MVR-D4000 Series

Development Kit

Version 1.31

2005.09.09

canopus

■ご購入製品を使用される際の注意事項

ここでは、プログラムを行うハードウェア(MVR-D4000,4400)をご使用になられるときにご注意いただきたい事項について説明しています。ハードウェアのご使用方法や、本注意事項の内容についてご不明な点、疑問点などございましたら、カノープス株式会社テクニカルサポートまでお問い合わせください。

MVR-D4000 Series Development Kit のサポートは、カノープスシステム開発サポート(p. 9 参照)で承っております。

カノープス株式会社
〒651-2241
神戸市西区室谷 1-2-2(神戸ハイテクパーク内)
テクニカルサポート TEL.078-992-6830
(祝祭日および当社指定休日を除く月～金 10:00～12:00　13:00～17:00)

●ご利用目的に関する注意事項

医療機器や人命に関るシステムでは、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

製品の取り付けや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)、機器の金属部分を触っ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(MVR-D4000,D4400)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)。

●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全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越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越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。

●他社製品との併用に関する注意事項

他社製品と併用されるとご購入製品は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、製品単体の環境でご購入製品が正常に動作することをご確認ください。また、他社製品との併用でご購入製品が正常に動作しないのであれば、その他社製品とご購入製品との併用はお止めください。

●その他の注意事項

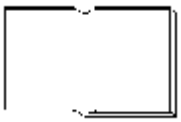
製品は指定された位置に指示通り取り付けてください。指示通りに取り付けられてない場合、製品（MVR-D4000,D4400）の金属部とパソコンの金属部が接触してショートするなどの要因で、製品やパソコン本体・周辺機器が破壊される恐れがあります。

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)本製品付属のソフトウェア、マニュアル、その他添付物を含めたすべての関連品に関して、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。
(6)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。MVR-D4000、MVR-D4400 および MVR-D4000 Series Development Kit はカノープス株式会社の商標です。
(7)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。



表記について

■本書では Microsoft® Windows® operating system を『Windows』、Microsoft® Windows® 2000 operating system、および Microsoft® Windows Server 2003 Standard Edition を『Windows 2000』、『Windows Server 2003』と表記します。
■本書は Windows 上でプログラムを作成することができる方を対象に書かれています。
■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。プログラム上の基本的な事項、プログラムの方法などについては記載していません。
■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も必ずお読みください。

目次

CHAPTER 0 Preliminaries

1. 関数の仕様変更

●**Version 1.00** ファーストリリース。

● **Version 1.00** から **Version 1.10** へのアップデートにより、次の関数について引数の変更や構造体メンバー追加などの仕様が変更になりました。詳細については **CHAPTER3** の各関数についての説明をご覧ください。

関数	変更内容
CDE_SetEncodeFile	引数の **CString strFileName** が **LPCSTR lpszFileName** に変更になりました。
CDE_SetDecodeFile	引数の **CString strFileName** が **LPCSTR lpszFileName** に変更になりました。

● **Version 1.10** から **Version 1.20** へのアップデートにより、次の関数について引数の変更や構造体メンバー追加などの仕様が変更になりました。詳細については **CHAPTER3** の各関数についての説明をご覧ください。

関数	変更内容
CDE_GetEncodeParam **CDE_SetEncodeParam**	**CDE_ENCODEPARAM:: dwFrameSkip** の値を設定できるようになりました。
CDE_Seek	**SEEK** の方法が変更になりました。
CDE_GetPlaybackTime	補足を変更しました。

● **Version 1.20** から **Version 1.30** へのアップデートにより、次の関数について引数の変更や構造体メンバー追加などの仕様が変更になりました。詳細については **CHAPTER3** の各関数についての説明をご覧ください。

関数	変更内容
CDE_GetVideoSetting **CDE_SetVideoSetting**	**CDE_VIDEOPARAM** 構造体にメンバが追加になりました。

＊新規追加された **API** については、ここでは記述しておりません。

2. Visual C++でのアプリケーション開発

●開発環境

この開発キットはマイクロソフト株式会社の **VisualC++6.0**、**VisualC++7.1(Visual Studio.Net 2003)**、**Visual Basic 6.0** 用です。

他の開発ツール（言語処理系）によるアプリケーションのコンパイル、リンクおよび実行結果に関するお問い合わせには十分にお応えできない場合があります。あらかじめご了承ください。

お使いの開発ツール（言語処理系）の設定や付属ユーティリティの使用により、**VisualC++**のインクルードファイルやライブラリを利用する方法については開発ツールのメーカーにお問い合わせください。

● コンパイル

次のファイルをインクルードしてください。

CdeApidef.h

●リンク

次のライブラリをリンクしてください。

VwApi.lib

3. 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

本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、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でお客様の営利を目的としたソフトウェア製品に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。

4. プログラム作成時の注意事項

本開発キットに関するご質問は、インターネット E-mail または FAX でのみ承っております。お電話や NIFTY-Serve 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** **mvrsdk@canopus.co.jp**
カノープスシステム開発サポート FAX:**078-992-4203**

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バージョン情報
   ☆エクスプローラからファイルを右クリックすることで確認できます。
2. 使用している開発環境
   ・ご使用の Windows 環境
   ☆対応 OS 環境は、MVR-D4000,D4400 動作環境に準拠します。
   ・コンパイラのメーカー、バージョン、対応言語
   ・使用されているその他の開発キット
3. 使用している実行環境
   ・PC 本体メーカーおよび機種名
   ・CPU の種類
   ・搭載メモリ容量
   ・ご使用の Windows 環境
   ☆対応 OS 環境は、MVR-D4000,D4400 動作環境に準拠します。
   ・周辺機器メーカーおよび型番

CHAPTER 1 Tutorial

1. アプリケーションの作成

アプリケーションでは

初期化
カードの使用開始
エンコードもしくはデコードの開始
エンコードもしくはデコードの停止
カードの使用終了

などの機能が必要となります。

この項では、ビデオ映像をエンコードもしくはファイルからデコードを行うアプリケーションの作成について説明します。本文中ではエラーチェックは省いてあります。

初期化

初期化の手順です。
ボードの初期化 -> ハンドルの取得 -> コールバック関数の設定 -> デバイスの準備といった
一連のシーケンスについて記述します。

```
INT nBoardNo;

// 使用できる可能性のあるボード番号まで、初期化できるか試みます。
for ( nBoardNo = 0;    nBoardNo <= ENABLE_MAX_BOARD_CNT; nBoardNo++ )
        if ( CDE_Initialize(nBoardNo) == CDE_SUCCESS ) {
                break;
        }
}

if    ( nBoardNo > ENABLE_MAX_BOARD_CNT ) {
        // エラー処理
}
else {
// ボードの番号を保持しておきます。
// この値は終了するときに必要になります。
        m_nBoardNo = nBoardNo;
}

// ハンドルの取得を行ないます。以後このハンドルを使用して制御を行ないます。
// 取得したハンドルは保存しておくか、または制御を行なうたびに毎回この関数を呼び
// 出してください。
if ( CDE_GetCodecHandle( nBoardNo, &m_hHandle) == CDE_SUCCESS) {
        CDECALLBACKSTRU CallBackStr;

        CallBackStr.dwSize = sizeof(CDECALLBACKSTRU);
// 各コールバック関数が呼ばれたとき第一引数に設定される任意の値を設定します。
        CallBackStr.pParam = nBoardNo;
// ステータスが変化したときに呼ばれるコールバック関数を設定します。
        CallBackStr.StatusCallback = StatusCallBack;
// エラーが発生したときに呼ばれるコールバック関数を設定します。
```

```
        CallBackStr.ErrorStatusCallback = ErrCallBack;
// メモリ渡しでのエンコードを行なうときに呼び出されるコールバック関数を設定します。
// 設定の必要がない場合は、NULL を設定してください。
        CallBackStr.EncodeCallback = EncodeCallBack;
// メモリ渡しでのデコードを行なうときに呼び出されるコールバック関数を設定します。
// 設定の必要がない場合は、NULL を設定してください。

        CallBackStr.DecodeCallback = DecodeCallBack;
// コールバック関数の設定をおこないます。
        if ( CDE_SetCallBackFunction( m_hHandle, &CallBackStr) != CDE_SUCCESS) {
                // エラー処理
        }
        else {
// デバイスの準備を行ないます。
                CDE_PrepareDevice( m_hHandle, CDE_PS_MODE) ;
        }
}
```

エンコードの開始

エンコードを開始します。

ファイルへのエンコード時の例

```
//   m_hHandle は初期化時に保存しておいたハンドル。
if ( CDE_SetEncodeFile( m_hHandle, strFileName) ! = CDE_SUCCESS) {
        // エラー処理
}
// ファイルモードでエンコードを開始する。
CDE_StartEncode(m_hHandle, CDE_ENC_FILE);
```

メモリ転送時の例

```
// メモリ転送に使用するバッファを設定します。
CDE_BSSPARAM    BssParam;
```

```
BssParam.dwSize = sizeof(CDE_BSSPARAM);
CDE_GetEncodeBssParam(m m_hHandle, &BssParam);
BssParam.dwFlg = -1;
BssParam. nBufSize = 8 * 1024;
BssParam. nBufCount = 16;
// 設定します。
CDE_SetEncodeBssParam(m_hHandle, &BssParam);
// メモリモードでエンコードを開始する。
CDE_StartEncode(m_hHandle, CDE_ENC_MEMORY);

// メモリ転送用コールバック関数
VOID CALLBACK EncodeCallBack(VOID* pParam, LPBYTE lpBuf, DWORD dwSize)
{
        // pParam は初期化時に CallBackStr.pParam に設定した値です。
        // lpBuf エンコードデータ
        // lpBuf の有効なデータのサイズ。基本的には、CDE_SetEncodeBssParamで
        // 設定した dwBufSize です。
        DWORD dwWriteByte;
        WriteFile( g_hFile, lpBuf, dwSize, &dwWriteByte, NULL)
}
```

エンコードの停止

エンコードを停止します。

```
//  m_hHandle は初期化時に保存しておいたハンドル。
CDE_StopEncode(m_hHandle);
```

デコードの開始

デコードを開始します。

ファイル再生時の例

```
//  m_hHandle は初期化時に保存しておいたハンドル。
if ( CDE_SetDecodeFile( m_hHandle, strFileName) ! = CDE_SUCCESS) {
        // エラー処理
```

```
}
// ファイルモードでデコードを開始する。
CDE_StartDecode(m_hHandle, CDE_DEC_FILE);
```

メモリ転送時の例

```
// メモリ転送に使用するバッファを設定します。
CDE_BSSPARAM    BssParam;

BssParam.dwSize = sizeof(CDE_BSSPARAM);
CDE_GetDecodeBssParam(m m_hHandle, &BssParam);
BssParam.dwFlg = -1;
BssParam. nBufSize = 8 * 1024;
BssParam. nBufCount = 16;
// 設定します。
CDE_SetDecodeBssParam(m_hHandle, &BssParam);
// メモリモードでデコードを開始する。
CDE_StartDecode (m_hHandle, CDE_DEC_MEMORY);


VOID CALLBACK DecodeCallBack(VOID* pParam, LPBYTE lpBuf,
                             LPDWORD lpdwBuffUsed, DWORD dwBuffSize)
{
        // pParam は初期化時に CallBackStr.pParam に設定した値です。
        // lpBuf デコードすべきデータ
        // lpdwBuffUsed の有効なデータのサイズを設定します。
        // dwBuffSize 基本的には、CDE_SetDecodeBssParam で設定した nBufSize です。
        // このサイズを超えてデコードするデータをコピーしないでください。
        ReadFile( g_hFile, lpBuf, dwBuffSize, lpdwBuffUsed, NULL);
}
```

デコードの停止

デコードを停止します。

```
//   m_hHandle は初期化時に保存しておいたハンドル。
CDE_StopDecode(m_hHandle);
```

終了

処理を終了します。

```
// m_nBoardNo は初期化の時に保持していた値です。
CDE_Terminate(m_nBoardNo);
```

CHAPTER 2 Quick Reference

§1. Video Codec Functions

CDE_GeDeviceCount	MVR-D4000 シリーズのデバイスの数を調べます。
CDE_GetDeviceConfig	MVR-D4000 シリーズのデバイス情報を取得します。
CDE_CanInitialize	初期化できるか調べます。
CDE_Initialize	初期化を行ないます。
CDE_Terminate	コントロールを終了します。
CDE_GetCodecHandle	ボードをコントロールするためのハンドルを取得します。
CDE_SetCallBakFunction	コールバック関数の設定を行ないます。
CDE_SetMesForStatus	ステータス情報を取得するためのメッセージ情報を設定します。
CDE_PrepareDevice	指定したモードに従ってデバイスの準備をします。
CDE_GetCurrentStatus	現在のステータスを取得します。
CDE_SetEncordFile	エンコードを行うファイル名を設定します。
CDE_StartEncode	エンコードを開始します。
CDE_StopEncode	エンコードを停止します。
CDE_PauseEncode	エンコードのポーズを行ないます。
CDE_ResumeEncode	エンコードのレジュームを行ないます。
CDE_GetEncodeTime	エンコードを行った時間を取得します。
CDE_GetEncodeFrame	エンコードされたフレーム数を取得します。
CDE_SetRecodeFrame	エンコードするフレーム数を設定します。
CDE_SetRecodeFrame2	エンコードするフレーム数を設定します。
CDE_DetectVideoInputSource	ビデオ入力ソースを検出します。
CDE_SetDecodeFile	デコードするファイルを設定します。
CDE_StartDecode	デコードを開始します。
CDE_Seek	再生を開始する位置を指定します。
CDE_PauseDecode	デコードを一時停止します。
CDE_ResumeDecode	デコードを再開します。
CDE_GetRepeat	リピート状態を取得します。
CDE_SetRepeat	リピート状態を設定します。
CDE_GetPlaybackTime	再生時間を取得します。
CDE_StopDecode	デコードを停止します。
CDE_GetDecodeTime	現在デコードを行っている位置の時間を取得します。
CDE_CanOverlay	オーバーレイウィンドウを使用できるかどうか判断します。
CDE_CreateOverlay	オーバーレイウィンドウを生成します。
CDE_DestroyOverlayWindow	オーバーレイウィンドウを破棄します。
CDE_MoveOverlayWindow	オーバーレイウィンドウを移動します。
CDE_ResizeOverlayWindow	オーバーレイウィンドウのサイズを変更します。
CDE_GetOverlayParameter	オーバーレイ表示パラメータを取得します。

CDE_SetOverlayParameter	オーバーレイ表示パラメータを設定します。
CDE_GetOverlayRect	オーバーレイウィンドウの表示領域を取得します。
CDE_SetOverlayRect	オーバーレイウィンドウの表示領域を設定します。
CDE_GetBitmapBits	静止画(**BMP**)をバッファで取得します。
CDE_SaveDIB	静止画(**BMP**)をファイルに保存します。
CDE_SaveJPEG	静止画(**JPEG**)をファイルに保存します。
CDE_GetEncodeBssParam	設定されているメモリ転送エンコード用のパラメータを取得します。
CDE_SetEncodeBssParam	メモリ転送エンコード用のパラメータを設定します。
CDE_GetEncodeParam	エンコードパラメータを取得します。
CDE_SetEncodeParam	エンコードパラメータを設定します。
CDE_GetVideoSetting	ビデオの設定を取得します。
CDE_SetVideoSetting	ビデオの設定を行います。
CDE_GetAudioSetting	オーディオの設定を取得します。
CDE_SetAudioSetting	オーディオの設定を行います。
CDE_GetAudioEncVol	エンコード時の音声レベルを取得します。
CDE_SetAudioEncVol	エンコード時の音声レベルを設定します。
CDE_GetTsEncInfParam	トランスポートストリーム用の追加設定を取得します。
CDE_SetTsEncInfParam	トランスポートストリーム用の追加設定を設定します。
CDE_GetEncodeParam2	拡張エンコードパラメータを取得します。
CDE_SetEncodeParam2	拡張エンコードパラメータを設定します。
CDE_GetMDSetting	モーションデテクションの設定を取得します。
CDE_SetMDSetting	モーションデテクションの設定を行います。
CDE_GetDecodeBssParam	メモリ転送デコード用のパラメータを取得します。
CDE_SetDecodeBssParam	メモリ転送デコード用のパラメータを設定します。
CDE_GetDecodeSetting	デコードのパラメータを取得します。
CDE_SetDecodeSetting	デコードのパラメータを設定します。
CDE_GetPciBusInf	**PCI** バスの情報を取得します。
CDE_GetIreVal	**NTSC** セットアップレベルを取得します。
CDE_SetIreVal	**NTSC** セットアップレベルを設定します。

CHAPTER 3 Video Codec Functions

CDE_GetDeviceCount

MVR-D4000 シリーズのデバイスの数を調べます。

書式

CDEAPI CDE_GetDeviceCount(LPDWORD pdwCount);

引数

pdwCount	取得したデバイスの数を格納するための **DWORD** 値へのポインタ

戻り値

成功	**CDE_SUCCESS**
失敗	**CDE_SUCCESS** 以外の値

CDE_GetDeviceConfig

MVR-D4000 シリーズのデバイス情報を取得します。

書式

CDEAPI CDE_GetDeviceConfig(INT nBoardNo, CDECONFIG* pParam);

引数

nBoardNo 情報を取得するボードの番号

pParam デバイスの情報が格納された **CDECONFIG** 構造体

CDECONFIG 構造体の定義

デバイスの情報。

```
typedef struct {
        DWORD dwSize;
        INT    nUsage;
        DWORD dwKind
} CDECONFIG;
```

dwSize **この構造体のサイズ。** sizeof(CDECONFIG)を設定してください。
pUsage **デバイスの使用状況。**
dwKind **ボードの種類。**

値	意味
CDE_KIND_D4000	MVR-D4000
CDE_KIND_D4400	MVR-D4400

戻り値

成功 **CDE_SUCCESS**
失敗 **CDE_SUCCESS** 以外の値

CDE_CanInitialize

初期化できるか調べます。

書式

CDEAPI CDE_CanInitalize(INT nBoardNo);

引数

nBoardNo	使用するボードの番号

戻り値

成功	**CDE_SUCCESS**
失敗	**CDE_SUCCESS 以外の値**

参照

CDE_Initialize

CDE_Initialize

初期化を行ないます。

書式

CDEAPI CDE_Initalize(INT nBoardNo);

引数

nBoardNo	使用するボードの番号

戻り値

成功	**CDE_SUCCESS**
失敗	**CDE_SUCCESS 以外の値**

CDE_Terminate

コントロールを終了します。

書式

CDEAPI CDE_Terminate(INT nBoardNo);

引数

nBoardNo	コントロールを終了させるボードの番号

戻り値

成功	**CDE_SUCCESS**
失敗	**CDE_SUCCESS** 以外の値

CDE_GetCodecHandle

ボードをコントロールするためのハンドルを取得します。

書式

CDEAPI CDE_GetCodecHandle(INT nBoardNo, CDEHANDLE* pCodecHandle);

引数

nBoardNo	ハンドルを取得するボードの番号
pCodecHandle	取得したハンドル

戻り値

成功	**CDE_SUCCESS**
失敗	**CDE_SUCCESS** 以外の値

補足

- ボードのコントロールは、この関数で取得したハンドルを使用して行われます。

CDE_SetCallBackFunction

コールバック関数の設定を行ないます。

書式

CDEAPI CDE_SetCallBackFunction(CDEHANDLE hHandle,
CDECALLBACKSTRU* pCallBackStr);

引数

hHandle	ボードを識別するためのハンドル
pCallBackStr	コールバック関数の情報が格納された **CDECALLBACKSTRU** 構造体

戻り値

成功	**CDE_SUCCESS**
失敗	**CDE_SUCCESS** 以外の値

CDECALLBACKSTRU 構造体の定義

コールバック関数の情報。

```
typedef struct {
    DWORD dwSize;
    LPVOID pParam;
    CDE_STATUS_CALLBACK StatusCallback;
    CDE_STATUS_CALLBACK ErrorStatusCallback;
    CDE_ENCODE_CALLBACK EncodeCallback;
    CDE_DECODE_CALLBACK DecodeCallback;
} CDECALLBACKSTRU;
```

dwSize	**この構造体のサイズ。 sizeof(CDECALLBACKSTRU)を設定してください。**
pParam	**ここで設定した値がコールバック関数の一つ目の引数にセットされます。**
StatusCallback	**ステータスに変化があった時に呼ばれます。**
ErrorStatusCallback	**エラーが発生した時に呼ばれます。**
EncodeCallback	**エンコードデータ転送用コールバック関数を設定します。**
DecodeCallback	**デコードデータ転送用コールバック関数を設定します。**

CDE_STATUS_CALLBACK コールバック関数の定義

エラー情報もしくはステータス情報を取得するためのコールバック関数

```
VOID CALLBACK StatusProc(VOID* pParam, DWORD dwStatus);
```

pParam	**CDECALLBACKSTRU::pParam に設定した値**
dwStatus	**ステータスコード**

CDE_ENCODE_CALLBACK コールバック関数の定義

エンコードデータを取得するためのコールバック関数

VOID CALLBACK EncProc(VOID* pParam, LPBYTE lpBuf, DWORD dwSize);

pParam	**CDECALLBACKSTRU::pParam に設定した値**
lpBuf	**エンコードデータが格納されているバッファへのポインタ**
dwSize	**エンコードデータのサイズ**

CDE_DECODE_CALLBACK コールバック関数の定義

デコードデータを取得するためのコールバック関数

VOID CALLBACK DecProc(VOID* pParam, LPBYTE lpBuf,
LPDWORD lpdwBuffUsed, DWORD dwBuffSize);

pParam	**CDECALLBACKSTRU::pParam に設定した値**
lpBuf	**デコードデータを格納するバッファへのポインタ**
lpdwBuffUsed	**lpBuf にデコードデータを格納したサイズ。**
dwBuffSize	**デコードデータを格納するバッファ(lpBuf)の実際のサイズ。このサイズまでデータを格納できます。**

補足

- 関数呼び出し時のエラーは戻り値で判断してください。
- **Encode か Decode** どちらかしか使用しない場合は使用しない方のコールバック関数へのポインタには **NULL** を指定してください。
- エンコードのコールバック関数は、**CDE_StartEncode で CDE_ENC_MEMORY** を指定したときのみ有効です。
- デコードのコールバック関数は、**CDE_StartDecode で CDE_DEC_MEMORY** を指定したときのみ有効です。
- **Visual Basic 6.0** では使用できません。**CDE_SetMesForStatus** を使用してください。

参照

CDE_StartEncode、CDE_StartDecode、CDE_SetMesForStatus

CDE_SetMesForStatus

ステータス情報を取得するためのメッセージ情報を設定します。

書式

CDEAPI CDE_SetMesForStatus(CDEHANDLE hHandle, CDESTAMESSTRU* pStatusMes);

引数

hHandle	ボードを識別するためのハンドル
pStatusMes	メッセージの情報が格納された **CDESTAMESSTRU** 構造体

戻り値

成功	**CDE_SUCCESS**
失敗	**CDE_SUCCESS** 以外の値

CDESTAMESSTRU 構造体の定義

メッセージの情報。

```
typedef struct {
    DWORD dwSize;
    WPARAM wParam;
    HWND hWnd;
    UINT uStatusMes;
    UINT uErrMes;
} CDESTAMESSTRU;
```

dwSize	**この構造体のサイズ。** sizeof(**CDESTAMESSTRU**)を設定してください。
wParam	**メッセージの最初のパラメータにセットされます。**
hWnd	**メッセージを受け取るウィンドウのハンドル。**
uStatusMes	**ステータスに変化があった時に送られるメッセージ。**
uErrMes	**エラーが発生した時に送られるメッセージ**

補足

- 関数呼び出し時のエラーは戻り値で判断してください。
- この関数は、**CDE_SetCallBackFunction** と同時に使用できません。

CDE_PrepareDevice

指定したモードに従ってデバイスの準備をします。

書式

CDEAPI CDE_PrepareDevice(CDEHANDLE hHandle, INT nDeviceMode);

引数

hHandle ボードを識別するためのハンドル
nDeviceMode **MODE** を指定します。

値	意味
CDE_PS_MODE	デバイスをプログラムストリーム用で初期化します。(チップ内にプログラムストリーム用のファームウェアをダウンロードします)
CDE_TS_MODE	デバイスをトランスポートストリーム用で初期化します。(チップ内にトランスポートストリーム用のファームウェアをダウンロードします。

戻り値

成功 **CDE_SUCCESS**
失敗 **CDE_SUCCESS** 以外の値

CDE_GetCurrentStatus

現在のステータスを取得します。

書式

CDEAPI CDE_GetCurrentStatus(CDEHANDLE hHandle, LPDWORD lpdwStatus);

引数

hHandle	ボードを識別するためのハンドル
lpdwStatus	ステータス情報を格納するための **DWORD** 値へのポインタ

戻り値

成功	**CDE_SUCCESS**
失敗	**CDE_SUCCESS** 以外の値

CDE_SetEncodeFile

エンコードを行うファイル名を設定します。

書式

CDEAPI CDE_SetEncodeFile(CDEHANDLE hHandle, LPCSTR lpszFileName);

引数

hHandle	ボードを識別するためのハンドル
lpszFileName	エンコードを行うファイル名

戻り値

成功	**CDE_SUCCESS**
失敗	**CDE_SUCCESS** 以外の値

補足

● エンコードのモードが **CDE_ENC_FILE** の時のみこの関数で設定した値は有効です。

参照

CDE_StartEncode

CDE_StartEncode

エンコードを開始します。

書式

CDEAPI CDE_StartEncode(CDEHANDLE hHandle, INT nEncodeMode);

引数

hHandle ボードを識別するためのハンドル
nEncodeMode エンコードのモードを指定します。

値	意味
CDE_ENC_FILE	エンコードしたデータを **CDE_SetEncodeFile** で指定したファイルで取得します。
CDE_ENC_MEMORY	エンコードしたデータをメモリで取得します。（**CDE_SetCallBackFunction** で指定したコールバック関数でエンコードしたデータを取得します）**Visaul Basic 6.0** ではこのモードは使用できません。

戻り値

成功 **CDE_SUCCESS**
失敗 **CDE_SUCCESS** 以外の値

参照

CDE_SetEncodeFile, CDE_SetCallBackFunction

CDE_StopEncode

エンコードを停止します。

書式

CDEAPI CDE_StopEncode(CDEHANDLE hHandle);

引数

hHandle ボードを識別するためのハンドル

戻り値

成功	**CDE_SUCCESS**
失敗	**CDE_SUCCESS** 以外の値

CDE_PauseEncode

エンコードのポーズを行ないます。

書式

CDEAPI CDE_PauseEncode(CDEHANDLE hHandle);

引数

hHandle ボードを識別するためのハンドル

戻り値

成功 **CDE_SUCCESS**
失敗 **CDE_SUCCESS** 以外の値

補足

一時停止点の不連続な音声データは再生時に音声ノイズとなる場合があります。

CDE_ResumeEncode

エンコードのレジュームを行ないます。

書式

CDEAPI CDE_ResumeEncode(CDEHANDLE hHandle);

引数

hHandle	ボードを識別するためのハンドル

戻り値

成功	**CDE_SUCCESS**
失敗	**CDE_SUCCESS 以外の値**

参照

CDE_PauseEncode

CDE_GetEncodeTime

エンコードを行った時間を取得します。

書式

CDEAPI CDE_GetEncodeTime(CDEHANDLE hHandle, LONGLONG* pllTime);

引数

hHandle	ボードを識別するためのハンドル
pllTime	**時間情報が格納されます。(単位:ミリ秒)**

戻り値

成功	**CDE_SUCCESS**
失敗	**CDE_SUCCESS 以外の値**

補足

- エンコーダデバイスとは別にソフトウェアで計算しているため実際に出来上がったファイルのトータル時間とは誤差が発生します。

CDE_GetEncodeFrame

エンコードされたフレーム数を取得します。

書式

CDEAPI CDE_GetEncodeFrame(CDEHANDLE hHandle, LONGLONG* pllFrame);

引数

hHandle	ボードを識別するためのハンドル
pllFrame	エンコードされたフレーム数を格納するための **LONGLONG** 変数へのポインタ

戻り値

成功	**CDE_SUCCESS**
失敗	**CDE_SUCCESS** 以外の値

CDE_SetRecordFrame

エンコードするフレーム数を設定します。

書式

CDEAPI CDE_SetRecordFrame(CDEHANDLE hHandle, LONGLONG llFrame);

引数

hHandle	ボードを識別するためのハンドル
llFrame	エンコードするフレーム数

戻り値

成功	**CDE_SUCCESS**
失敗	**CDE_SUCCESS** 以外の値

補足

- 実際にエンコードされるフレーム数は、**PS : M** の倍数、**TS : 1GOP** 当たりのピクチャ数の倍数、になります。
- **Visual Basic 6.0** では使用できません。**CDE_SetRecordFrame2** を使用してください。

参照

CDE_SetRecordFrame2

CDE_SetRecordFrame2

エンコードするフレーム数を設定します。

書式

CDEAPI CDE_SetRecordFrame2(CDEHANDLE hHandle,
DWORD dwUpper, DWORD dwLower);

引数

hHandle	ボードを識別するためのハンドル
dwUpper	エンコードするフレーム数（上位 **32** ビット）
dwLower	エンコードするフレーム数（下位 **32** ビット）

戻り値

成功	**CDE_SUCCESS**
失敗	**CDE_SUCCESS** 以外の値

補足

● 実際にエンコードされるフレーム数は、**M** の倍数になります。

CDE_DetectVideoInputSource

ビデオ入力ソースを検出します。

書式

CDEAPI CDE_DetectVideoInputSource(CDEHANDLE hHandle,
LPDWORD lpdwInputSource);

引数

hHandle ボードを識別するためのハンドル
lpdwInputSource **ビデオ入力ソースの情報が格納されます。**

値	意味
CDE_STANDARD_SVIDEO	**S-VIDEO**
CDE_STANDARD_COMPOSITE	**COMPOSIT**
CDE_STANDARD_UNKNOWN	入力ソースなし

戻り値

成功 **CDE_SUCCESS**
失敗 **CDE_SUCCESS 以外の値**

CDE_SetDecodeFile

デコードするファイルを設定します。

書式

CDEAPI CDE_SetDecodeFile(CDEHANDLE hHandle, LPCSTR lpszFileName);

引数

hHandle	ボードを識別するためのハンドル
lpszFileName	デコードするファイル名

戻り値

成功	**CDE_SUCCESS**
失敗	**CDE_SUCCESS** 以外の値

補足

● デコードのモードが **CDE_DEC_FILE** の時のみこの関数で設定した値は有効です。

参照

CDE_StartDecode

CDE_StartDecode

デコードを開始します。

書式

CDEAPI CDE_StartDecode(CDEHANDLE hHandle, INT nDecodeMode);

引数

hHandle ボードを識別するためのハンドル
nDecodeMode デコードのモードを指定します。

値	意味
CDE_DEC_FILE	**CDE_SetDecodeFile** で指定したファイルをデコードします。
CDE_DEC_MEMORY	バッファ単位でデコードするデータを渡します。（**CDE_SetCallBackFunction** で指定したコールバック関数でデコードしたデータを渡します）**Visual Basic 6.0** ではこのモードは使用できません。

戻り値

成功 **CDE_SUCCESS**
失敗 **CDE_SUCCESS** 以外の値

参照

CDE_SetDecodeFile, CDE_SetCallBackFunction

CDE_Seek

再生を開始する位置を指定します。

書式

CDEAPI CDE_Seek(CDEHANDLE hHandle, DWORD dwPosition);

引数

hHandle	ボードを識別するためのハンドル
dwPosition	再生開始位置（単位:ミリ秒）

戻り値

成功	**CDE_SUCCESS**
失敗	**CDE_SUCCESS** 以外の値

補足

- **SEEK** にはプログラムストリームはPTSを、トランスポートストリームは **PCR** 利用します。
- **24** 時間を超えるファイルの **SEEK** は動作保証外になります。
- 再生位置に関しては、指定された位置を超えないもっとも近い位置が再生開始位置となります。

CDE_PauseDecode

デコードを一時停止します。

書式

CDEAPI CDE_PauseDecode(CDEHANDLE hHandle);

引数

hHandle ボードを識別するためのハンドル

戻り値

成功	**CDE_SUCCESS**
失敗	**CDE_SUCCESS** 以外の値

CDE_ResumeDecode

デコードを再開します。

書式

CDEAPI CDE_ResumeDecode(CDEHANDLE hHandle);

引数

hHandle ボードを識別するためのハンドル

戻り値

成功	**CDE_SUCCESS**
失敗	**CDE_SUCCESS** 以外の値

CDE_GetRepeat

リピート状態を取得します。

書式

CDEAPI CDE_GetRepeat(CDEHANDLE hHandle, BOOL* fRepeat);

引数

hHandle	ボードを識別するためのハンドル
fRepeat	**リピートの状態を格納する BOOL 型変数へのポインタ**

戻り値

成功	**CDE_SUCCESS**
失敗	**CDE_SUCCESS 以外の値**

CDE_SetRepeat

リピート状態を設定します。

書式

CDEAPI CDE_SetRepeat(CDEHANDLE hHandle, BOOL fRepeat);

引数

hHandle	ボードを識別するためのハンドル
fRepeat	設定するリピート状態

戻り値

成功	**CDE_SUCCESS**
失敗	**CDE_SUCCESS** 以外の値

補足

● デコードのモードが **CDE_DEC_FILE** の時のみこの関数で設定した値は有効です。

参照

CDE_StartDecode

CDE_GetPlaybackTime

再生時間を取得します。

書式

CDEAPI CDE_GetPlaybackTime(CDEHANDLE hHandle, LPDWORD lpdwTime);

引数

hHandle	ボードを識別するためのハンドル
lpdwTime	**再生時間（単位：ミリ秒）を格納する DWORD 型変数へのポインタ**

戻り値

成功	**CDE_SUCCESS**
失敗	**CDE_SUCCESS 以外の値**

補足

- **CDE_SetDecodeFile** で指定したファイルについての再生時間を取得します。
- **CDE_Seek** により再生位置の指定可能なファイルについての再生時間を取得することができます。

参照

CDE_SetDecodeFile、CDE_Seek

CDE_StopDecode

デコードを停止します。

書式

CDEAPI CDE_StopDecode(CDEHANDLE hHandle);

引数

hHandle	ボードを識別するためのハンドル

戻り値

成功	**CDE_SUCCESS**
失敗	**CDE_SUCCESS** 以外の値

CDE_GetDecodeTime

現在デコードを行っている位置の時間を取得します。

書式

CDEAPI CDE_GetDecodeTime (CDEHANDLE hHandle, LONGLONG* pllTime);

引数

hHandle	ボードを識別するためのハンドル
pllTime	**時間情報が格納されます。(単位:ミリ秒)**

戻り値

成功	**CDE_SUCCESS**
失敗	**CDE_SUCCESS 以外の値**

補足

- デコーダデバイスとは別にソフトウェアで計算しているため実際に出力されている時間とは誤差が発生します。

CDE_CanOverlay

オーバーレイウィンドウを使用できるかどうか判断します。

書式

CDEAPI CDE_CanOverlay (CDEHANDLE hHandle, HANDLE hMonitor);

引数

hHandle	ボードを識別するためのハンドル
hMonitor	**モニタハンドル（予約 : NULL）**

戻り値

成功	**CDE_SUCCESS**
失敗	**CDE_SUCCESS 以外の値**

CDE_CreateOverlay

オーバーレイウィンドウを生成します。

書式

CDEAPI CDE_CreateOverlay (CDEHANDLE hHandle, HWND hWndParent,
HWND* hWndOverlay, INT x, INT y, UINT nWidth, UINT nHeight);

引数

hHandle	ボードを識別するためのハンドル
hWndParent	オーバーレイウィンドウを生成するウィンドウのハンドル
hWndOverlay	オーバーレイウィンドウのハンドルを取得するアドレス
x	オーバーレイウィンドウの位置(左)
y	オーバーレイウィンドウの位置(上)
nWidth	オーバーレイウィンドウの幅
nHeight	**オーバーレイウィンドウの高さ**

戻り値

成功	**CDE_SUCCESS**
失敗	**CDE_SUCCESS** 以外の値

補足

● **MVR-D4000** ではオーバーレイは作成できません。

CDE_DestroyOverlayWindow

オーバーレイウィンドウを破棄します。

書式

CDEAPI CDE_DestroyOvelayWindow(CDEHANDLE hHandle);

引数

hHandle	ボードを識別するためのハンドル

戻り値

成功	**CDE_SUCCESS**
失敗	**CDE_SUCCESS** 以外の値

CDE_MoveOverlayWindow

オーバーレイウィンドウを移動します。

書式

CDEAPI CDE_MoveOvelayWindow(CDEHANDLE hHandle, INT x, INT y);

引数

hHandle	ボードを識別するためのハンドル
x	オーバーレイウィンドウの左上隅のスクリーン x 座標
y	オーバーレイウィンドウの左上隅のスクリーン y 座標

戻り値

成功	**CDE_SUCCESS**
失敗	**CDE_SUCCESS** 以外の値

CDE_ResizeOverlayWindow

オーバーレイウィンドウのサイズを変更します。

書式

CDEAPI CDE_ResizeOvelayWindow(CDEHANDLE hHandle, UINT nWidth, UINT nHeight);

引数

hHandle	ボードを識別するためのハンドル
nWidth	オーバーレイウィンドウの幅
nHeight	オーバーレイウィンドウの高さ

戻り値

成功	**CDE_SUCCESS**
失敗	**CDE_SUCCESS** 以外の値

CDE_GetOverlayParameter

オーバーレイ表示パラメータを取得します。

書式

CDEAPI CDE_GetOverlayParameter(CDEHANDLE hHandle,
CDE_OVERLAYSETTING* pParam);

引数

hHandle ボードを識別するためのハンドル
pParam オーバーレイ表示パラメータの情報を格納する
CDE_OVERLAYSETTING 構造体

CDE_OVERLAYSETTTING 構造体の定義

オーバーレイ表示パラメータ。

```
typedef struct {
    DWORD dwSize;
    UINT nOverlayBrightness;
    UINT nOverlayContrast;
    UINT nOverlaySaturation;
} CDE_OVERLAYSETTING;
```

dwSize **この構造体のサイズ。**
sizeof(CDE_OVERLAYSETTING)を設定してください。

nOverlayBrightness **明るさの値が格納されます。**

値	意味
MIN_VIDEO_OVERLAY_BRIGHTNESS	最小値
MAX_VIDEO_OVERLAY_BRIGHTNESS	最大値
DEF_VIDEO_OVERLAY_BRIGHTNESS	デフォルト値

nOverlayContrast **コントラストの値が格納されます。**

値	意味
MIN_VIDEO_OVERLAY_CONTRAST	最小値
MAX_VIDEO_OVERLAY_CONTRAST	最大値
DEF_VIDEO_OVERLAY_CONTRAST	デフォルト値

nOverlaySaturation **色の濃さの値が格納されます。**

値	意味
MIN_VIDEO_OVERLAY_SATURATION	最小値
MAX_VIDEO_OVERLAY_SATURATION	最大値
DEF_VIDEO_OVERLAY_SATURATION	デフォルト値

戻り値

成功	**CDE_SUCCESS**
失敗	**CDE_SUCCESS 以外の値**

CDE_SetOverlayParameter

オーバーレイ表示パラメータを設定します。

書式

CDEAPI CDE_SetOverlayParameter(CDEHANDLE hHandle,
CDE_OVERLAYSETTING* pParam);

引数

hHandle	ボードを識別するためのハンドル
pParam	オーバーレイ表示パラメータの情報を格納する **CDE_OVERLAYSETTING** 構造体

戻り値

成功	**CDE_SUCCESS**
失敗	**CDE_SUCCESS** 以外の値

参照

CDE_GetOverlayParameter

CDE_GetOverlayRect

オーバーレイウィンドウの表示領域を取得します。

書式

CDEAPI CDE_GetOverlayRect(CDEHANDLE hHandle, LPRECT pParam);

引数

hHandle	ボードを識別するためのハンドル
pParam	オーバーレイウィンドウの表示領域情報を格納する**RECT**構造体へのポインタ

戻り値

成功	**CDE_SUCCESS**
失敗	**CDE_SUCCESS** 以外の値

CDE_SetOverlayRect

オーバーレイウィンドウの表示領域を設定します。

書式

CDEAPI CDE_SetOverlayRect(CDEHANDLE hHandle, LPRECT pParam);

引数

hHandle	ボードを識別するためのハンドル
pParam	オーバーレイウィンドウの表示領域情報が格納された **RECT** 構造体へのポインタ 範囲 **NTSC :** 水平位置 **: 1 ～ 720**、垂直位置 **1 ～ 480** **PAL :** 水平位置 **1 ～ 720**、垂直位置 **1 ～ 576**

戻り値

成功	**CDE_SUCCESS**
失敗	**CDE_SUCCESS** 以外の値

CDE_GetBitmapBits

静止画（BMP）をバッファで取得します。

書式

CDEAPI CDE_GetBitmapBits(CDEHANDLE hHandle,
LPBYTE* pByte, LPDWORD lpdwLen, CDESTILLCAPPARAM* pParam);

引数

hHandle	ボードを識別するためのハンドル
pByte	データを格納するバッファへのポインタを格納するポインタ
pdwLen	取得した静止画のバイト数を格納するポインタ
pParam	静止画キャプチャパラメータを格納する **CDESTILLCAPPARAM** 構造体

戻り値

成功	**CDE_SUCCESS**
失敗	**CDE_SUCCESS** 以外の値

CDESTILLCAPPARAM 構造体の定義

静止画キャプチャパラメータ

```
typedef struct {
    DWORD dwSize;
    INT nFrameCap;
    BOOL fExpandRGB;
    double dCorrectGamma;
    CDE_UYVY2RGB pfnuyvy2rgb;
} CDESTILLLCAPPARAM;
```

dwSize **この構造体のサイズ。**
sizeof(CDESTILLLCAPPARAM)を設定してください。

nFrameCap **キャプチャの方法を指定します。**

値	意味
CDE_STILL_FRAME	**1** フレームをキャプチャします。
CDE_STILL_ODD_FIELD	奇数ラインのみ取得し **2** 倍に引き伸ばします。
CDE_STILL_EVEN_FIELD	偶数ラインのみ取得し **2** 倍に引き伸ばします。

fExpandRGB	**TRUE の場合、YUV -> RGB 変換時に、[16-235] の範囲を [0-255] の範囲へ拡大する変換を行ないます。**
dCorrectGamma	**ガンマ補正値を設定します。標準で NTSC の時 2.2 で PAL の時 2.8 です。使用しない時は 1 に設定します。**
pfnuyvy2rgb	**変換を行なう関数を指定できます。指定しない場合は NULL でないといけません。**

CDE_UYVY2RGB 関数の定義

YUV -> RGB に変換するための関数

INT ConvertYUVtoRGB(BYTE u1, BYTE y1, BYTE v1, BYTE y2,
COLORREF *pColor1, COLORREF *pColor2)

u1	**Cb(色差)**
y1	**RGB 1 バイト目用の Y(輝度)**
v1	**Cr(色差)**
y2	**RGB 2 バイト目用の Y(輝度)**
pColor1	**変換後 RGB 1 バイト目**
pColor2	**変換後 RGB 2 バイト目**

戻り値

必ず1を返す

補足

- **MVR-D4000** では使用できません。
- キャプチャ時に映像の **DMA** の転送を一度停止するため **PC** 上のオーバーレイ映像が一瞬停止します。(外部モニタへ出力されている映像は停止しません)
- 取得できるサイズはソースサイズです。
- 画像の縦のサイズが **480** 以下の場合 **CDESTILLLCAPPARAM:: nFrameCap** の値は無視されます。
- **Visual Basic 6.0** では使用できません。

CDE_SaveDIB

静止画（BMP）をファイルに保存します。

書式

CDEAPI CDE_Save_DIB(CDEHANDLE hHandle, LPCTSTR lpszFileName,
CDESTILLCAPPARAM* pParam);

引数

hHandle	ボードを識別するためのハンドル
lpszFileName	**保存するファイル名**
pParam	静止画キャプチャパラメータを格納する **CDESTILLCAPPARAM** 構造体

戻り値

成功	**CDE_SUCCESS**
失敗	**CDE_SUCCESS** 以外の値

参照

CDE_GetBitmapBits の補足を参照してください。

CDE_SaveJPEG

静止画(JPEG)をファイルに保存します。

書式

CDEAPI CDE_Save_JPEG(CDEHANDLE hHandle, LPCTSTR lpszFileName,
UINT nQuality, CDESTILLCAPPARAM* pParam);

引数

hHandle	ボードを識別するためのハンドル
lpszFileName	**保存するファイル名**
nQuality	**JPEG の圧縮率(0～9の範囲で設定できます)** **0 : 約 1/5** **1 : 約 1/7** **2 : 約 1/10** **3 : 約 1/12** **4 : 約 1/15** **5 : 約 1/17** **6 : 約 1/20** **7 : 約 1/25** **8 : 約 1/30** **9 : 約 1/40**
pParam	静止画キャプチャパラメータを格納する **CDESTILLCAPPARAM** 構造体

戻り値

成功	**CDE_SUCCESS**
失敗	**CDE_SUCCESS** 以外の値

参照

CDE_GetBitmapBits の補足を参照してください。

CDE_GetEncodeBssParam

設定されているメモリ転送エンコード用のパラメータを取得します。

書式

CDEAPI CDE_GetEncodeBssParam(CDEHANDLE hHandle, CDE_BSSPARAM* pParam);

引数

hHandle	ボードを識別するためのハンドル
pParam	メモリ転送エンコード用パラメータの情報を格納する **CDE_BSSPARAM** 構造体

CDE_BSSPARAM 構造体の定義

メモリ転送に関する情報の設定。

```
typedef struct {
    DWORD dwSize;
    INT nBufSize;
    INT nBufCount;
} CDE_BSSPARAM;
```

dwSize	この構造体のサイズ。**sizeof(CDE_BSSPARAM)**を設定します。
nBufSize	確保する(している)バッファのサイズ。
nBufCount	確保する(している)バッファの個数。

戻り値

成功	**CDE_SUCCESS**
失敗	**CDE_SUCCESS** 以外の値

CDE_SetEncodeBssParam

メモリ転送エンコード用のパラメータを設定します。

書式

CDEAPI CDE_SetEncodeBssParam(CDEHANDLE hHandle, CDE_BSSPARAM* pParam);

引数

hHandle	ボードを識別するためのハンドル
pParam	メモリ転送エンコード用パラメータの情報が格納された **CDE_BSSPARAM** 構造体

戻り値

成功	**CDE_SUCCESS**
失敗	**CDE_SUCCESS** 以外の値

補足

● 指定できるバッファのサイズは **0x1000** 単位でしか指定できません。それ以外のサイズを指定した場合は **0x1000** 単位に切り下げられます。

参照

CDE_GetEncodeBssParam

CDE_GetEncodeParam

エンコードパラメータを取得します。

書式

CDEAPI CDE_GetEncodeParam(CDEHANDLE hHandle,
CDE_ENCODEPARAM* pParam);

引数

hHandle ボードを識別するためのハンドル
pParam エンコードパラメータの情報を格納する **CDE_ENCODEPARAM** 構造体

CDE_ENCODEPARAM 構造体の定義

エンコードパラメータの設定。

```
typedef struct {
    DWORD dwSize;
    INT nVideoStreamType;
    DWORD dwHorizontalSizeValue;
    DWORD dwVerticalSizeValue;
    DWORD dwAspectRatioInformation;
    DWORD dwFrameRateCode;
    DWORD dwBitRate;
    DWORD dwVbvBufferSize;
    DWORD dwFrameSkip;
    INT nLowDelay;
    INT nInverseTelecine;
    INT nVbr;
    DWORD dwAverageBitRate;
    DWORD dwN;
    DWORD dwM;
    INT nClosedGop;
} CDE_ENCODEPARAM;
```

dwSize この構造体のサイズ、sizeof(CDE_ENCODEPARAM)を指定します。

nVideoStreamType MPEG の種類が格納されます。(default : CDE_MPEG4)
トランスポートストリームの時は、MPEG2、MPEG4 のみです。

値	意味
CDE_MPEG1	MPEG1
CDE_MPEG2	MPEG2
CDE_MPEG4	MPEG4

dwHorizontalSizeValue ビデオの入力サイズ(幅)が格納されます。設定できるサイズは、次の組み合わせに限定されます。幅(高さ)、720(480,576), 704(480,576), 640(480,576), 480(480,576), 352(480,576,240,288), : (default : 704)

dwVerticalSizeValue ビデオの入力サイズ(高さ)が格納されます。設定できるサイズは次のとおりです。240, 288, 480, 576 : (default=480),

dwAspectRatioInformation アスペクト比の情報が格納されます。
(default : ASPECTRATIO_4_3_FOR_525)

値	意味
ASPECTRATIO_4_3	アスペクト比を 4 : 3 で設定します。 (MPEG1,2 の時のみ有効です。)
ASPECTRATIO_16_9	アスペクト比を 16 : 9 で設定します。 (MPEG1,2 の時のみ有効です。)
ASPECTRATIO_4_3_FOR_625	アスペクト比を 4 : 3 で設定します。放送規格が PAL の時選択します。(MPEG4 の時のみ有効です。)
ASPECTRATIO_4_3_FOR_525	アスペクト比を 4 : 3 で設定します。放送規格が NTSC の時選択します。(MPEG4 の時のみ有効です。)
ASPECTRATIO_16_9_FOR_625	アスペクト比を 16 : 9 で設定します。放送規格が PAL の時選択します。(MPEG4 の時のみ有効です。)
ASPECTRATIO_16_9_FOR_525	アスペクト比を 16 :9 で設定します。放送規格が NTSC の時選択します。(MPEG4 の時のみ有効です。)

dwFrameRateCode フレームレートの情報が格納されます。(default : RATE_29_97_NTSC)

値	意味
RATE_25_PAL	PAL
RATE_29_97_NTSC	29.97fps NTSC

dwBitRate ビデオビットレートの情報が格納されます。(default : 4000000)

MPEG4 : Full D1, VGA 2Mbps ～ 15Mbps
MPEG4 : Half D1, 2/3 D1 1.5Mbps ～ 8Mbps
MPEG4 : SIF 512Kbps ～ 4Mbps
MPEG2 : Full D1, VGA 3Mbps ～ 15Mbps
MPEG2 : Half D1, 2/3 D1 2Mbps ～ 8Mbps
MPEG2 : SIF 1Mbps ～ 4Mbps
MPEG1 : SIF 1Mbps ～ 1.8Mbps

VBR の場合はこの値は max bitrate になります。

	(dwAverageBitRate の倍以上の値を設定するようにしてください。)
dwVbvBufferSize	VBV バッファサイズ。(0 ～ 0x70 、default : 0x70)
dwFrameSkip	スキップさせるフレーム数を指定します。(0 : default, 0 ～ 15) 0 以外を指定する場合は nLowDelay を1に設定してください。 トランスポートストリームの時は 0 を指定してください。
nLowDelay	Low Delay モードを設定します。(0 : disable , 1 : enable, default : 0)
nInverseTelecine	予約 : 0 を設定してください。
nVbr	VBR もしくは CBR の情報が格納されます。 (0 : CBR , 1 : VBR, default : 0) トランスポートストリームの時は、CBR を指定してください。
dwAverageBitRate	VBR での平均ビットレートが格納されます。(default : 3500000)
dwN	gop 内のピクチャ数の情報が格納されます。(1 ～ 30, default : 15) dwM で設定する値以下の値は設定できません。また dwM の整数倍の数値を設定します。
dwM	gop 内の I ピクチャまたは P ピクチャが現れる周期の情報が格納されます。(MPEG1 : 3)、(MPEG2 : 1 ～ 3, default : 1)、(MPEG4 : 1)
nClosedGop	Closed gop の設定状態が格納されます。 (0 : disable , 1 : enable, default : 0)

戻り値

成功	**CDE_SUCCESS**
失敗	**CDE_SUCCESS** 以外の値

CDE_SetEncodeParam

エンコードパラメータを設定します。

書式

CDEAPI CDE_SetEncodeParam(CDEHANDLE hHandle,
CDE_ENCODEPARAM* pParam);

引数

hHandle	ボードを識別するためのハンドル
pParam	設定するエンコードパラメータの情報を格納する **CDE_ENCODEPARAM** 構造体

戻り値

成功	**CDE_SUCCESS**
失敗	**CDE_SUCCESS** 以外の値

参照

CDE_GetEncodeParam

CDE_GetVideoSetting

ビデオの設定を取得します。

書式

CDEAPI CDE_GetVideoSetting(CDEHANDLE hHandle, CDE_VIDEOPARAM* pParam);

引数

hHandle ボードを識別するためのハンドル
pParam ビデオの設定の情報を格納する **CDE_VIDEOPARAM** 構造体

CDE_VIDEOPARAM 構造体の定義

ビデオの設定。

```
typedef struct {
        DWORD dwSize;
        INT nInputTVSystem;
        INT nInputStandard;
        DWORD dwLumaSharpness;
        DWORD dwLumaBrightness;
        DWORD dwLumaContrast;
        DWORD dwChromaSaturation;
        DWORD dwChromaHue;
        DWORD dwPreFilter;
        INT nVideoProcessing;
} CDE_VIDEOPARAM;
```

dwSize この構造体のサイズ。Sizeof(CDE_VIDEOPARAM) を指定。
nInputTVSystem 放送規格の種類が格納されます。
(default : CDE_TVSYSTEM_NTSC)

値	意味
CDE_TVSYSTEM_NTSC	NTSC
CDE_TVSYSTEM_PAL	PAL

nInputStandard 入力ソースの種類が格納されます。
(default : CDE_STANDARD_COMPOSIT)

値	意味
CDE_STANDARD_SVIDEO	S-Video
CDE_STANDARD_COMPOSIT	Composite

dwLumaSharpness　　ビデオ入力の Sharpness の値が格納されます。

値	意味
CDE_SHARPNESS_MIN_RANGE	最小値
CDE_SHARPNESS_MAX_RANGE	最大値

dwLumaBrightness　　ビデオ入力の Brightness の値が格納されます。

値	意味
CDE_BRIGHTNESS_MIN_RANGE	最小値
CDE_BRIGHTNESS_MAX_RANGE	最大値

dwLumaContrast　　ビデオ入力の Contrast の値が格納されます。

値	意味
CDE_CONTRAST_MIN_RANGE	最小値
CDE_CONTRAST_MAX_RANGE	最大値

dwChromaSaturation　　ビデオ入力の Saturation の値が格納されます。

値	意味
CDE_SATURATION_MIN_RANGE	最小値
CDE_SATURATION_MAX_RANGE	最大値

dwChromaHue　　ビデオ入力の Hue の値が格納されます。

値	意味
CDE_HUE_MIN_RANGE	最小値
CDE_HUE_MAX_RANGE	最大値

dwPreFilter　　ビデオ入力の PreFilter の値が格納されます。

値	意味
CDE_FILTER_MIN_RANGE	最小値
CDE_FILTER_MAX_RANGE	最大値

nVideoProcessing　　ビデオプロセッシング の値が格納されます。

値	意味
CDE_VIDEO_PROCESSING_NONE	ビデオプロセッシングを使用しません。
CDE_VIDEO_PROCESSING_3DYC	3 次元 YC 分離を使用します。 コンポジット入力時のみ有効です。
CDE_VIDEO_PROCESSING_NR_W	ノイズリダクション弱を使用します。
CDE_VIDEO_PROCESSING_NR_S	ノイズリダクション強を使用します。

戻り値

成功　　CDE_SUCCESS
失敗　　CDE_SUCCESS 以外の値

補足

- **CDE_VIDEOPARAM:: dwPreFilter は MVR-D4400 では有効でありません。**
- **CDE_VIDEOPARAM:: nVideoProcessing は MVR-D4000 では有効でありません。**

CDE_SetVideoSetting

ビデオの設定を行います。

書式

CDEAPI CDE_SetVideoSetting(CDEHANDLE hHandle, CDE_VIDEOPARAM* pParam);

引数

hHandle	ボードを識別するためのハンドル
pParam	設定するビデオの情報を格納する **CDE_VIDEOPARAM** 構造体

戻り値

成功	**CDE_SUCCESS**
失敗	**CDE_SUCCESS** 以外の値

参照

CDE_GetVideoSetting

CDE_GetAudioSetting

オーディオの設定を取得します。

書式

CDEAPI CDE_GetAudioSetting(CDEHANDLE hHandle, CDE_AUDIOPARAM* pParam);

引数

hHandle ボードを識別するためのハンドル
pParam オーディオの設定の情報を格納する **CDE_AUDIOPARAM** 構造体

CDE_AUDIOPARAM 構造体の定義

オーディオの設定。

```
typedef struct {
        DWORD dwSize;
        INT nStreamType;
        INT nAudioLayer;
        INT nErrorProtection;
        DWORD dwBitRate;
        DWORD dwAudioSamplingRate;
        INT nAudioMode;
} CDE_AUDIOPARAM;
```

dwSize この構造体のサイズ。sizeof(CDE_ AUDIOPARAM) を指定。
nStreamType Audio Stream のタイプが格納されます。
CDE_MPEG1_AUDIO を指定します。

値	意味
CDE_MPEG1_AUDIO	Mpeg1 Audio

nAudioLayer Audio Layer の情報が格納されます。
CDE_AUDIO_LAYER2 を指定します。

値	意味
CDE_AUDIO_LAYER2	Layer2

nErrorProtection 予約 : 0 を設定してください
dwBitRate オーディオビットレートが格納されます。設定できる値は次のとおりです。(256000, 320000, 384000 , default : 384000)
dwAudioSamplingRate オーディオのサンプリングレートが格納されます。設定できる値は次の通りです。(32000, 44100, 48000,, default : 48000)
nAudioMode オーディオのモードが格納されます。

(default : CDE_AUDIO_STEREO)

値	意味
CDE_AUDIO_STEREO	ステレオ
CDE_AUDIO_JOINT_STEREO	ジョイント・ステレオ
CDE_AUDIO_DUAL_CHANNEL	デュアル・チャネル
CDE_AUDIO_SINGLE_CHANNEL	モノラル

戻り値

成功　　**CDE_SUCCESS**
失敗　　**CDE_SUCCESS 以外の値**

CDE_SetAudioSetting

オーディオの設定を行います。

書式

CDEAPI CDE_SetAudioSetting(CDEHANDLE hHandle, CDE_AUDIOPARAM* pParam);

引数

hHandle	ボードを識別するためのハンドル
pParam	設定するオーディオの情報を格納する **CDE_ AUDIOPARAM** 構造体

戻り値

成功	**CDE_SUCCESS**
失敗	**CDE_SUCCESS** 以外の値

参照

CDE_GetAudioSetting

CDE_GetAudioEncVol

エンコード時の音声レベルを取得します。

書式

CDEAPI CDE_GetAudioEncVol(CDEHANDLE hHandle, INT* pnVal);

引数

hHandle	ボードを識別するためのハンドル
pnVal	オーディオのエンコードボリュームを格納するための **INT** 変数へのポインタ

戻り値

成功	**CDE_SUCCESS**
失敗	**CDE_SUCCESS** 以外の値

CDE_SetAudioEncVol

エンコード時の音声レベルを設定します。

書式

CDEAPI CDE_SetAudioEncVol(CDEHANDLE hHandle, INT nVal);

引数

hHandle	ボードを識別するためのハンドル
nVal	設定するオーディオのエンコードボリューム **範囲 : 48 ～ -128 (Default : 5)**

戻り値

成功	**CDE_SUCCESS**
失敗	**CDE_SUCCESS 以外の値**

CDE_GetTsEncInfParam

トランスポートストリーム用の追加設定を取得します。

書式

CDEAPI CDE_GetTsEncInfParam(CDEHANDLE hHandle, CDE_TSENCINFPARAM* pParam);

引数

hHandle	ボードを識別するためのハンドル
pParam	トランスポートストリーム用の追加設定の情報を格納する **CDE_TSENCINFPARAM** 構造体

CDE_TSENCINFPARAM 構造体の定義

トランスポートストリーム追加設定情報

```
typedef struct {
        DWORD dwSize;
        INT nPatRate;
        DWORD dwPmtPid;
        INT nPmtRate;
        DWORD dwPcrPid;
        INT nPcrRate;
        DWORD dwVideoPid;
        DWORD dwAudioPid;
} CDE_TSENCINFPARAM;
```

dwSize	この構造体のサイズ。Sizeof(CDE_TSENCINFPARAM) を指定。
nPatRate	プログラム・アソシエーション・テーブル の出現する頻度。 単位 mS （default : 100 範囲 20 ～ 500 ）
dwPmtPid	プログラム・マップ・テーブルの id （default : 0x20 範囲 21 ～ 8190）
nPmtRate	プログラム・マップ・テーブルの出現する頻度。 単位 mS (default : 400 範囲 20 ～ 500)
dwPcrPid	PCR の id（deault : 0x25 範囲 21 ～ 8190）
nPcrRate	予約（30） PCR の出現する頻度。単位 Hz（default : 30 範囲 20 ～ 40）
dwVideoPid	ビデオの id （default : 0x21 範囲 21 ～ 8190）
dwAudioPid	オーディオの id（default : 0x22 範囲 21 ～ 8190）

戻り値

成功	**CDE_SUCCESS**
失敗	**CDE_SUCCESS 以外の値**

CDE_SetTsEncInfParam

トランスポートストリーム用の追加設定を設定します。

書式

CDEAPI CDE_SetTsEncInfParam(CDEHANDLE hHandle, CDE_TSENCINFPARAM* pParam);

引数

hHandle	ボードを識別するためのハンドル
pParam	トランスポートストリーム用の追加設定の情報を格納する **CDE_TSENCINFPARAM** 構造体

戻り値

成功	**CDE_SUCCESS**
失敗	**CDE_SUCCESS** 以外の値

参照

CDE_GetTsEncInfParam

CDE_GetEncodeParam2

拡張エンコードパラメータを取得します。

書式

CDEAPI CDE_GetEncodeParam2(CDEHANDLE hHandle, CDE_ENCODEPARAM2* pParam);

引数

hHandle	ボードを識別するためのハンドル
pParam	拡張エンコードパラメータの情報を格納する CDE_ENCODEPARAM2 構造体

CDE_ENCODEPARAM2 構造体の定義

拡張エンコードパラメータ設定情報

```
typedef struct {
    DWORD dwSize;
    WORD wVertStartPosF1;
    WORD wVertStartPosF2;
    DWORD dwHorizStartPos;
} CDE_ENCODEPARAM2;
```

dwSize	この構造体のサイズ。sizeof(CDE_ENCODEPARAM2) を指定。
wVertStartPosF1	フィールド1のキャプチャしないライン数を指定します。-1 を指定すると NTSC : 21、PAL : 25 が設定されます。(default: -1)
wVertStartPosF2	フィールド2のキャプチャしないライン数を指定します。-1 を指定すると NTSC : 21、PAL : 24 が設定されます。(default: -1)
dwHorizStartPos	水平スタートポジション。(default : 0)

戻り値

成功	CDE_SUCCESS
失敗	CDE_SUCCESS 以外の値

補足

- この API は CDE_PrepareDevice で CDE_PS_MODE を指定したときのみ有効です。
- Ver 1.31 では MVR-D4400 はサポートしていません。

参照

CDE_SetEncodeParam2

CDE_SetEncodeParam2

拡張エンコードパラメータを設定します。

書式

CDEAPI CDE_SetEncodeParam2(CDEHANDLE hHandle, CDE_ENCODEPARAM2* pParam);

引数

hHandle	ボードを識別するためのハンドル
pParam	拡張エンコードパラメータの情報を格納する **CDE_ENCODEPARAM2** 構造体

戻り値

成功	**CDE_SUCCESS**
失敗	**CDE_SUCCESS** 以外の値

補足

- この **API** は **CDE_PrepareDevice** で **CDE_PS_MODE** を指定したときのみ有効です。
- **Ver 1.31** では **MVR-D4400** はサポートしていません。

参照

CDE_GetEncodeParam2

CDE_GetMDSetting

動き検出の設定を取得します。

書式

CDEAPI CDE_GetMDSetting (CDEHANDLE hHandle,
CDE_MDPARAMARRAY* pParam);

引数

hHandle	ボードを識別するためのハンドル
pParam	動き検出の設定情報を格納する CDE_MDPARAMARRAY 構造体

CDE_MDPARAM 構造体の定義

動き検出の範囲の設定情報

```
typedef struct {
    BOOL fEnable;
    INT nLeft;
    INT nTop;
    INT nRight;
    INT nBottom;
    DWORD dwSensitivity;
} CDE_MDPARAM;
```

fEnable	設定が有効か無効かを指定します。
nLeft	設定範囲の左上隅 x 座標を指定します。 範囲（0 ～ 入力サイズ(幅) / 16)
nTop	設定範囲の左上隅 y 座標を指定します。 範囲（0 ～ 入力サイズ(高さ) / 16)
nRight	設定範囲の右下隅 x 座標を指定します。 範囲（0 ～ 入力サイズ(幅) / 16)
nBottom	設定範囲の右下隅 y 座標を指定します。 範囲（0 ～ 入力サイズ(高さ) / 16)
dwSensitivity	感度を指定します。範囲（最大 0 ～ 最小 0xFFFF)

CDE_MDPARAMARRAY 構造体の定義

動き検出の設定情報

```
typedef struct {
        LPVOID pParam;
        DWORD dwInterval;
        CDE_MD_CALLBACK MdCallback;
        CDE_MDPARAM_MdParam[MAX_MD_ENABLE];
} CDE_MDPARAMARRAY;
```

pParam	**ここで設定した値がコールバック関数の一つ目の引数にセットされます。**
dwInterval	**予約。0を設定してください。**
MdCallback	**設定範囲に動きがあった場合呼ばれるコールバック関数を指定します。**
MdParam[MAX_MD_ENABLE]	**動き検出の設定。最大 MAX_MD_ENABLE（9）までのポイントを設定できます。**

CDE_MD_CALLBACK コールバック関数の定義

動き検出の情報を取得するためのコールバック関数

```
VOID CALLBACK MDCallBack(VOID* pParam, LPDWORD lpdwMDStatus);
```

pParam	**CDE_MDCALLBACK::pParam に設定した値**
lpdwMDStatus	**動き検出の情報。** **CDE_MDPARAMARRAY::MdParam[MAX_MD_ENABLE]** に対応して情報が設定されます。 例＞ **CDE_MDPARAMARRAY::MdParam[0]** に対応する情報は、LpdwMDStatus[0]。 **CDE_MDPARAMARRAY::MdParam[1]** に対応する情報は、LpdwMDStatus[1]。 ・・・ **CDE_MDPARAMARRAY::MdParam[MAX_MD_ENABLE]** に対応する情報は、LpdwMDStatus[MAX_MD_ENABLE]。

戻り値

成功	**CDE_SUCCESS**
失敗	**CDE_SUCCESS 以外の値**

補足

- この **API** は **CDE_PrepareDevice** で **CDE_PS_MODE** を指定したときに有効です。
- 動き検出はエンコードを開始しているときに有効です。
- **MVR-D4400** ではこの機能は使用できません。
- **Visual Basic 6.0** では使用できません。

参照

CDE_SetMDSetting

CDE_SetMDSetting

動き検出の設定を行います。

書式

CDEAPI CDE_SetMDSetting (CDEHANDLE hHandle,
CDE_MDPARAMARRAY* pParam);

引数

hHandle	ボードを識別するためのハンドル
pParam	動き検出の設定情報を格納する **CDE_MDPARAMARRAY** 構造体

戻り値

成功	**CDE_SUCCESS**
失敗	**CDE_SUCCESS** 以外の値

補足

- この **API** は **CDE_PrepareDevice** で **CDE_PS_MODE** を指定したときに有効です。
- 動き検出はエンコードを開始しているときに有効です。
- **MVR-D4400** ではこの機能は使用できません。
- **Visual Basic 6.0** では使用できません。

参照

CDE_GetMDSetting

CDE_GetDecodeBssParam

メモリ転送デコード用のパラメータを取得します。

書式

CDEAPI CDE_GetDecodeBssParam (CDEHANDLE hHandle, CDE_BSSPARAM* pParam);

引数

hHandle	ボードを識別するためのハンドル
pParam	メモリ転送デコード用パラメータの情報が格納された **CDE_BSSPARAM** 構造体

戻り値

成功	**CDE_SUCCESS**
失敗	**CDE_SUCCESS** 以外の値

参照

CDE_GetEncodeBssParam

CDE_SetDecodeBssParam

メモリ転送デコード用のパラメータを設定します。

書式

CDEAPI CDE_SetDecodeBssParam(CDEHANDLE hHandle, CDE_BSSPARAM* pParam);

引数

hHandle	ボードを識別するためのハンドル
pParam	メモリ転送デコード用パラメータの情報を格納した **CDE_BSSPARAM** 構造体

戻り値

成功	**CDE_SUCCESS**
失敗	**CDE_SUCCESS** 以外の値

補足

- **MVR-D4000** では指定できるバッファのサイズは **0x1000** 単位でしか指定できません。それ以外のサイズを指定した場合は **0x1000** 単位に切り下げられます。
- **MVR-D4400** ではドライバが設定可能な値まで切り下げられます。

参照

CDE_GetEncodeBssParam

CDE_GetDecodeSetting

デコードのパラメータを取得します。

書式

CDEAPI CDE_GetDecodeSetting(CDEHANDLE hHandle,
CDE_DECODEPARAM* pParam);

引数

hHandle ボードを識別するためのハンドル
pParam デコードのパラメータを格納した **CDE_DECODEPARAM** 構造体

CDE_DECODEPARAM 構造体の定義

デコードの設定。

```
typedef struct {
    DWORD dwSize;
    INT nMpegStandard;
    INT nTVSystem;
}CDE_DECODEPARAM;
```

dwSize **この構造体のサイズ。sizeof(CDE_DECODEPARAM)** を指定。
nMpegStandard **MPEG の種類が格納されます。**
(default : CDE_AUTO_DETECT)

値	意味
CDE_AUTO_DETECT	自動的に検知します。
CDE_MPEG1	**MPEG1**
CDE_MPEG2	**MPEG2**
CDE_MPEG4	**MPEG4**

nTVSystem **放送規格の種類が格納されます。nMpegStandard** に
CDE_AUTO_DETECT が設定されている場合、
自動的に検知します。**(default : CDE_TVSYSTEM_NTSC)**

値	意味
CDE_TVSYSTEM_NTSC	**NTSC**
CDE_TVSYSTEM_PAL	**PAL**

戻り値

成功 **CDE_SUCCESS**
失敗 **CDE_SUCCESS** 以外の値

CDE_SetDecodeSetting

デコードのパラメータを設定します。

書式

CDEAPI CDE_SetDecodeSetting(CDEHANDLE hHandle,
CDE_DECODEPARAM* pParam);

引数

hHandle	ボードを識別するためのハンドル
pParam	デコードのパラメータを格納した **CDE_DECODEPARAM** 構造体

戻り値

成功	**CDE_SUCCESS**
失敗	**CDE_SUCCESS** 以外の値

参照

CDE_GetDecodeSetting

CDE_GetPciBusInf

PCI バスの情報を取得します。

書式

CDEAPI CDE_GetPciBusInf(CDEHANDLE hHandle, CDE_PCIBUSINF* pBusInf);

引数

hHandle	ボードを識別するためのハンドル
pBusInf	**PCI バスの情報を格納する CDE_PCIBUSINF 構造体**

CDE_PCIBUSINF 構造体の定義

PCI バスの情報

```
typedef struct {
        WORD wBus;
        WORD wDevice;
        WORD wFunction;
} CDE_TSENCINFPARAM;
```

wBus;	**PCI バス番号**
wDevice	**デバイス番号**
wFunction	**機能 情報**

戻り値

成功	**CDE_SUCCESS**
失敗	**CDE_SUCCESS 以外の値**

CDE_GetIreVal

NTSC セットアップレベル（黒レベル）を取得します。

書式

CDEAPI CDE_GetIreVal(CDEHANDLE hHandle, LPDWORD pdwVal);

引数

hHandle	ボードを識別するためのハンドル
pdwVal	セットアップレベルの情報を格納するための **DWORD** 変数へのポインタ。

戻り値

成功	**CDE_SUCCESS**
失敗	**CDE_SUCCESS** 以外の値

CDE_SetIreVal

NTSC セットアップレベル（黒レベル）を設定します。

書式

CDEAPI CDE_SetIreVal(CDEHANDLE hHandle, DWORD dwVal);

引数

hHandle	ボードを識別するためのハンドル
dwVal	ビデオ入力部の設定を **7.5 IRE** に設定する場合は **CDE_VIDEO_DECODE_75IRE が表すビットをセットします。初期値：0IRE** **ビデオ出力部の設定を 7.5 I RE に設定する場合は** **CDE_VIDEO_ENCODE_75IRE が表すビットをセットします。初期値:0IRE**

戻り値

成功	**CDE_SUCCESS**
失敗	**CDE_SUCCESS** 以外の値

補足

- 日本では **0IRE** が標準です。
- ビデオ入力部のセットアップレベル設定を変更した場合、**CDE_SetVideoSetting** で設定されるビデオ入力パラメータは対応するデフォルトに変更されます。

APPENDIX

1. エラーコード一覧

エラーコードは次の通りです。

● 関数戻り値

値	意味
CDE_SUCCESS	正常に終了
CDE_FAILURE	失敗
CDE_FAIL_BADPARAM	不正なパラメータです。
CDE_FAIL_BADVERSION	**DLL** のバージョンが不正です。
CDE_FAIL_NO_INPUT_SOURCE	映像ソースの入力がありません。
CDE_FAIL_INVALID_HANDLE	不正なハンドルです。
CDE_FAIL_ILLEGAL_FORMAT	不正なファイルフォーマットです。
CDE_FAIL_INVALID_MODE	不正なストリームのモードです。
CDE_FAIL_UNSUPPORTED	サポートしていない処理です。
CDE_FAIL_ALLOC_FAILURE	リソースの割り当てに失敗しました。
CDE_FAIL_INITIALIZING	正しく初期化できていません。
CDE_FAIL_CHIP_CTRL	デバイスの設定に失敗しました
CDE_FAIL_BAD_STATUS	不正なステータスです。

● コールバック

値	意味
CDE_ER_NOERR	エラーのコールバック関数の設定に成功した時（初期化時に一度のみ）に発生します。
CDE_ER_CHIP_CTRL	なんらかのデバイスのコントロールに失敗しました。
CDE_ER_SYSTEM_ERR	メモリの確保に失敗したなど継続が困難なエラーが発生しました。
CDE_ER_ENC_CHIP_CTRL	エンコードデバイスのコントロールに失敗しました。
CDE_ER_ENC_READ_TOUT	エンコードデータのリード時にタイムアウトが発生しました。
CDE_ER_ENC_NO_FREE_BUF	エンコード用に用意したバッファがオーバーフローしました。
CDE_ER_ENC_WRITE_FILE_FAIL	エンコードデータをファイルに保存するのに失敗しました。
CDE_ER_DEC_CHIP_CTRL	デコードデバイスのコントロールに失敗しました。

CDE_ER_DEC_WRITE_TOUT	デコードのデータリード時にタイムアウトが発生しました。
CDE_ER_DEC_NO_BUF_SUPPLY	データの供給が一定時間ありませんでした。
CDE_ER_DEC_REBUF_FAIL	データの供給が一定時間なかったため、再バッファリングを試みたが失敗しました。
CDE_ER_DEC_READ_FILE_FAIL	デコードデータをファイルから読み出すのに失敗しました。
CDE_ER_COMM_FAIL	ファームからのリターンコード操作が失敗しました。
CDE_ER_COMM_FAIL	ファームからのリターンコード操作が失敗しました。
CDE_ER_COMM_INVALID_HANDLE	ファームからのリターンコード不正なハンドルです。
CDE_ER_COMM_DMA_FAIL	ファームからのリターンコード **DMA** の転送が失敗しました。
CDE_ER_COMM_INVALID_STATE	ファームからのリターンコード不正なステータスです。
CDE_ER_COMM_INVALID_COMMAND	ファームからのリターンコード不正なコマンドです。
CDE_ER_COMM_OUT_OF_RESOURCE	ファームからのリターンコードメモリ範囲外です。

2. ステータスコード一覧

ステータスコードは次の通りです。

それぞれのビットが対応しているため、複数同時にセットされることもあります。

値	意味
CDE_STATUS_MASK	全てのステータス範囲のマスクです。
CDE_ENC_STATUS_MASK	エンコードのステータスの範囲のマスクです。
CDE_ENC_STOP	エンコード停止中
CDE_ENC_START	エンコード開始中
CDE_ENC_PAUSE	エンコード一時停止
CDE_ENC_MACV_DETECT	マクロビジョンを検知しました。
CDE_ENC_GOP_RESET_DETECT	ストリームがリセットされました。
CDE_DEC_STATUS_MASK	デコードのステータスの範囲のマスクです。
CDE_DEC_STOP	デコード停止中
CDE_DEC_START	デコード開始中
CDE_DEC_PAUSE	デコード一時停止中

カノープス株式会社
本社／〒651-2241 神戸市西区室谷 1-2-2（神戸ハイテクパーク内）

● MVR-D4000 Series Development Kit の
技術的なお問い合わせは〈カノープスシステム開発サポート〉へ

インターネット E-mail :mvrsdk@canopus.co.jp
FAX :078-992-4203

※FAX でお問い合わせの際は、FAX 番号をよくお確かめください。

☆最新の情報をホームページでご覧になれます。

http://www.canopus.co.jp/