidth As Long, ByVal dwHeight As Long) As Long
```

引数

hmge	MGE_Initializeで返されたMGEライブラリのハンドル
lphmgesrf	サーフェスハンドルを格納するメモリアドレス
hWnd	OverlayするWindowハンドル このWindowはOverlayを終了するまで有効でなければならない このhWndのClient領域全体にoverlayされる
dwWidth	Overlayサーフェスの幅(pixel単位)
dwHeight	Overlayサーフェスの高さ(pixel単位)

戻り値

正常に終了した場合、MGE_S_OKを返す。それ以外の場合はエラーコードを返す。

MGE_CreateDDrawOvlSurface

DirectDraw Overlayのサーフェスを作成する。

書式

〈C/C++〉

```
MGEAPI MGERESULT WINAPI
MGE_CreateDDrawOvlSurface(HMGE hmge, LPHMGESURFACE lphmgesrf, HWND hWnd,
                          DWORD dwWidth, DWORD dwHeight);
```

〈Basic〉

```
Declare Function MGE_CreateDDrawOvlSurface Lib "mge.dll" (ByVal hmge As Long, lphmgesrf As
Long, ByVal hWnd As Long, ByVal dwWidth As Long, ByVal dwHeight As Long) As Long
```

引数

hmge	MGE_Initializeで返されたMGEライブラリのハンドル
lphmgesrf	サーフェスハンドルを格納するメモリアドレス
hWnd	OverlayするWindowハンドル このWindowはOverlayを終了するまで有効でなければならない
dwWidth	Overlayサーフェスの幅(pixel単位)
dwHeight	Overlayサーフェスの高さ(pixel単位)

戻り値

正常に終了した場合、MGE_S_OKを返す。それ以外の場合はエラーコードを返す。

MGE_CreateMemorySurface

サーフェスをメモリ上に作成する。

書式

〈C/C++〉

```
MGEAPI MGERESULT WINAPI
MGE_CreateMemorySurface(HMGE hmge, LPHMGESURFACE lphmgesrf,
            DWORD dwWidth, DWORD dwHeight, LONG lPitch,
                    MGEPIXELFORMAT mgpfFormat);
```

〈Basic〉

```
Declare Function MGE_CreateMemorySurface Lib "mge.dll" (ByVal hmge As Long, lphmgesrf As
Long, ByVal dwWidth As Long, ByVal dwHeight As Long, ByVal lPitch As Long, ByVal mgefFormat
As Long) As Long
```

引数

hmge	MGE_Initializeで返されたMGEライブラリのハンドル
lphmgesrf	サーフェスハンドルを格納するメモリアドレス
dwWidth	Memoryサーフェスの幅(pixel単位)
dwHeight	Memoryサーフェスの高さ(pixel単位)
lPitch	Memoryサーフェスのピッチ(ストライド)(byte単位) 現在は正の値のみ
mgpfFormat	Memoryサーフェスのフォーマット MGEPixelFormatRGB32 ・・・ RGB32bit MGEPixelFormatRGB24 ・・・ RGB24bit MGEPixelFormatRGB16 ・・・ RGB16bit MGEPixelFormatRGB15 ・・・ RGB15bit MGEPixelFormatYUY2 ・・・ YUY2

戻り値

正常に終了した場合、MGE_S_OKを返す。それ以外の場合はエラーコードを返す。

補足

*作成できるサーフェスは最大1024x768、最小80x60です。

MGE_DestroySurface

サーフェスを破棄する。

書式

〈C/C++〉

```
MGEAPI MGERESULT WINAPI
MGE_DestroySurface(HMGESURFACE hmgesrfSurface);
```

〈Basic〉

```
Declare Function MGE_DestroySurface Lib "mge.dll" (ByVal hmgesrfSurface As Long) As Long
```

引数

hmgesrfSurface　　サーフェスハンドル

戻り値

正常に終了した場合、MGE_S_OKを返す。それ以外の場合はエラーコードを返す。

MGE_SurfaceSetColorKey

キーカラーを設定する。

書式

〈C/C++〉

```
MGEAPI MGERESULT WINAPI
MGE_SurfaceSetColorKey(HMGESURFACE hmgesrfSurface, LPCMGECOLORKEY lpmgck);
```

〈Basic〉

```
Declare Function MGE_SurfaceSetColorKey Lib "mge.dll" (ByVal hmgesrfSurface As Long, lpmgck
As MGECOLORKEY) As Long
```

引数

hmgesrfSurface	キーカラーを設定するサーフェスハンドル
lpmgck	設定するキーカラー 24bit full colorを指定する 00..07bit 青 08..15bit 緑 16..23bit 赤 24..31bit 0を指定する

戻り値

正常に終了した場合、MGE_S_OKを返す。それ以外の場合はエラーコードを返す。

補足

*DirectDraw Overlay サーフェスのみ有効となる。
*画面(グラフィックボード)モードによっては適宜カラーキーの色変換を行うので、意図した色にならない場合がある。
(例)
24bit→15bit, 24bit→16bit変換の場合

MGE_SurfaceGetColorKey

現在のキーカラーを取得する。

書式

〈C/C++〉

```
MGEAPI MGERESULT WINAPI
MGE_SurfaceGetColorKey(HMGESURFACE hmgesrfSurface, LPMGECOLORKEY lpmgck);
```

〈Basic〉

```
Declare Function MGE_SurfaceGetColorKey Lib "mge.dll" (ByVal hmgesrfSurface As Long, lpmgck
As MGECOLORKEY) As Long
```

引数

hmgesrfSurface	キーカラーを取得するサーフェスハンドル
lpmgck	キーカラーを返すアドレス

戻り値

正常に終了した場合、MGE_S_OKを返す。それ以外の場合はエラーコードを返す。

補足

*DirectDraw Overlayサーフェスのみ有効となる。

MGE_SurfaceSetOverlayMode

オーバーレイモードを設定する。

書式

〈C/C++〉

MGEAPI MGERESULT WINAPI
MGE_SurfaceSetOverlayMode(HMGESURFACE *hmgesrfSurface*, DWORD *dwMode*);

〈Basic〉

Declare Function MGE_SurfaceSetOverlayMode Lib "mge.dll" (ByVal *hmgesrfSurface* As Long, ByVal *dwMode* As Long) As Long

引数

hmgesrfSurface	オーバーレイモードを設定するサーフェスハンドル
dwMode	設定するモード MGE_SOM_COLORKEY

戻り値

正常に終了した場合、MGE_S_OKを返す。それ以外の場合はエラーコードを返す。

補足

*DirectDraw Overlay サーフェスのみ有効となる。
*Power Movie PCIはMGE_SOM_COLORKEYのみサポートしているのでこの関数は使用しない。

MGE_SurfaceGetOverlayMode

現在のオーバーレイモードを取得する。

書式

〈C/C++〉

```
MGEAPI MGERESULT WINAPI
MGE_SurfaceGetOverlayMode(HMGESURFACE hmgesrfSurface, LPDWORD lpdwMode);
```

〈Basic〉

```
Declare Function MGE_SurfaceGetOverlayMode Lib "mge.dll" (ByVal hmgesrfSurface As Long,
lpdwMode As Long) As Long
```

引数

hmgesrfSurface	オーバーレイモードを取得するサーフェスハンドル
lpdwMode	モードを返すアドレス

戻り値

正常に終了した場合、MGE_S_OKを返す。それ以外の場合はエラーコードを返す。

補足

*DirectDraw Overlayサーフェスのみ有効となる。
*Power Movie PCIはMGE_SOM_COLORKEYのみサポートしているのでこの関数は使用しない。

MGE_SurfaceStartOverlay

オーバーレイを開始する。

書式

〈C/C++〉

```
MGEAPI MGERESULT WINAPI
MGE_SurfaceStartOverlay(HMGESURFACE hmgesrfSurface);
```

〈Basic〉

```
Declare Function MGE_SurfaceStartOverlay Lib "mge.dll" (ByVal hmgesrfSurface As Long) As Long
```

引数

hmgesrfSurface　　オーバーレイを開始するサーフェスハンドル

戻り値

正常に終了した場合、MGE_S_OKを返す。それ以外の場合はエラーコードを返す。

補足

*Memoryサーフェスにはオーバーレイできない。

MGE_SurfaceStopOverlay

オーバーレイを停止する。

書式

〈C/C++〉

```
MGEAPI MGERESULT WINAPI
MGE_SurfaceStopOverlay(HMGESURFACE hmgesrfSurface);
```

〈Basic〉

```
Declare Function MGE_SurfaceStopOverlay Lib "mge.dll" (ByVal hmgesrfSurface As Long) As Long
```

引数

hmgesrfSurface　　オーバーレイを停止するサーフェスハンドル

戻り値

正常に終了した場合、MGE_S_OKを返す。それ以外の場合はエラーコードを返す。

補足

*Memoryサーフェスにはオーバーレイできない。

MGE_SurfaceSetBitmapBits

メモリからサーフェスへBitmapをコピーする。

書式

〈C/C++〉

```
MGEAPI MGERESULT WINAPI
MGE_SurfaceSetBitmapBits(HMGESURFACE hmgesrfSurface, LPVOID lpData,
                MGEPIXELFORMAT mgpfFormat,
                LPMGEPALINFO lppi,
                LONG lPitch, LPCRECT lprc);
```

〈Basic〉

```
Declare Function MGE_SurfaceSetBitmapBits Lib "mge.dll" (ByVal hmgesrfSurface As Long, lpData
As Any, ByVal mgpfFormat As Long, lppi As Long, ByVal lPitch As Long, lprc As RECT) As Long
```

引数

hmgesrfSurface	Bitmapをコピーするサーフェスハンドル
lpData	Bitmapデータ データの並びは一番最後のラインがデータの先頭にくる (例) 縦が240ラインのデータは240line目のデータがlpDataの先頭にくる
mgpfFormat	Bitmapのフォーマット MGEPixelFormatRGB24 ・・・ RGB24bit MGEPixelFormatRGB16 ・・・ RGB16bit MGEPixelFormatPAL8 ・・・ 8bitパレット
lppi	mgpFormatがMGEPixelFormatPAL8の時のパレット情報を指定する
lPitch	lpDataのピッチ 4の倍数でなければならない
lprc	Surface上の位置とサイズ

戻り値

正常に終了した場合、MGE_S_OKを返す。それ以外の場合はエラーコードを返す。

補足

*Super Overlayサーフェスにはデータをコピーできない。
*サーフェスがYUY2の場合にはRectは以下の関係を保つ必要がある。
1. Rectのスタート位置は偶数
2. Rectのサイズは4の倍数

MGE_SurfaceGetBitmapBits

サーフェスからメモリへBitmapをコピーする。

書式

〈C/C++〉

```
MGEAPI MGERESULT WINAPI
MGE_SurfaceGetBitmapBits(HMGESURFACE hmgesrfSurface, LPVOID lpData,
                MGEPIXELFORMAT mgpfFormat,
                LPMGEPALINFO lppi,
                LONG lPitch, LPCRECT lprc);
```

〈Basic〉

```
Declare Function MGE_SurfaceGetBitmapBits Lib "mge.dll" (ByVal hmgesrfSurface As Long, lpData
As Any, ByVal mgpfFormat As Long, lppi As Long, ByVal lPitch As Long, lprc As RECT) As Long
```

引数

hmgesrfSurface	Bitmapをコピーするサーフェスハンドル
lpData	Bitmapデータをコピーするメモリのアドレス データの並びは一番最後のラインがデータの先頭にくる （例） 縦が240ラインのデータは240line目のデータがlpDataの先頭にくる
mgpfFormat	Bitmapのフォーマット MGEPixelFormatRGB24 ・・・ RGB24bit MGEPixelFormatRGB16 ・・・ RGB16bit
lppi	将来の拡張用、現在は無視される。
lPitch	lpDataのピッチ 4の倍数でなければならない
lprc	サーフェス上の位置とサイズ

戻り値

正常に終了した場合、MGE_S_OKを返す。それ以外の場合はエラーコードを返す。

補足

*サーフェスがYUY2の場合にはRectは以下の関係を保つ必要がある。

1. Rectのスタート位置は偶数
2. Rectのサイズは4の倍数

*Super Overlayサーフェスからは Stop状態での動作は保証されない。

*Super Overlayサーフェスは指定されたCropping windowが最大の大きさとなる。

MGE_SurfaceFillBuffer

サーフェスを指定した色で塗る。

書式

〈C/C++〉

```
MGEAPI MGERESULT WINAPI
MGE_SurfaceFillBuffer(HMGESURFACE hmgesrfSurface, LPRECT lprc, DWORD dwFillColor);
```

〈Basic〉

```
Declare Function MGE_SurfaceFillBuffer Lib "mge.dll" (ByVal hmgesrfSurface As Long, lprc As
RECT, ByVal dwFillColor As Long) As Long
```

引数

hmgesrfSurface	Bitmapをコピーするサーフェスハンドル
lprc	サーフェス上の位置とサイズ
dwFillColor	塗る色、24bit colorを指定する

戻り値

正常に終了した場合、MGE_S_OKを返す。それ以外の場合はエラーコードを返す。

補足

*Super Overlayサーフェスには使用できない。
*サーフェスがYUY2の場合にはRectは以下の関係を保つ必要がある。
1. Rectのスタート位置は偶数
2. Rectのサイズは4の倍数

MGE_SurfaceLoadDIB

サーフェスにDIBをロードする。

書式

〈C/C++〉

```
MGEAPI MGERESULT WINAPI
MGE_SurfaceLoadDIB(HMGESURFACE hmgesrfSurface,
                   LPCSTR lpszFileName, LPCRECT lprc);
```

〈Basic〉

```
Declare Function MGE_SurfaceLoadDIB Lib "mge.dll" Alias "MGE_SurfaceLoadDIBA" (ByVal
hmgesrfSurface As Long, ByVal lpszFileName As String, lprc As RECT) As Long
```

引数

hmgesrfSurface	DIBをロードするサーフェスハンドル
lpszFileName	DIBのファイル名
lprc	DIBをロードするサーフェスの位置、大きさ

戻り値

正常に終了した場合、MGE_S_OKを返す。それ以外の場合はエラーコードを返す。

補足

*Super Overlayサーフェスにはロードできない。
*サーフェスがYUY2の場合にはRectは以下の関係を保つ必要がある。
1. Rectのスタート位置は偶数
2. Rectのサイズは4の倍数

*1024x768より大きいDIBはロードできない。

MGE_SurfaceSaveDIB

サーフェスのデータをDIBとしてファイルにセーブする。

書式

〈C/C++〉

```
MGEAPI MGERESULT WINAPI
MGE_SurfaceSaveDIB(HMGESURFACE hmgesrfSurface,
                   LPCSTR lpszFileName, LPCRECT lprc,
                   MGEPIXELFORMAT mgpfFormat);
```

〈Basic〉

```
Declare Function MGE_SurfaceSaveDIB Lib "mge.dll" Alias "MGE_SurfaceSaveDIBA" (ByVal
hmgesrfSurface As Long, ByVal lpszFileName As String, lprc As RECT, ByVal mgpfFormat As
Long) As Long
```

引数

hmgesrfSurface	DIBをセーブするサーフェスハンドル
lpszFileName	DIBのファイル名
lprc	DIBをセーブするサーフェスの位置、大きさ
mgpfFormat	DIBのフォーマット MGEPixelFormatRGB24 ･･･ RGB24bit MGEPixelFormatRGB16 ･･･ RGB16bit

戻り値

正常に終了した場合、MGE_S_OKを返す。それ以外の場合はエラーコードを返す。

補足

*サーフェスがYUY2の場合にはRectは以下の関係を保つ必要がある。

1. Rectのスタート位置は偶数
2. Rectのサイズは4の倍数

*Super Overlayサーフェスからは Stop状態での動作は保証されない。

*Super Overlayサーフェスは指定されたCropping windowが最大の大きさとなる。

MGE_SurfaceLoadJPEG

サーフェスに JPEG データを展開してロードする。

書式

〈C/C++〉

```
MGEAPI MGERESULT WINAPI
MGE_SurfaceLoadJPEG(HMGESURFACE hmgesrfSurface,
                    LPCSTR lpszFileName, LPCRECT lprc);
```

〈Basic〉

```
Declare Function MGE_SurfaceLoadJPEG Lib "mge.dll" Alias "MGE_SurfaceLoadJPEGA" (ByVal
hmgesrfSurface As Long, ByVal lpszFileName As String, lprc As RECT) As Long
```

引数

hmgesrfSurface	JPEGをロードするサーフェスハンドル
lpszFileName	JPEGのファイル名
lprc	JPEGをロードするサーフェスの位置、大きさ

戻り値

正常に終了した場合、MGE_S_OKを返す。それ以外の場合はエラーコードを返す。

補足

*Super Overlayサーフェスにはロードできない。
*サーフェスがYUY2の場合にはRectは以下の関係を保つ必要がある。
1. Rectのスタート位置は偶数
2. Rectのサイズは4の倍数

MGE_SurfaceSaveJPEG

サーフェスのデータを JPEG に圧縮してファイルにセーブする。

書式

〈C/C++〉

```
MGEAPI MGERESULT WINAPI
MGE_SurfaceSaveJPEG(HMGESURFACE hmgesrfSurface,
                    LPCSTR lpszFileName, LPCRECT lprc,
                    MGEJPEGQUALITY mgjqQuality);
```

〈Basic〉

```
Declare Function MGE_SurfaceSaveJPEG Lib "mge.dll" Alias "MGE_SurfaceSaveJPEGA" (ByVal
hmgesrfSurface As Long, ByVal lpszFileName As String, lprc As RECT, ByVal mgjqQuality As
Long) As Long
```

引数

hmgesrfSurface	JPEGをセーブするサーフェスハンドル
lpszFileName	JPEGのファイル名
lprc	JPEGをセーブするサーフェスの位置、大きさ
mgjqQuality	JPEGの圧縮率 1/nのn×1000を指定する 現在サポートしている値は以下の通り 5000(1/5) 7000(1/7) 10000(1/10) 12000(1/12) 15000(1/15) 17000(1/17) 20000(1/20) 25000(1/25) 30000(1/30) 40000(1/40)

戻り値

正常に終了した場合、MGE_S_OKを返す。それ以外の場合はエラーコードを返す。

補足

*Super Overlayサーフェスデータのセーブをサポートしています。
*サーフェスがYUY2の場合にはRectは以下の関係を保つ必要がある。
1. Rectのスタート位置は偶数
2. Rectのサイズは4の倍数

*Super Overlayサーフェスからは Stop状態での動作は保証されない。
*Super Overlayサーフェスは指定されたCropping windowが最大の大きさとなる。

Chapter 6

Video Decoder Functions

MGE_CreateVideoDecoder

ビデオデコーダオブジェクトを作成する。

書式

〈C/C++〉

```
MGEAPI MGERESULT WINAPI
MGE_CreateVideoDecoder(HMGE hmge, MGECHANNEL mgcChannel, LPHMGEVSRC lphmgevs);
```

〈Basic〉

```
Declare Function MGE_CreateVideoDecoder Lib "mge.dll" (ByVal hmge As Long, ByVal mgcChannel
As Long, lphmgevs As Long) As Long
```

引数

hmge	MGE_Initializeで返されたMGEライブラリのハンドル
mgcChannel	ビデオデコーダ番号。次の値のいずれかを指定します。 MGEChannel1 MGEChannel2
lphmgevs	ビデオデコーダへのハンドルを格納するメモリアドレス

戻り値

正常に終了した場合、MGE_S_OKを返す。それ以外の場合はエラーコードを返す。

補足

*ビデオデコーダ番号と使用するサーフェスは密接に関係しています。MGE_VStreamConnectを参照ください。

MGE_DestroyVideoDecoder

ビデオデコーダオブジェクトを破棄する。

書式

〈C/C++〉

```
MGEAPI MGERESULT WINAPI
MGE_DestroyVideoDecoder(HMGEVSRC hmgevs);
```

〈Basic〉

```
Declare Function MGE_DestroyVideoDecoder Lib "mge.dll" (ByVal hmgevs As Long) As Long
```

引数

hmgevs　　ビデオデコーダへのハンドル

戻り値

正常に終了した場合、MGE_S_OKを返す。それ以外の場合はエラーコードを返す。

MGE_VDGetStatus

ビデオデコーダの状態を取得する。

書式

〈C/C++〉

```
MGEAPI MGERESULT WINAPI
MGE_VDGetStatus(HMGEVSRC hmgevs, MGELINE mgILine, LPMGEVDSTATUS lpmgevds);
```

〈Basic〉

```
Declare Function MGE_VDGetStatus Lib "mge.dll" (ByVal hmgevs As Long, ByVal mgILine As
Long, ByVal lpmgevds As Long) As Long
```

引数

hmgevs	ビデオデコーダへのハンドル
mgILine	入力ライン番号
lpmgevds	Video Decoderのステータスを格納するアドレス

戻り値

正常に終了した場合、MGE_S_OKを返す。それ以外の場合はエラーコードを返す。

補足

*現在選択されているLineと違うLineを指定した場合には、一度指定したLineに入力が変更された後に元のLineに戻る。

MGE_VDSetParam

ビデオデコーダのパラメータを調整する。

書式

〈C/C++〉

```
MGEAPI MGERESULT WINAPI
MGE_VDSetParam(HMGEVSRC hmgevs, MGELINE mgLine,
               MGEVIDEOAMPPROPERTY mgvap, DWORD dwValue);
```

〈Basic〉

```
Declare Function MGE_VDSetParam Lib "mge.dll" (ByVal hmgves As Long, ByVal mgLine As Long,
ByVal mgvap As Long, ByVal dwValue As Long) As Long
```

引数

hmgevs　ビデオデコーダへのハンドル

mgLine　入力ライン番号

mgvap　調整可能項目

- MGEVideoAmpBrightness ･･･ 明るさ
- MGEVideoAmpHue ･･･ 色合い
- MGEVideoAmpContrast ･･･ コントラスト
- MGEVideoAmpSaturation ･･･ 色の濃さ
- MGEVideoAmpSharpness ･･･ 輪郭強調
- MGEVideoAmpColorEnable ･･･ Power Movie PCIでは使用しません
- MGEVideoAmpGamma ･･･ Power Movie PCIでは使用しません
- MGEVideoAmpWhiteBalance ･･･ Power Movie PCIでは使用しません

dwValue　設定する値

戻り値

正常に終了した場合、MGE_S_OKを返す。それ以外の場合はエラーコードを返す。

MGE_VDGetParam

現在のビデオデコーダのパラメータを取得する。

書式

〈C/C++〉

```
MGEAPI MGERESULT WINAPI
MGE_VDGetParam(HMGEVSRC hmgevs, MGELINE mgLine,
               MGEVIDEOAMPPROPERTY mgvap, LPDWORD lpdwValue);
```

〈Basic〉

```
Declare Function MGE_VDGetParam Lib "mge.dll" (ByVal hmgves As Long, ByVal mgLine As Long,
ByVal mgvap As Long, lpdwValue As Long) As Long
```

引数

hmgevs	ビデオデコーダへのハンドル
mgLine	入力ライン番号
mgvap	調整項目
	MGEVideoAmpBrightness ・・・ 明るさ
	MGEVideoAmpHue ・・・ 色合い
	MGEVideoAmpContrast ・・・ コントラスト
	MGEVideoAmpSaturation ・・・ 色の濃さ
	MGEVideoAmpSharpness ・・・ 輪郭強調
	MGEVideoAmpColorEnable ・・・ Power Movie PCIでは使用しません
	MGEVideoAmpGamma ・・・ Power Movie PCIでは使用しません
	MGEVideoAmpWhiteBalance ・・・ Power Movie PCIでは使用しません
lpdwValue	値を格納するアドレス

戻り値

正常に終了した場合、MGE_S_OKを返す。それ以外の場合はエラーコードを返す。

MGE_VDGetParamRange

ビデオデコーダの調整可能パラメータの範囲を取得する。

書式

〈C/C++〉

```
MGEAPI MGERESULT WINAPI
MGE_VDGetParamRange(HMGEVSRC hmgevs, MGELINE mgLine,
            MGEVIDEOAMPPROPERTY mgvap,
            LPDWORD lpdwMin, LPDWORD lpdwMax, LPDWORD lpdwDef);
```

〈Basic〉

```
Declare Function MGE_VDGetParamRange Lib "mge.dll" (ByVal hmgves As Long, ByVal mgLine As
Long, ByVal mgvap As Long, lpdwMin As Long, lpdwMax As Long, lpdwDef As Long) As Long
```

引数

hmgevs	ビデオデコーダへのハンドル
mgLine	入力ライン番号
mgvap	調整項目
	MGEVideoAmpBrightness ･･･ 明るさ
	MGEVideoAmpHue ･･･ 色合い
	MGEVideoAmpContrast ･･･ コントラスト
	MGEVideoAmpSaturation ･･･ 色の濃さ
	MGEVideoAmpSharpness ･･･ 輪郭強調
	MGEVideoAmpColorEnable ･･･ Power Movie PCIでは使用しません
	MGEVideoAmpGamma ･･･ Power Movie PCIでは使用しません
	MGEVideoAmpWhiteBalance ･･･ Power Movie PCIでは使用しません
lpdwMin	最小値を格納するアドレス
lpdwMax	最大値を格納するアドレス
lpdwDef	default値を格納するアドレス

戻り値

正常に終了した場合、MGE_S_OKを返す。それ以外の場合はエラーコードを返す。

補足

*VideoDecoder用の調整パラメータの最小値、最大値、default値を得ます。

*この関数はVideoDecoder用の調整Dialog等を作成するためにあります。Applicationはこの関数を使用し、各調整可能項目(もしくは調整させたい項目)の最小値、最大値、default値を取得して記憶します。最小値、最大値はscroll bar等に渡し、範囲を限定することに使用できます。またdefault値は"初期値"などのボタンを実装した場合に使用できます。Applicationのユーザが値を指定したときにMGE_VDSetParamを通じてPower Movie PCIに通知できます。

MGE_VDSetLine

ビデオデコーダへの入力を選択する。

書式

〈C/C++〉

```
MGEAPI MGERESULT WINAPI
MGE_VDSetLine(HMGEVSRC hmgevs, MGELINE mgLine);
```

〈Basic〉

```
Declare Function MGE_VDSetLine Lib "mge.dll" (ByVal hmgves As Long, ByVal mgLine As Long)
As Long
```

引数

hmgevs	ビデオデコーダへのハンドル
mgLine	入力ライン番号

戻り値

正常に終了した場合、MGE_S_OKを返す。それ以外の場合はエラーコードを返す。

補足

*Power Movie PCIの実装はMGELine1, MGELine2, MGELine3, MGELine4に対応しています。
*入力ライン番号はコネクタの番号に対応しています。MGELine1がIN1..MGELine4がIN4です。

MGE_VDGetLine

現在のビデオデコーダへの入力を取得する。

書式

〈C/C++〉

```
MGEAPI MGERESULT WINAPI
MGE_VDGetLine(HMGEVSRC hmgevs, LPMGELINE lpmgLine);
```

〈Basic〉

```
Declare Function MGE_VDGetLine Lib "mge.dll" (ByVal hmgves As Long, lpmgLine As Long) As Long
```

引数

hmgevs	ビデオデコーダへのハンドル
lpmgLine	入力ライン番号を格納するアドレス

戻り値

正常に終了した場合、MGE_S_OKを返す。それ以外の場合はエラーコードを返す。

MGE_VDSaveConfiguration

ビデオデコーダの設定を registry に格納する。

書式

〈C/C++〉

MGEAPI MGERESULT WINAPI
MGE_VDSaveConfiguration(HMGEVSRC *hmgevs*, HKEY *hKey*, LPCSTR *lpszBase*);

〈Basic〉

Declare Function MGE_VDSaveConfiguration Lib "mge.dll" (ByVal *hmgves* As Long, ByVal *hKey* As Long, ByVal *lpszBase* As String) As Long

引数

hmgevs	ビデオデコーダへのハンドル
hKey	registryのroot key HKEY_CURRENT_USERなど
lpszBase	registryのkey、実際の値は対応するchannel番号の下に作成される (MGE_SaveConfiguration参照)

戻り値

正常に終了した場合、MGE_S_OKを返す。それ以外の場合はエラーコードを返す。

補足

*ビデオデコーダの設定を個別に保存したい場合に使用します。通常はMGE_SaveConfigurationを使用することになります。

MGE_VDLoadConfiguration

ビデオデコーダの設定を registry からロードする。

書式

〈C/C++〉

MGEAPI MGERESULT WINAPI
MGE_VDLoadConfiguration(HMGEVSRC *hmgevs*, HKEY *hKey*, LPCSTR *lpszBase*);

〈Basic〉

Declare Function MGE_VDLoadConfiguration Lib "mge.dll" (ByVal *hmgves* As Long, ByVal *hKey* As Long, ByVal *lpszBase* As String) As Long

引数

hmgevs	ビデオデコーダへのハンドル
hKey	registryのroot key HKEY_CURRENT_USERなど
lpszBase	registryのkey (MGE_VDSaveConfiguration参照)

戻り値

正常に終了した場合、MGE_S_OKを返す。それ以外の場合はエラーコードを返す。

補足

*ビデオデコーダの設定を個別にロードしたい場合に使用します。通常はMGE_LoadConfiguration を使用することになります。

Chapter 7

Video Encoder Functions

MGE_CreateVideoEncoder

ビデオエンコーダオブジェクトを作成する。

書式

〈C/C++〉

MGEAPI MGERESULT WINAPI
MGE_CreateVideoEncoder(HMGE *hmge*, MGECHANNEL *mgcChannel*, LPHMGEVDST *lphmgevd*);

〈Basic〉

Declare Function MGE_CreateVideoEncoder Lib "mge.dll" (ByVal *hmge* As Long, ByVal *mgcChannel* As Long, *lphmgevd* As Long) As Long

引数

hmge	MGE_Initializeで返されたMGEライブラリのハンドル
mgcChannel	ビデオエンコーダ番号
lphmgevd	ビデオエンコーダへのハンドルを格納するメモリアドレス

戻り値

正常に終了した場合、MGE_S_OKを返す。それ以外の場合はエラーコードを返す。

補足

*Power Movie PCIの実装は、MGEChannel1のみをサポートしています。

MGE_DestroyVideoEncoder

ビデオエンコーダオブジェクトを破棄する。

書式

〈C/C++〉

```
MGEAPI MGERESULT WINAPI
MGE_DestroyVideoEncoder(HMGEVDST hmgevd);
```

〈Basic〉

```
Declare Function MGE_DestroyVideoEncoder Lib "mge.dll" (ByVal hmgevd As Long) As Long
```

引数

hmgevd　　ビデオエンコーダへのハンドル

戻り値

正常に終了した場合、MGE_S_OKを返す。それ以外の場合はエラーコードを返す。

MGE_VESetLine

ビデオエンコーダの出力を選択する。

書式

〈C/C++〉

```
MGEAPI MGERESULT WINAPI
MGE_VESetLine(HMGEVDST hmgevd, MGELINE mglLine);
```

〈Basic〉

```
Declare Function MGE_VESetLine Lib "mge.dll" (ByVal hmgevd As Long, ByVal mglLine As Long)
As Long
```

引数

hmgevd	ビデオエンコーダへのハンドル
mglLine	出力ライン番号

戻り値

正常に終了した場合、MGE_S_OKを返す。それ以外の場合はエラーコードを返す。

補足

*Power Movie PCIの実装はMGELine1のみをサポートしているため、この関数を使用する必要はありません。

MGE_VEGetLine

現在のビデオエンコーダの出力ライン番号を取得する。

書式

〈C/C++〉

```
MGEAPI MGERESULT WINAPI
MGE_VEGetLine(HMGEVDST hmgevd, LPMGELINE lpmgLine);
```

〈Basic〉

```
Declare Function MGE_VEGetLine Lib "mge.dll" (ByVal hmgevd As Long, lpmgLine As Long) As Long
```

引数

hmgevd	ビデオエンコーダへのハンドル
lpmgLine	出力ライン番号を格納するアドレス

戻り値

正常に終了した場合、MGE_S_OKを返す。それ以外の場合はエラーコードを返す。

補足

*Power Movie PCIの実装はMGELine1のみをサポートしているため、この関数を使用する必要はありません。

MGE_VESaveConfiguration

ビデオエンコーダの設定を registry に格納する。

書式

〈C/C++〉

MGEAPI MGERESULT WINAPI
MGE_VESaveConfiguration(HMGEVDST *hmgevd*, HKEY *hKey*, LPCSTR *lpszBase*);

〈Basic〉

Declare Function MGE_VESaveConfiguration Lib "mge.dll" (ByVal *hmgevd* As Long, ByVal *hKey* As Long, ByVal *lpszBase* As String) As Long

引数

hmgevd	ビデオエンコーダへのハンドル
hKey	registryのroot key HKEY_CURRENT_USERなど
lpszBase	registryのkey、実際の値は対応するchannel番号の下に作成される (MGE_SaveConfiguration参照)

戻り値

正常に終了した場合、MGE_S_OKを返す。それ以外の場合はエラーコードを返す。

補足

*今は何も情報は書かれていない。

MGE_VELoadConfiguration

ビデオエンコーダの設定を registry からロードする。

書式

〈C/C++〉

```
MGEAPI MGERESULT WINAPI
MGE_VELoadConfiguration(HMGEVDST hmgevd, HKEY hKey, LPCSTR lpszBase);
```

〈Basic〉

```
Declare Function MGE_VELoadConfiguration Lib "mge.dll" (ByVal hmgevd As Long, ByVal hKey As
Long, ByVal lpszBase As String) As Long
```

引数

hmgevd	ビデオエンコーダへのハンドル
hKey	registryのroot key HKEY_CURRENT_USERなど
lpszBase	registryのkey (MGE_VESaveConfiguration参照)

戻り値

正常に終了した場合、MGE_S_OKを返す。それ以外の場合はエラーコードを返す。

補足

*今は何も情報はロードされない。

Chapter 8

Video Stream Functions

MGE_CreateVideoInStream

入力用のビデオストリームを作成する。

書式

〈C/C++〉

```
MGEAPI MGERESULT WINAPI
MGE_CreateVideoInStream(HMGEVSRC hmgevs, LPHMGESTREAM lphmgestm);
```

〈Basic〉

```
Declare Function MGE_CreateVideoInStream Lib "mge.dll" (ByVal hmgevs As Long, lphmgestm As
Long) As Long
```

引数

hmgevs	ビデオソースへのハンドル
lphmgestm	ビデオストリームハンドルを格納するメモリアドレス

戻り値

正常に終了した場合、MGE_S_OKを返す。それ以外の場合はエラーコードを返す。

MGE_CreateVideoOutStream

出力用のビデオストリームを作成する。

書式

〈C/C++〉

```
MGEAPI MGERESULT WINAPI
MGE_CreateVideoOutStream(HMGEVDST hmgevd, LPHMGESTREAM lphmgestm);
```

〈Basic〉

```
Declare Function MGE_CreateVideoOutStream Lib "mge.dll" (ByVal hmgevd As Long, lphmgestm As
Long) As Long
```

引数

hmgevd	ビデオ出力へのハンドル
lphmgestm	ビデオストリームハンドルを格納するメモリアドレス

戻り値

正常に終了した場合、MGE_S_OKを返す。それ以外の場合はエラーコードを返す。

補足

*このストリームにConnectできるのはMGEPixelFormatYUY2のMemoryサーフェスのみです。

MGE_DestroyVideoStream

ビデオストリームを破棄する。

書式

〈C/C++〉

```
MGEAPI MGERESULT WINAPI
MGE_DestroyVideoStream(HMGESTREAM hmgestm);
```

〈Basic〉

```
Declare Function MGE_DestroyVideoStream Lib "mge.dll" (ByVal hmgestm As Long) As Long
```

引数

hmgestm　　破棄するビデオストリームハンドル

戻り値

正常に終了した場合、MGE_S_OKを返す。それ以外の場合はエラーコードを返す。

MGE_VStreamConnect

ビデオストリームをサーフェスに結び付ける。

書式

〈C/C++〉

MGEAPI MGERESULT WINAPI
MGE_VStreamConnect(HMGESTREAM *hmgestmStream*, HMGESURFACE *hmgesrfSurface*);

〈Basic〉

Declare Function MGE_VStreamConnect Lib "mge.dll" (ByVal *hmgestmStream* As Long, ByVal *hmgesrfSurface* As Long) As Long

引数

hmgestmStream	結び付けるストリームハンドル
hmgesrfSurface	結び付けるサーフェスハンドル

戻り値

正常に終了した場合、MGE_S_OKを返す。それ以外の場合はエラーコードを返す。

補足

*Power Movie PCIの実装では下記の制限があります。
1.MGE_CreateVideoInStreamで作成したストリームとサーフェスを結び付けた場合。
1-1. MGEChannel1で作成された(MGE_CreateVideoDecoder参照)ビデオデコーダを使用して作成されたストリームは下記のサーフェスにしか結び付けられません。
1-1-1 Super Overlayサーフェス
1-1-2 下記のフォーマットのMemoryサーフェス
1-1-2-1 MGEPixelFormatRGB32
1-1-2-2 MGEPixelFormatRGB24
1-1-2-3 MGEPixelFormatRGB15
1-2. MGEChannel2で作成された(MGE_CreateVideoDecoder参照)ビデオデコーダを使用して作成されたストリームは下記のサーフェスにしか結び付けられません。
1-2-1 DirectDraw Overlayサーフェス
2. MGE_CreateVideoOutStreamで作成したストリームとサーフェスを結び付けた場合。
MGEPixelFormatYUY2で作成されたMemory surfaceにしか結び付けられません。

MGE_VStreamDisconnect

ビデオストリームとサーフェスを切り離す。

書式

〈C/C++〉

```
MGEAPI MGERESULT WINAPI
MGE_VStreamDisconnect(HMGESTREAM hmgestmStream, HMGESURFACE hmgesrfSurface);
```

〈Basic〉

```
Declare Function MGE_VStreamDisconnect Lib "mge.dll" (ByVal hmgestmStream As Long, ByVal
hmgesrfSurface As Long) As Long
```

引数

hmgestmStream	切り離すストリームハンドル
hmgesrfSurface	切り離すサーフェスハンドル

戻り値

正常に終了した場合、MGE_S_OKを返す。それ以外の場合はエラーコードを返す。

MGE_VStreamReplace

ビデオストリームのサーフェスを付け替える。

書式

〈C/C++〉

```
MGEAPI MGERESULT WINAPI
MGE_VStreamReplace(HMGESTREAM hmgestmStream, HMGESURFACE hmgesrfSurface);
```

〈Basic〉

```
Declare Function MGE_VStreamReplace Lib "mge.dll" (ByVal hmgestmStream As Long, ByVal
hmgesrfSurface As Long) As Long
```

引数

hmgestmStream	付け替えるストリームハンドル
hmgesrfSurface	付け替えるサーフェスハンドル

戻り値

正常に終了した場合、MGE_S_OKを返す。それ以外の場合はエラーコードを返す。

補足

*付け替えるサーフェスは付け替える前のサーフェスと同様の属性を持っている必要がある。
*現在Power Movie PCIの実装では MGE_CreateVideoOutStreamで作成されたストリームに結び付けられたサーフェスのみを付け替えることができる。

MGE_VStreamSetVideoMode

ビデオストリーム上のビデオ信号モードを設定する。

書式

〈C/C++〉

```
MGEAPI MGERESULT WINAPI
MGE_VStreamSetVideoMode(HMGESTREAM hmgestmStream,
                MGEVIDEOMODE mgevm, BOOL flmmediate);
```

〈Basic〉

```
Declare Function MGE_VStreamSetVideoMode Lib "mge.dll" (ByVal hmgestmStream As Long, ByVal
mgevm As Long, ByVal flmmediate As Boolean) As Long
```

引数

hmgestmStream	設定するストリームハンドル
mgevm	ビデオモード
	MGEVideoModeFrame ・・・ 両フィールドモード
	MGEVideoModeOddField ・・・ 奇数フィールドモード
	MGEVideoModeEvenField ・・・ 偶数フィールドモード
flmmediate	すぐに反映させるかどうかを指定
	(MGE_Update 参照)

戻り値

正常に終了した場合、MGE_S_OKを返す。それ以外の場合はエラーコードを返す。

補足

*DirectDraw Overlayサーフェスへのストリーム以外はエラーとなる。

MGE_VStreamGetVideoMode

現在のビデオストリーム上のビデオ信号モードを取得する。

書式

〈C/C++〉

```
MGEAPI MGERESULT WINAPI
MGE_VStreamGetVideoMode(HMGESTREAM hmgestmStream, LPMGEVIDEOMODE lpmgevm);
```

〈Basic〉

```
Declare Function MGE_VStreamGetVideoMode Lib "mge.dll" (ByVal hmgestmStream As Long,
lpmgevm As Long) As Long
```

引数

hmgestmStream　取得するストリーム

lpmgevm　ビデオモードを格納するアドレス

- MGEVideoModeFrame　・・・ 両フィールドモード
- MGEVideoModeOddField　・・・ 奇数フィールドモード
- MGEVideoModeEvenField ・・・ 偶数フィールドモード

戻り値

正常に終了した場合、MGE_S_OKを返す。それ以外の場合はエラーコードを返す。

補足

*DirectDraw Overlayサーフェスへのストリーム以外はエラーとなる。

MGE_VStreamSetRect

次の各矩形を指定する。

1. ビデオ信号の取り込み
2. フレームバッファへの書き込み
3. フレームバッファからの読み出し
4. 画面の位置 (screen coordinate)

書式

〈C/C++〉

```
MGEAPI MGERESULT WINAPI
MGE_VStreamSetRect(HMGESTREAM hmgestmStream, MGESTREAMRECT mgestmrc,
                   LPCRECT lprc, BOOL flmmediate);
```

〈Basic〉

```
Declare Function MGE_VStreamSetRect Lib "mge.dll" (ByVal hmgestmStream As Long, ByVal
mgestmrc As Long, lprc As RECT, ByVal flmmediate As Boolean) As Long
```

引数

hmgestmStream	ストリームへのハンドル
mgestmrc	各矩形の識別子
	MGEStreamRectCrop ・・・ ビデオ信号の取り込み
	MGEStreamRectFrame ・・・ フレームバッファへの書き込み
	MGEStreamRectView ・・・ フレームバッファからの読み出し
	MGEStreamRectWindow ・・・ 画面の位置(screen coordinate)
lprc	矩形のアドレス
flmmediate	すぐに反映させるかどうかを指定
	(MGE_Update 参照)

戻り値

正常に終了した場合、MGE_S_OKを返す。それ以外の場合はエラーコードを返す。

補足

＊ flmmediateがFALSEの時にはrectの整合性はチェックされない。
＊ ストリームが停止/一時停止しているときには画像内容は更新されない。
＊ サーフェスがSuperOverlayの場合は以下の関係を保つ必要がある。
1.各Rectの関係はRectCrop 〉= RectWindow
2.FrameとViewは無視される。
3.マップされたhWndのWindowが動くと内部的にRectWindowが調整される。ただし、RectCrop 〉= RectWindowを保つ必要があるので、hWndのOwnerはその関係を保つ必要がある。
＊ サーフェスがDDrawOvlSurfaceの場合には各Rectは以下の関係を保つ必要がある。
1. 各rectのスタート位置は偶数
2. 各rectのサイズは偶数
3. RectCropのサイズ 〉= RectFrameのサイズ
RectFrameの幅はRectCropの1/1, 1/2, 1/4でなければならない。
RectFrameの高さはRectCropの1/1, 1/2, 1/4, 1/8でなければならない。
4. RectWindowのサイズ 〉= RectViewのサイズ
＊ サーフェスがYUY2の場合には各Rectは以下の関係を保つ必要がある。
1. 各rectのスタート位置は偶数
2. Rectのサイズは4の倍数
＊ サーフェスがMemory Surfaceの場合は以下の関係を保つ必要がある。
1.ストリームがVideo inではFrameとCropのみ有効となる。
2.ストリームがVideo outではViewとWindowのみ有効となる。

MGE_VStreamGetRect

現在の矩形情報を取得する。

1. ビデオ信号の取り込み
2. フレームバッファへの書き込み
3. フレームバッファからの読み出し
4. 画面の位置 (screen coordinate)

書式

〈C/C++〉

```
MGEAPI MGERESULT WINAPI
MGE_VStreamGetRect(HMGESTREAM hmgestmStream, MGESTREAMRECT mgestmrc, LPRECT lprc);
```

〈Basic〉

```
Declare Function MGE_VStreamGetRect Lib "mge.dll" (ByVal hmgestmStream As Long, ByVal
mgestmrc As Long, lprc As RECT) As Long
```

引数

hmgestmStream	ストリームへのハンドル
mgestmrc	各矩形の識別子
	MGEStreamRectCrop ・・・ ビデオ信号の取り込み
	MGEStreamRectFrame ・・・ フレームバッファへの書き込み
	MGEStreamRectView ・・・ フレームバッファからの読み出し
	MGEStreamRectWindow ・・・ 画面の位置(screen coordinate)
lprc	情報を返す矩形のアドレス

戻り値

正常に終了した場合、MGE_S_OKを返す。それ以外の場合はエラーコードを返す。

MGE_VStreamGetRectRange

各矩形情報を取得する。

1. ビデオ信号の取り込み
2. フレームバッファへの書き込み
3. フレームバッファからの読み出し
4. 画面の位置 (screen coordinate)

書式

〈C/C++〉

```
MGEAPI MGERESULT WINAPI
MGE_VStreamGetRectRange(HMGESTREAM hmgestmStream,
                        MGESTREAMRECT mgestmrc,
                        LPRECT lprcMin,
                        LPRECT lprcMax,
                        LPRECT lprcDef);
```

〈Basic〉

```
Declare Function MGE_VStreamGetRectRange Lib "mge.dll" (ByVal hmgestmStream As Long, ByVal
mgestmrc As Long, lprcMin As RECT, lprcMax As RECT, lprcDef As RECT) As Long
```

引数

hmgestmStream	ストリームへのハンドル
mgestmrc	各矩形の識別子
	MGEStreamRectCrop ・・・ ビデオ信号の取り込み
	MGEStreamRectFrame ・・・ フレームバッファへの書き込み
	MGEStreamRectView ・・・ フレームバッファからの読み出し
	MGEStreamRectWindow ・・・ 画面の位置(screen coordinate)
lprcMin	最小の矩形サイズを返す矩形のアドレス
lprcMax	最大の矩形サイズを返す矩形のアドレス
lprcDef	デフォルトの矩形サイズを返す矩形のアドレス

戻り値

正常に終了した場合、MGE_S_OKを返す。それ以外の場合はエラーコードを返す。

MGE_VStreamStart

ストリームを開始する。

書式

〈C/C++〉

```
MGEAPI MGERESULT WINAPI
MGE_VStreamStart(HMGESTREAM hmgestmStream);
```

〈Basic〉

```
Declare Function MGE_VStreamStart Lib "mge.dll" (ByVal hmgestmStream As Long) As Long
```

引数

hmgestmStream　　開始するストリームハンドル

戻り値

正常に終了した場合、MGE_S_OKを返す。それ以外の場合はエラーコードを返す。

MGE_VStreamPause

ストリームを一時停止させる。

書式

〈C/C++〉

```
MGEAPI MGERESULT WINAPI
MGE_VStreamPause(HMGESTREAM hmgestmStream);
```

〈Basic〉

```
Declare Function MGE_VStreamPause Lib "mge.dll" (ByVal hmgestmStream As Long) As Long
```

引数

hmgestmStream　　一時停止させるストリームハンドル

戻り値

正常に終了した場合、MGE_S_OKを返す。それ以外の場合はエラーコードを返す。

補足

*Super Overlayは一時停止できない。

MGE_VStreamRestart

ストリームを再開させる。

書式

〈C/C++〉

```
MGEAPI MGERESULT WINAPI
MGE_VStreamRestart(HMGESTREAM hmgestmStream);
```

〈Basic〉

```
Declare Function MGE_VStreamRestart Lib "mge.dll" (ByVal hmgestmStream As Long) As Long
```

引数

hmgestmStream　　再開させるストリームハンドル

戻り値

正常に終了した場合、MGE_S_OKを返す。それ以外の場合はエラーコードを返す。

補足

*Super Overlayは再開できない。

MGE_VStreamStop

ストリームを停止させる。

書式

〈C/C++〉

```
MGEAPI MGERESULT WINAPI
MGE_VStreamStop(HMGESTREAM hmgestmStream);
```

〈Basic〉

```
Declare Function MGE_VStreamStop Lib "mge.dll" (ByVal hmgestmStream As Long) As Long
```

引数

hmgestmStream　　停止させるストリームハンドル

戻り値

正常に終了した場合、MGE_S_OKを返す。それ以外の場合はエラーコードを返す。

補足

*Super Overlayに対して停止をかけた場合の画面更新は、アプリケーションが行う。

MGE_VStreamGetState

指定したストリームの状態を取得する。

書式

〈C/C++〉

```
MGEAPI MGERESULT WINAPI
MGE_VStreamGetState(HMGESTREAM hmgestmStream, LPMGESTREAMSTATE lpmgess);
```

〈Basic〉

```
Declare Function MGE_VStreamGetState Lib "mge.dll" (ByVal hmgestmStream As Long, lpmgess As
Long) As Long
```

引数

hmgestmStream	状態を取得するストリーム
lpmgess	返ってくるストリームの状態を格納するアドレス

戻り値

正常に終了した場合、MGE_S_OKを返す。それ以外の場合はエラーコードを返す。
lpmgessに返ってくるストリームの状態

MGEStreamStateNotComplete	・・・ 未完成のストリーム
MGEStreamStateStarted	・・・ ストリームは動いている
MGEStreamStatePause	・・・ ストリームは一時停止している
MGEStreamStateStop	・・・ ストリームは停止している

MGE_VStreamSetEffect

ストリームに特殊効果を付加する。

書式

〈C/C++〉

```
MGEAPI MGERESULT WINAPI
MGE_VStreamSetEffect(HMGESTREAM hmgestmStream, DWORD dwEffect, BOOL flmmediate);
```

〈Basic〉

```
Declare Function MGE_VStreamSetEffect Lib "mge.dll" (ByVal hmgestmStream As Long, ByVal
dwEffect As Long, ByVal flmmediate As Boolean) As Long
```

引数

hmgestmStream	特殊効果を付加するストリームハンドル
dwEffect	各effectを指定する。 MGEEF_MIRROR ･･･ 左右反転 MGEEF_UPSIDEDOWN ･･･ 上下反転
flmmediate	すぐに反映させるかどうかを指定 (MGE_Update 参照)

戻り値

正常に終了した場合は MGE_S_OKを、指定したストリームが無効な場合はMGE_INVALIDVSTREAMを返す。それ以外の場合はエラーコードを返す。

補足

*Effectが加えられるのはMGEChannel1につながっているストリームのみ。
*Power Movie PCIドライバ Ver1.00、またはVer1.01 使用時、MGEChannel2につながっているストリームを指定してもエラーを返さない。

MGE_VStreamGetEffect

ストリーム上の現在の特殊効果を取得する。

書式

〈C/C++〉

```
MGEAPI MGERESULT WINAPI
MGE_VStreamGetEffect(HMGESTREAM hmgestmStream, LPDWORD lpdwEffect);
```

〈Basic〉

```
Declare Function MGE_VStreamGetEffect Lib "mge.dll" (ByVal hmgestmStream As Long,
lpdwEffect As Long) As Long
```

引数

hmgestmStream	特殊効果を取得するストリームハンドル
lpdwEffect	現在の特殊効果を返すアドレス

戻り値

正常に終了した場合、MGE_S_OKを、指定したストリームが無効な場合はMGE_INVALIDVSTREAMを返す。それ以外の場合はエラーコードを返す。

APPENDIX

ファンクション使用時の戻り値一覧

MGEAPI 使用時の戻り値は次のとおりです。

戻り値	意味
MGE_S_OK	正常に終了。
MGE_S_FALSE	ファンクションの実行自体は成功したが、ファンクションの意味的には失敗した。
MGE_S_MEANINGLESS	ファンクションの実行自体は成功したが、実行ファンクションとしては意味なし。
MGE_S_DISABLED	ファンクションの実行自体は成功したが、実行ファンクションとしては無効。
MGE_E_GENERALERROR	その他のエラーが発生。
MGE_E_INVALIDPARAMETER	設定されたパラメータは無効。
MGE_E_NOMEMORY	メモリが不足している。
MGE_E_BUSY	オブジェクト（またはデバイス）がビジー状態である。
MGE_E_NOTEQUIPPED	デバイスが実装されていない。
MGE_E_NORESOURCE	使用するリソースがない。
MGE_E_NOTSUPPORTED	サポートされていないファンクションを実行した。
MGE_E_SURFACE_CONNECTED	サーフェスは既に接続されている。
MGE_E_CHANNEL	無効なチャネルを指定した。
MGE_E_INVALIDSURFACE	無効なサーフェスを指定した。
MGE_E_VSRC_CONNECTED	ビデオ入力オブジェクト（ビデオデコーダオブジェクト）は既に接続されている。
MGE_E_NOT_IMPLEMENT	ファンクションは実装されていない。
MGE_E_INVALIDSTREAM	指定したストリームは無効。
MGE_E_HW_NOTFOUND	Power Movie PCI が見つからない 。
MGE_E_IO	Power Movie PCI が動作していない。

戻り値	意味
MGE_E_VSRC_NOTCONNECTED	ビデオ入力オブジェクト（ビデオデコーダオブジェクト）が接続されていない。
MGE_E_VDST_CONNECTED	ビデオ出力オブジェクト（ビデオエンコーダオブジェクト）は既に接続されている。
MGE_E_VDST_NOTCONNECTED	ビデオ出力オブジェクト（ビデオエンコーダオブジェクト）が接続されていない。
MGE_E_STREAMNOTCOMPLETED	ストリームがサーフェスに結びついていない。
MGE_E_NODIRECTDRAW	DirectDraw によるオーバーレイが行なえない。もしくは DirectDraw がない。
MGE_E_INVALIDSURFACEFORMAT	サーフェスのフォーマット指定が間違っている。
MGE_E_INVALIDRECT	rect の指定が無効。
MGE_E_SUPEROVLSURFACE	Super Overlay 時には実行できない。
MGE_E_FILE	ファイルの入出力で失敗した。
MGE_E_BITMAPFORMAT	ビットマップ形式の指定が無効。
MGE_E_NOJPEGCODEC	JPEG コーデックが見つからない。
MGE_E_JPEGSTREAM	無効な JPEG データ。
MGE_E_JPEGENCODE	JPEG エンコード時にエラーが発生。
MGE_E_DIRECTDRAW	DirectDraw API がエラーを返した。
MGE_E_NOSCLCALLED	MGE_SetCooperativeLevel を呼び出していない。
MGE_E_REGISTRY	指定された Registry にアクセス中にエラーが発生した。
MGE_E_STREAMNOTSTARTED	ストリームが開始されていないか一時停止中である。
MGE_E_UNICODE	UNICODE はサポートしていない。
MGE_E_INVALIDPIXELCOMBINATION	無効な Pixel Format の組を指定した。
MGE_E_INTERNAL	内部エラーです。サポート窓口まで連絡してください。

備考：戻り値は mgeerror.h（Visual C++）/mgeerror.bas（Visual Basic）に定義されています。

定数・定義一覧

信号状態の定義	意味
MGE_SS_STABLE	ビデオ信号が安定して供給されている。 または
MGE_SS_NOTSTABLE	ビデオ信号が安定してないか、供給されていない。
MGE_SS_COLOR_UNKNOWN	ビデオ信号のカラー信号が判別不能。 (MGE_SS_NOTSTABLE 状態の時に返ります。) または
MGE_SS_COLOR_COLOR	ビデオ信号のカラー信号を検出した。 または
MGE_SS_COLOR_MONO	ビデオ信号のカラー信号を検出できないか、白黒信号を検出した。
MGE_SS_TYPE_UNKNOWN	ビデオ信号の信号方式が判別不能。 (MGE_SS_NOTSTABLE 状態の時に返ります。) または
MGE_SS_TYPE_NTSC	ビデオ信号が NTSC 方式。 または
MGE_SS_TYPE_PAL	ビデオ信号が PAL 方式 (Power Movie PCI ではサポートされない)。

カラーキーに使用するための代表的な色の定義	意味
MGE_COLORKEY_BLACK	黒
MGE_COLORKEY_RED	赤
MGE_COLORKEY_GREEN	緑
MGE_COLORKEY_YELLOW	黄色
MGE_COLORKEY_BLUE	青
MGE_COLORKEY_MAGENTA	赤紫
MGE_COLORKEY_CYAN	水色
MGE_COLORKEY_WHITE	白

特殊効果を指定するための定数	意味
MGE_EF_MIRROR_ON	左右反転します。
MGE_EF_MIRROR_OFF	左右反転をしません。
MGE_EF_UD_ON	上下反転をします。
MGE_EF_UD_OFF	上下反転をしません。
MGE_EF_BALL_ON	Power Movie PCI ではサポートされません。
MGE_EF_BALL_OFF	Power Movie PCI ではサポートされません。

備考：各定数は or(|) で複数指定できます。xx_ON と xx_OFF を同時に指定した場合は、xx_ON を指定したのと同じ扱いとなります。
例えば、(MGE_EF_MIRROR_ON|MGE_EF_MIRROR_OFF) は MGE_EF_MIRROR_ON となります。

データ構造解説

PixelFormat の詳細は次のとおりです。

8bit パレット (MGEPixelFormatPAL8)

7 6 5 4 3 2 1 0

Palette Index

*Palette データへのインデックスです。

RGB555 (MGEPixelFormatRGB15)

1 1 1 1 1 1
5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0

U	Red	Green	Blue

＊U は未定義ビットです。

RGB565 (MGEPixelFormatRGB16)

1 1 1 1 1 1
5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0

Red	Green	Blue

RGB24 (MGEPixelFormatRGB24)

2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1
3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0

Red	Green	Blue

RGB32 (MGEPixelFormatRGB32)

3 3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1
1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0

Undefined	Red	Green	Blue

YUY2 (MGEPixelFormatYUY2)

3 3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1
1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0

V	Y1	U	Y0

2pixel 分で色を表現します。

Y0　1pixel 目の輝度信号
Y1　2pixel 目の輝度信号
U　輝度信号と赤色成分の差
V　輝度信号と青色成分の差

用語解説

オーバーレイ（Overlay）

Power Movie PCI ではある画面とある画面を重ね合わせることをオーバーレイすると表現します。通常キーカラーを使用して重ね合わせる方法（キーカラーに相当する部分を別のデータで表示します。DirectDraw Overlay サーフェスがこれに相当します）で行うことを指します。

Power Movie PCI では DirectDraw Overlay の他にグラフィックボードに直接外部入力を表示する Super Overlay も用意しています。Super Overlay はキーカラーは使用できませんが、Window handle の矩形ベースで表示することができます。

DirectDraw Overlay

Microsoft DirectDraw の機能を使用してオーバーレイを実現します。キーカラーが使用できます。

Super Overlay

グラフィックボードに直接外部入力映像を表示しオーバーレイを実現します。キーカラーは使用できませんが、Window handle で表される矩形のクリッピングが可能です。

サーフェス

イメージを保持する領域のことを表し、３種類のサーフェスがあります。

Super Overlay サーフェス

グラフィックボード画面上にサーフェスとして領域を確保します。

グラフィックボード画面に作成されるため、このサーフェスは RGB 系のサーフェスとなります。接続できる Video デコーダーのチャンネルは MGEChannel1 で作成されたもののみとなります。詳しくは MGE_VStreamConnect を参照ください。

DirectDraw Overlay サーフェス

DirectDraw のオーバーレイ機能を利用し、オフスクリーン（グラフィックボードの画面外）にサーフェスとして領域を確保します。

DirectDraw Overlay サーフェスは YUV 系のサーフェスとなります。接続できる Video デコーダーのチャンネルは MGEChannel2 で作成されたもののみとなります。詳しくは MGE_VStreamConnect を参照ください。

このサーフェスはキーカラーを使用することによってグラフィックボード上にオーバーレイ表示することができます。

メモリサーフェス

メインメモリ上に領域を確保します。入力（Video デコーダー）から静止画を得るために使用したり、出力（Video エンコーダー）への表示のために使用します。

ストリーム

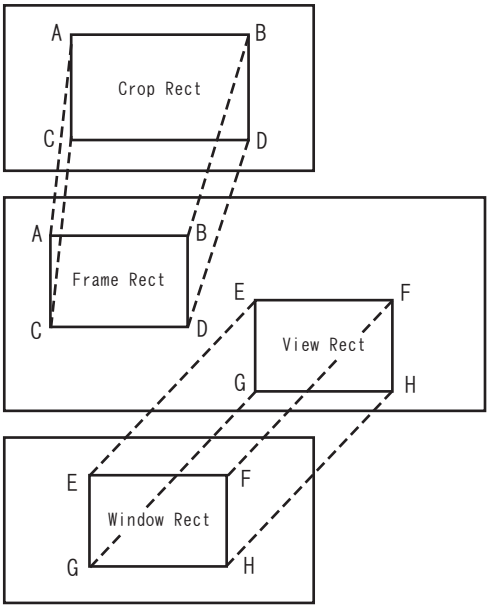
ビデオデータの流れを表す仮想的なもの。例えば、ビデオデコーダ（入力）からサーフェスへのデータの流れを結びつけるために使用します。

レクト

レクトは下記４種類のレクト（矩形）が存在します。

Crop Rect	Data Producer（例えば Video デコーダー）の取り込み位置を指定します。
Frame Rect	Crop Rect で指定された矩形を Frame buffer のどこに置くかを指定します。 この時（可能ならば）拡大縮小が行われます。 Power Movie PCI では拡大は行なえません。
View Rect	Frame buffer 上のどこの領域を表示するかを指定します。
Window Rect	View Rect で指定された矩形を Data Consumer（例えば Video エンコーダー）のどこに置くかを指定します。 この時（可能ならば）拡大縮小が行われます。 Power Movie PCI では拡大縮小は行なえません。

下記の図の Crop Rect の A 点、B 点、C 点、D 点は Frame Rect の A 点、B 点、C 点、D 点に対応します。また、View Rect の E 点、F 点、G 点、H 点は Window Rect の E 点、F 点、G 点、H 点に対応します。



サーフェスが Super Overlay の場合は、Frame buffer が存在しませんので Frame Rect と View Rect は指定できません。
（指定すると MGE_S_FALSE が返ります。）
Data Source(Video デコーダー）と Data Consumer(Super Overlay) が直結し、Crop Rect と Window Rect の関係で縮小が行われます。
（拡大はできません。MGE_E_INVALIDRECT が返ります。）

サーフェスが DirectDraw Overlay の場合は、すべての Rect が存在します。

Memory Surface が VideoInStream に接続されている場合 View Rect と Window Rect は指定できません。
（指定すると MGE_S_FALSE が返ります）
Memory Surface が VideoOutStream に接続されている場合 Crop Rect と Frame Rect は指定できません。
（指定すると MGE_S_FALSE が返ります）
現在の Power Movie PCI の実装では VideoInStream と VideoOutStream を Memory Surface 経由で混合することはできません。

INDEX

INDEX

C

M

V