DLL 関数一覧

PCI-3120, PCI-3126, PCI-3133, PCI-3135, PCI-3163, PCI-3165, PCI-3166, PCI-3168C, PCI-3170A, PCI-3171A, PCI-3172A, PCI-3173A, PCI-3177C, PCI-3178, PCI-3180, PCI-3521, PCI-3522A, PCI-3523A

No	関 数 名	機 能
1	AdOpen	アナログ入力ボードのオープンを行い、以後ボードへのアクセスを行えるようにします。
2	AdClose	アナログ入力ボードのクローズを行い、ボードアクセスのために使用されていた各種リソースの解放を行い、以後ボードへのアクセスを禁止します。
3	AdGetDeviceInfo	アナログ入力ボードの仕様を取得します。
4	AdSetBoardConfig	アナログ入力ボードのイベント設定を行います。
5	AdGetBoardConfig	アナログ入力ボードのイベント要因を取得します。
6	AdSetSamplingConfig	アナログ入力ボードのサンプリング条件の設定を行います。
7	AdGetSamplingConfig	アナログ入力ボードの現在設定されているサンプリング条件を取得します。
8	AdAllocateSamplingBuffer	アナログ入力ボードのサンプリングバッファを確保します。
9	AdGetSamplingData	アナログ入力ボードから入力されたサンプリングデータを取得します。
10	AdClearSamplingData	サンプリング入力バッファ内のサンプリングデータをクリアします。
11	AdReadSamplingBuffer	サンプリングバッファに格納されているサンプリングデータを取得します。
12	AdStartSampling	アナログ入力ボードの連続サンプリングをスタートさせます。
13	AdStartFileSampling	アナログ入力ボードからのサンプリングデータをデータファイルに書き込みながらサンプリングを行います。
14	AdTriggerSampling	トリガ（EXTRG IN）が入力されるたびに、アナログ入力ボードから 1 件のアナログ入力を行います。
15	AdMemTriggerSampling	外部トリガが入力されるたびに、アナログ入力ボードから指定件数のサンプリングを行います。
16	AdStopSampling	アナログ入力ボードのサンプリングを停止させます。
17	AdGetStatus	アナログ入力ボードのサンプリング動作状態を取得します。
18	AdInputAD	アナログ入力ボードから 1 件のアナログ入力を行います。
19	AdInputDI	アナログ入力ボードの汎用デジタル入力端子を読み出します。
20	AdOutputDO	アナログ入力ボードの汎用デジタル出力端子へデータを出力します。
21	AdAdjustVR	アナログ入力ボードの電子ボリューム制御を行います。
22	AdDataConv	アナログデータの形式を変換します。形式の変換とともにデータに対し平均処理やスムージング処理を行うことができます。
23	AdReadFile	ファイルに保存されているサンプリングデータを、指定のバッファに読み込む。
24	fnConv	AdDataConv関数で使用するコールバック関数のプレースホルダです。データ変換時にfnConv関数を呼び出すことができます。

		fnConv関数は、データ 1 点毎に呼び出されます。
25	CallBackProc	非同期サンプリング終了時に呼び出されるコールバック関数のプレースホルダです。非同期サンプリング終了時にCallBackProc関数を呼び出すことができます。
26	AdCommonGetPciDeviceInfo	アナログ入力ボードのリソース情報を取得します。

AdOpen

【機　能】

アナログ入力ボードのオープンを行い、以後のボードへのアクセスを行えるようにします。
Windows XP/2000/Me/98/95 では、「デバイス マネージャ」に「FbiPciAd」が追加され認識された弊社アナログ入力ボードが一覧表示されます。一覧の製品型式の横にボード上のロータリスイッチの値とデバイス名が表示されます。AD診断プログラム（DiagAd.exe）、AD調整プログラム（AdAdjust.exe）、AD波形入力プログラム（AdWaveSmp.exe）でも確認できます。
Windows NT 4.0 では、AD診断プログラム（DiagAd.exe）、AD調整プログラム（AdAdjust.exe）またはAD波形入力プログラム(AdWaveSmp.exe)にて割り当てられたデバイス名をご確認ください。
デバイス名は使用するボード枚数やスロット位置の変更等でシステム構成が変化すると割り当てられる名前が変化する場合があります。システム構成が変化する環境でご使用になる場合は、デバイス名の指定が変更できるようにアプリケーションを作成ください。

【書　式】

●C 言語

```
HANDLE AdOpen(
  LPCTSTR lpszName // デバイス名
);
```

【パラメータ】

lpszName	オープンするデバイス名を指定します。 デバイス名は FBIAD1～FBIAD255。

【戻り値】

AdOpen 関数が正常に終了した場合には、有効なハンドルが返されます。
オープンに失敗した場合には、INVALID_HANDLE_VALUE(FFFFFFFFh)が返されます。

【使用例】

●C 言語

```
HANDLE hDeviceHandle;
hDeviceHandle = AdOpen("FBIAD1");
```

デバイス名"FBIAD1"のアナログ入力ボードをオープンし、変数 hDeviceHandle にデバイスハンドルを返します。

AdClose

【機　能】

アナログ入力ボードのクローズを行い、ボードアクセスのために使用されていた各種リソースの解放を行い、以後のボードへのアクセスを禁止します。

【書　式】

●C 言語

```
INT AdClose(
  HANDLE hDeviceHandle
);
```

【パラメータ】

hDeviceHandle　AdOpen関数で取得したデバイスハンドルを指定してください。

【戻り値】

AdClose関数は処理が正常終了するとAD_ERROR_SUCCESSを返します。正常に処理が終了しなかった場合AD_ERROR_SUCCESS以外の値を返します。AD_ERROR_SUCCESS以外の値が返された場合については、戻り値一覧を参照してください。

【備　考】

再度、ボードへのアクセスを行う場合にはオープン処理(AdOpen関数)を呼び出してください。
サンプリング中に本関数を実行した場合、サンプリングを終了させます。

【使用例】

●C 言語

```
nRet = AdClose( hDeviceHandle );
```

デバイスハンドル hDeviceHandle のアナログ入力ボードのクローズ処理を行います。

AdGetDeviceInfo

【機　能】

アナログ入力ボードからボードの各種仕様の取得を行います。

【書　式】

●C 言語

```
INT AdGetDeviceInfo(
  HANDLE hDeviceHandle,
  PADBOARDSPEC pAdBoardSpec
);
```

【パラメータ】

hDeviceHandle　AdOpen関数で取得したデバイスハンドルを指定してください。
pAdBoardSpec　ADボードの仕様を格納する構造体(ADBOARDSPEC構造体)へのポインタを指定します。

【戻り値】

AdGetDeviceInfo関数は正常に終了するとAD_ERROR_SUCCESSを返します。それ以外の場合はAD_ERROR_SUCCESS以外の値を返します。AD_ERROR_SUCCESS以外の値が返された場合については、戻り値一覧を参照してください。

【使用例】

●C 言語

```
ADBOARDSPEC AdBoardSpec;
nRet = AdGetDeviceInfo( hDeviceHandle, &AdBoardSpec );
```

ADBOARDSPEC 構造体

ボードの仕様を格納する構造体です。AdGetDeviceInfo関数で使用されます。

● C 言語

```
typedef struct {
    ULONG     ulBoardType;
    ULONG     ulBoardID;
    DWORD     dwSamplingMode;
    ULONG     ulChCountS;
    ULONG     ulChCountD;
    ULONG     ulResolution;
    DWORD     dwRange;
    ULONG     ulIsolation;
    ULONG     ulDi;
    ULONG     ulDo;
} ADBOARDSPEC, *PADBOARDSPEC;
```

ulResolution　ボードの分解能を格納します。
・12bit AD ボードの場合：12 が格納されます。

AdInputAD

【機　能】

アナログ入力ボードから 1 件のアナログ入力（1 件サンプリング）を行います。
AdStartSampling関数を使用した通常の連続サンプリング入力とは異なり、ボードのアナログ入力機能のみを利用します。

【書　式】

●C 言語

```
INT AdInputAD(
    HANDLE hDeviceHandle,
    ULONG ulCh,
    ULONG ulSingleDiff,
    PADSMPLCHREQ lpAdSmplChReq,
    LPVOID lpData
);
```

【パラメータ】

hDeviceHandle	AdOpen関数で取得したデバイスハンドルを指定してください。
ulCh	アナログ入力を行うチャンネル数を指定します。 設定可能範囲は 1～そのボードのチャンネル数です。 必ず 1 以上の値を指定してください。
ulSingleDiff	シングルエンド入力／差動入力を指定します。 AD_INPUT_SINGLE　：シングルエンド入力 AD_INPUT_DIFF　：差動入力
lpAdSmplChReq	アナログ入力を行うチャンネル番号、レンジを指定するための構造体配列（ADSMPLCHREQ構造体）へのポインタを指定します。
lpData	アナログ入力したデータを格納する位置へのポインタを指定します。 lpData が指す位置にアナログ入力データを格納します。

【戻り値】

AdInputAD関数は正常に終了するとAD_ERROR_SUCCESSを返します。それ以外の場合はAD_ERROR_SUCCESS以外の値を返します。AD_ERROR_SUCCESS以外の値が返された場合については、戻り値一覧を参照してください。

【備　考】

- *ulCh* パラメータは入力を行うチャンネル数を指定します。例えば、入力チャンネル番号が 1,3,5,7 であればチャンネル数は 4 となります。
- 入力を行うチャンネル番号をADSMPLCHREQ構造体(サンプリングチャンネル構造体)のulChNoメンバに格納します。
- 連続サンプリング中に本関数を実行する事はできません。
- バスマスタ方式のボードで本関数を使用する場合には、AD 変換のタイミングが AdSetSamplingConfig 関数にて設定しているサンプリング周波数に依存するため、サンプリング周波数で設定している時間だけ本関数の時間が長くなってしまいます。
- 1 データあたりのサイズはボードの分解能により異なります。　（データ形式参照）
  各言語で使用する変数の型は次のようになります。

分解能	サイズ (バイト)	C 言語	Visual Basic	Delphi
8bit	1	BYTE	Byte	Byte
12bit	2	WORD	Integer	Word
16bit	2	WORD	Integer	Word
24bit	4	DWORD	Long	Dword

【使用例】

●C 言語

```
WORD pData[4];
ADSMPLCHREQ SmplChReq[4];

SmplChReq[0].ulChNo = 1;
SmplChReq[0].ulRange = AD_5V;
SmplChReq[1].ulChNo = 3;
SmplChReq[1].ulRange = AD_5V;
SmplChReq[2].ulChNo = 5;
SmplChReq[2].ulRange = AD_5V;
SmplChReq[3].ulChNo = 7;
SmplChReq[3].ulRange = AD_5V;

nRet = AdInputAD( hDeviceHandle, 4, AD_INPUT_SINGLE, &SmplChReq[0],
pData );
```

ADSMPLCHREQ 構造体

各チャンネル毎のサンプリング条件を設定する構造体です。
ADSMPLREQ構造体、ADBMSMPLREQ構造体のメンバ、AdInputAD関数で使用されます。

● C 言語

```
typedef struct {
   ULONG    ulChNo;
   ULONG    ulRange;
} ADSMPLCHREQ, *PADSMPLCHREQ;
```

メンバ	説　明
ulChNo	サンプリングを行うチャンネルの番号を指定します。 指定の範囲は以下の通りです。 ・ 1 ～ そのボードが提供する最大チャンネル番号

ulRange　　ulChNo で指定したチャンネルのレンジを指定します。
マルチ ADC 方式のボードではチャンネル毎にレンジの設定が可能です。
マルチプレクサ方式のボードでは全チャンネル同一のレンジ設定にする必要があります。
（CTP-3183 については、チャンネル毎にレンジの設定が可能です。）
レンジの指定は、レンジ識別子のうちいずれか 1 つを指定します。

AD_0_1V	：電圧	ユニポーラ	0～1V
AD_0_2P5V	：電圧	ユニポーラ	0～2.5V
AD_0_5V	：電圧	ユニポーラ	0～5V
AD_0_10V	：電圧	ユニポーラ	0～10V
AD_1_5V	：電圧	ユニポーラ	1～5V
AD_0_2V	：電圧	ユニポーラ	0～2V
AD_0_0P125V	：電圧	ユニポーラ	0～0.125V
AD_0_0P156V	：電圧	ユニポーラ	0～0.156V
AD_0_1P25V	：電圧	ユニポーラ	0～1.25V
AD_0_0P625V	：電圧	ユニポーラ	0～0.625V
AD_0_20mA	：電流	ユニポーラ	0～20mA
AD_4_20mA	：電流	ユニポーラ	4～20mA
AD_1V	：電圧	バイポーラ	±1V
AD_2P5V	：電圧	バイポーラ	±2.5V
AD_5V	：電圧	バイポーラ	±5V
AD_10V	：電圧	バイポーラ	±10V
AD_20V	：電圧	バイポーラ	±20V
AD_50V	：電圧	バイポーラ	±50V
AD_0P125V	：電圧	バイポーラ	±0.125V
AD_0P156V	：電圧	バイポーラ	±0.156V
AD_1P25V	：電圧	バイポーラ	±1.25V
AD_0P625V	：電圧	バイポーラ	±0.625V
AD_0P156V_AC	：電圧	バイポーラ	±0.156V(AC カップリング機能搭載)
AD_1P25V_AC	：電圧	バイポーラ	±1.25V(AC カップリング機能搭載)
AD_0P625V_AC	：電圧	バイポーラ	±0.625V(AC カップリング機能搭載)
AD_GND	：内部 GND 接続		

AdInputDI

【機　能】

アナログ入力ボードの汎用入力端子を読み出します。

【書　式】

●C 言語

```
INT AdInputDI(
    HANDLE hDeviceHandle,
    DWORD *dwData
);
```

【パラメータ】

hDeviceHandle　　AdOpen関数で取得したデバイスハンドルを指定してください。
dwData　　入力したデジタルデータを返す位置を指すポインタを指定します。
(データ形式参照)

【戻り値】

AdInputDI関数は正常に終了するとAD_ERROR_SUCCESSを返します。それ以外の場合はAD_ERROR_SUCCESS以外の値を返します。AD_ERROR_SUCCESS以外の値が返された場合については、戻り値一覧を参照してください。

【備　考】

本関数は、汎用入力端子を搭載していないボード (PCI-3161, 3163) では使用できません。

【使用例】

●C 言語

```
DWORD dwData;
nRet = AdInputDI( hDeviceHandle, &dwData );
```

8.3 デジタル入力データ

デジタル入力データは、そのボードが持っている汎用デジタル入力端子の状態を bit 単位で表したものです。ボードにより汎用デジタル入力端子の数および極性が異なります。USER'S MANUAL を参照してください。

デジタルデータの形式

	bit31	～	bit16	bit15	～	bit0
DWORD 型(32bit)	未使用			IN16 の状態	・・・・・	IN1 の状態

DLL 関数一覧

PCI-3325	PCI-3341A	PCI-3342A	PCI-3343A	PCI-3345A	PCI-3346A	PCI-3521
PCI-3522A	PCI-3523A	CTP-3325	CTP-3342	CTP-3343	CTP-3346	CTP-3350
CTP-3351	CTP-3521	CTP-3522	CTP-3523			

No	関 数 名	機 能
1	DaOpen	アナログ出力ボードのオープンを行い、以後のボードへのアクセスを行えるようにします。
2	DaClose	アナログ出力ボードのクローズを行い、ボードアクセスのために使用されていた各種リソースの解放を行い、以後のボードへのアクセスを禁止します。
3	DaGetDeviceInfo	アナログ出力ボードの仕様を取得します。
4	DaSetBoardConfig	アナログ出力ボードのボード動作の設定を行います。
5	DaGetBoardConfig	アナログ出力ボードの現在のボード動作設定を取得します。
6	DaSetSamplingConfig	アナログ出力ボードのアナログ出力更新条件の設定を行います。
7	DaGetSamplingConfig	アナログ出力ボードの現在設定されているアナログ出力更新条件を取得します。

8	DaSetSamplingData	アナログ出力ボードから出力するアナログ出力データのセットを行います。
9	DaClearSamplingData	アナログ出力バッファ内のデータをクリアします。
10	DaStartSampling	アナログ出力ボードのアナログ出力更新をスタートさせます。
11	DaStartFileSampling	データファイルを読み込み、アナログ出力更新を行います。
12	DaSyncSampling	複数枚同期アナログ出力機能を使用したアナログ出力更新をスタートさせます。
13	DaStopSampling	アナログ出力ボードのアナログ出力更新を停止させます。
14	DaGetStatus	アナログ出力ボードのアナログ出力更新動作状態を取得します。
15	DaOutputDA	アナログ出力ボードから 1 件のアナログ出力を行います。
16	DaInputDI	アナログ出力ボードの汎用デジタル入力端子を読み出します。
17	DaOutputDO	アナログ出力ボードの汎用デジタル出力端子へデータを出力します。
18	DaAdjustVR	アナログ出力ボードの電子ボリュームの制御を行います。
19	DaDataConv	アナログデータの形式を変換します。形式の変換とともにデータに対し平均処理やスムージング処理を行うことができます。
20	DaWriteFile	指定バッファ内のアナログ出力データを、ファイルに保存します。
21	fnConv	DaDataConv関数で使用するコールバック関数のプレースホルダーです。データ変換時にfnConv関数を呼び出すことができます。fnConv関数は、データ 1 点毎に呼び出されます。
22	CallBackProc	アナログ出力更新終了時に呼び出されるコールバック関数のプレースホルダーです。アナログ出力更新終了時に CallBackProc 関数を呼び出すことができます。

DaOutputDA

【機　能】

アナログ出力ボードから 1 件のアナログ出力を行います。
DaStartSampling関数を使用した通常のアナログ出力更新とは異なり、ボードのアナログ出力機能のみを利用します。

【書　式】

●C 言語

```
INT DaOutputDA(
  HANDLE          hDeviceHandle,
  ULONG           ulCh,
  PDASMPLCHREQ    pDaSmplChReq,
  LPVOID          pData
);
```

【パラメータ】

hDeviceHandle	DaOpen関数で取得したデバイスハンドルを指定してください。
ulCh	アナログ出力を行うチャンネル数を指定します。 1～そのボードの提供する最大チャンネル数

	ulCh には必ず 1 以上の値を指定してください。
pDaSmplChReq	アナログ出力を行うチャンネル番号、レンジを指定するための構造体配列(DASMPLCHREQ構造体)へのポインタを指定します。
pData	アナログ出力するデータを格納してある位置へのポインタを指定します。 *pData*が指す位置に格納されているアナログデータを出力します。(データ形式参照)

【戻り値】

DaOutputDA関数は正常に終了するとDA_ERROR_SUCCESSを返します。それ以外の場合はDA_ERROR_SUCCESS以外の値を返します。DA_ERROR_SUCCESS以外の値が返された場合については、戻り値一覧を参照してください。

5.2 DLL 関数一覧

No	関 数 名	機　　能
1	PencOpen	エンコーダカウンタボードのオープンを行い、以後のボードへのアクセスを行えるようにします
2	PencClose	エンコーダカウンタボードのクローズを行い、ボードアクセスのために使用されていた各種リソースの解放を行い、以後のボードへのアクセスを禁止します
3	PencSetMode	エンコーダカウンタボードのモード切り替えを行います。動作モード、カウンタ方向、一致検出フラグの設定、ラッチ方法の設定を行います
4	PencGetMode	エンコーダカウンタボードの現在の設定状態を取得します
5	PencSetCounter	カウンタプリセット値を設定します
6	PencGetCounter	カウンタ値を取得します
7	PencSetCounterEx	複数のチャンネルにカウンタプリセット値を設定します
8	PencGetCounterEx	複数チャンネルのカウンタ値を取得します。
9	PencSetComparator	比較カウンタ値を設定します
10	PencGetComparator	比較カウンタ値を取得します
11	PencSetZMode	Z 相の極性、外部信号によるカウンタクリア・ラッチ条件を設定します
12	PencGetZMode	Z 相の極性・外部信号によるカウンタクリア・ラッチ条件を取得します
13	PencEnableCount	複数チャンネルのカウンタ動作を許可／禁止します
14	PencSetEventMask	イベントマスクを設定します
15	PencGetEventMask	イベントのマスク設定を取得します
16	PencSetEvent	コールバック関数を登録します (PCI-6204, CTP-6204 専用)
17	PencSetEventEx	コールバック関数を登録します
18	PencKillEvent	登録されたコールバック関数を削除します
19	PencEventRequestPending	エンコーダカウンタボードからのイベントを待ちます
20	PencSetTimerConfig	インターバルタイマの設定を行います
21	PencGetTimerConfig	インターバルタイマの設定情報を取得します
22	PencGetTimerCount	インターバルタイマのカウント値を取得します

23	PencGetStatus	カウンタのステータスを取得します
24	PencGetStatusEx	複数チャンネルのカウンタステータスとカウンタ値を取得します
25	PencReset	指定されたチャンネルのカウンタをリセットします

PencSetMode

【機　能】

エンコーダカウンタボードのモード切り替えを行います。動作モード、カウンタ方向、一致検出フラグの設定、ラッチ方法の設定を行います。

【書　式】

●C 言語

```
INT PencSetMode(
    HANDLE  hDeviceHandle,   // デバイスハンドル
    INT     nChannel,        // モードを切り替えるチャンネル
    INT     nMode,           // モード
    INT     nDirection,      // カウンタ方向
    INT     nEqual,          // 一致検出指定
    INT     nLatch           // ラッチ設定
);
```

【パラメータ】

hDeviceHandle　PencOpen関数で取得したデバイスハンドルを指定してください。

nChannel　モードを切り替えるチャンネルを指定します。
PCI-6204,CTP-6204 は 1～2
PCI-6205C,CTP-6205 は 1～8 を指定してください。

nMode　カウンタ動作モードを指定します。

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0	0	0	0	MD1	MD0	SEL1	SEL0

MD1	MD0	SEL1	SEL0	モード	カウント逓倍	クリアモード
0	0	0	0	ゲート付き単相パルスモード	標　準	非同期クリア
0	0	0	1		2 逓倍	
0	1	0	0	位相差パルスカウントモード	標　準	非同期クリア
0	1	0	1		2 逓倍	
0	1	1	0		4 逓倍	
1	1	0	0		標　準	同期クリア
1	1	0	1		2 逓倍	
1	1	1	0		4 逓倍	
1	0	0	0	アップ･ダウンパルスカウントモード	標　準	非同期クリア

nDirection　カウンタ方向を指定します。
0　カウンタ UP
1　カウンタ DOWN

nEqual　一致検出を行うかどうか指定します。
0　一致検出を行わない

1　一致検出を行う

nLatch　　ラッチ方法を指定します。
nLatch の設定により、**GetCounter 関数**、**GetCounterEx 関数**、**GetStatusEx 関数** 実行時に得られるカウンタの値は、次のように変化します。

0　ソフトウェアラッチ

読み出しレジスタの値を読み出します。関数を実行したときのカウンタの値が読み出せます。

1　外部ラッチ

読み出しレジスタの値を読み出します。カウンタの値は、外部ラッチ信号が入力されたときに読み出しレジスタに転送されていますので、最後に外部ラッチが入力されたときのカウンタの値が読み出せます。
外部ラッチ（PencSetZMode関数にて、外部ラッチ方法を設定してください）

【戻り値】

PencSetMode関数は正常に終了するとPENC_ERROR_SUCCESSを返します。それ以外の場合はPENC_ERROR_SUCCESS以外の値を返します。PENC_ERROR_SUCCESS以外の値が返された場合については、戻り値一覧を参照してください。

【使用例】

●C 言語

```
nRet = PencSetMode( hDeviceHandle, 1, 5, 0, 1, 0 );
```

デバイスハンドル hDeviceHandle のエンコーダカウンタボードのチャンネル 1 を位相差パルスカウントモード、2 逓倍、非同期クリア、カウンタ方向 UP、一致検出有効、ソフトウェアラッチに設定します。

PencGetCounter

【機　能】

カウンタ値を取得します。

【書　式】

●C 言語

```
INT PencGetCounter(
    HANDLE  hDeviceHandle,  // デバイスハンドル
    INT     nChannel,       // チャンネル
    PDWORD  pdwCounter      // カウンタ値格納用アドレスへのポインタ
);
```

【パラメータ】

hDeviceHandle　　PencOpen関数で取得したデバイスハンドルを指定してください。
nChannel　　カウンタ値を読み込むチャンネルを指定します。
PCI-6204,CTP-6204 は 1～2
PCI-6205C,CTP-6205 は 1～8 を指定してください。

pdwCounter	取得したカウンタ値を格納する変数へのポインタを指定します。 Visual Basic、Delphi では、変数の参照渡しとなります。

【戻り値】

PencGetCounter関数は正常に終了するとPENC_ERROR_SUCCESSを返します。それ以外の場合はPENC_ERROR_SUCCESS以外の値を返します。PENC_ERROR_SUCCESS以外の値が返された場合については、戻り値一覧を参照してください。

【使用例】

●C 言語

```
DWORD dwCounter;
nRet = PencGetCounter( hDeviceHandle, 1,&dwCounter );
```

デバイスハンドル hDeviceHandle のエンコーダカウンタボードのチャンネル 1 から dwCounter にカウンタ値を読み出します