API リファレンスマニュアル

Ver 1.03

履歴

バージョン	日付	内容
Ver 1.00	2010/10/11	新規作成
Ver 1.02	2011/12/21	バージョン合わせ
Ver 1.03	2015/08/21	誤記修正 API リファレンス関数追加 API リファレンス注意書き追加

目次

1. はじめに

PCI/PCIExpress Device Driver を操作するためのアプリケーションが使用する API I/F 仕様書です。アプリケーションがデバイスにアクセスするためにはこの仕様書にあるAPIを使用してアクセスして下さい。

ご注意

- このソフトウェアの著作権は株式会社スカイウェーブにあります。
- このマニュアルに記載されている事項は、予告なしに変更されることがあります。
- このソフトウェアおよびマニュアルはフリーで使用することができます。 （ユーザーカスタマイズした場合を除いて）
- このソフトウェアの仕様は予告なしに変更することがあります。
- このソフトウェアを使用して発生した問題については、使用者の責任において解決して下さい。
- このマニュアルの一部または全部を、株式会社スカイウェーブの文書による承諾なく、無断で複写、複製、転載、文書化することを禁じます。

株式会社スカイウェーブ
214-0014
神奈川県川崎市多摩区登戸２７０６－５白井ビル３Ｆ
TEL:044-931-1608 FAX:044-931-1609
E-MAIL:mr@skywave.co.jp
Home Page:http://www.skywave.co.jp/

2. API リファレンス

API リファレンス関数を使用するアプリケーションは include フォルダ配下にある、“pcielib.h“をインクルードする必要があります。また、pcielib フォルダ配下にある適切なインポートライブラリとリンクする必要があります。サンプルアプリケーションは Visual Studio2008 のプロジェクトファイルが含まれていますので、設定に関してはそちらを参照して下さい。

注：API リファレンス関数は Windows,Linux 共通で使用することが出来ます。
ただし次の関数は Linux では使用する事が出来ません

- ・Pcie_SetDeviceAndEventName
- ・Pcie_RegisterIntr
- ・Pcie_UnRegisterIntr
- ・Pcie_WritePciConfig
- ・Pcie_ReadPciConfig

2.1. Pcie_SetDeviceAndEventName

[説明]

デバイス名とイベント名を変更します。デバイス名とイベント名を変更する場合に呼び出します。(ユーザーズマニュアル、”デバイス名とイベント名の変更”を参照して下さい)本関数は Pcie_OpenDevice()関数より前に呼び出さなければなりません。

```
HANDLE
Pcie_SetDeviceAndEventName (
        BYTE    *lpszDeviceName,
        BYTE    *lpszEventName
)
```

[パラメータ]

lpszDeviceName

デバイス名を指定します。

lpszEventName

イベント名を指定します。

[戻り値]

なし

2.2. Pcie_OpenDevice

[説明]

PCI/PCIe デバイスドライバのオープンを行います。

```
HANDLE
Pcie_OpenDevice(
        int         nDevNum
)
```

[パラメータ]

nDevNum

デバイス番号を指定します(0～)。本ドライバは複数デバイスを取り扱うことができるため、同じデバイスを複数毎挿入した場合は、0, 1, 2～の番号を割り振って管理します。(COM1/COM2 のようなイメージです)管理するデバイスが 1 枚しかない場合は 0 を指定して下さい。

[戻り値]

値	説明
INVALID_HANDLE_VALUE	ドライバオープンエラー
その他	正常終了

2.3. Pcie_CloseDevice

[説明]

PCI/PCIe デバイスドライバのクローズを行います。

```
void
Pcie_CloseDevice(
        HANDLE hDev
)
```

[パラメータ]

hDev

Pcie_OpenDevice()関数で取得したデバイスハンドルを指定します。

[戻り値]

なし

2.4. Pcie_WriteRegister32/Pcie_WriteRegister16/Pcie_WriteRegister8

[説明]

レジスタにライトします。IO/MMIO の違いはドライバ内部で判断していますので、ユーザーが気にする必要はありません。

```
int
Pcie_WriteRegister32(
        HANDLE      hDev,
        DWORD       bar,
        DWORD       offset,
        DWORD       value
)
int
Pcie_WriteRegister16(
        HANDLE      hDev,
        DWORD       bar,
        DWORD       offset,
        WORD        value
)
int
Pcie_WriteRegister8(
        HANDLE      hDev,
        DWORD       bar,
        DWORD       offset,
        BYTE        value
)
```

[パラメータ]

hDev

Pcie_OpenDevice()関数で取得したデバイスハンドルを指定します。

bar

書き込みたいレジスタのベースアドレス番号を指定します。

offset

書き込みたいレジスタのオフセットを指定します。

value

書き込みたい値を指定します。

[戻り値]

値	説明
PCIE_NORMAL	正常終了
PCIE_ERR_PARAM	ハンドル異常 (ハンドルが正しいか確認して下さい)
PCIE_ERR_REG_WRITE	レジスタライトエラー (bar/offset が正しいか確認して下さい)

2.5. Pcie_ReadRegister32/Pcie_ReadRegister16/Pcie_ReadRegister8

[説明]

レジスタをリードします。IO/MMIO の違いはドライバ内部で判断していますので、ユーザーが気にする必要はありません。

```
int
Pcie_ReadRegister32(
        HANDLE      hDev,
        DWORD       bar,
        DWORD       offset,
        DWORD       *pvalue
)
int
Pcie_ReadRegister16(
        HANDLE      hDev,
        DWORD       bar,
        DWORD       offset,
        WORD        *pvalue
)
int
Pcie_ReadRegister8(
        HANDLE      hDev,
        DWORD       bar,
        DWORD       offset,
        BYTE        *pvalue
)
```

[パラメータ]

hDev

Pcie_OpenDevice()関数で取得したデバイスハンドルを指定します。

bar

書き込みたいレジスタのベースアドレス番号を指定します。

offset

書き込みたいレジスタのオフセットを指定します。

*pvalue

リードした結果を格納する変数のポインタを指定します。

[戻り値]

値	説明
PCIE_NORMAL	正常終了
PCIE_ERR_PARAM	ハンドル異常 (ハンドルが正しいか確認して下さい)
PCIE_ERR_REG_READ	レジスタリードエラー (bar/offset が正しいか確認して下さい)

2.6. Pcie_WriteBlock

[説明]

MMIO 空間に 32bit アクセスで連続ライトします。マザーボードによってはバースト転送となります。

```
int
Pcie_WriteBlock(
        HANDLE      hDev,
        DWORD       bar,
        DWORD       offset,
        DWORD       size,
        void        *buffer
)
```

[パラメータ]

hDev

Pcie_OpenDevice()関数で取得したデバイスハンドルを指定します。

bar

書き込みたい MMIO 空間のベースアドレス番号を指定します。

offset

書き込みたい MMIO 空間のオフセットを指定します。

size

書き込みたいサイズを BYTE 単位で指定します。(4 の倍数であること)

buffer

書き込みたいデータが格納されているバッファを指定します。

[戻り値]

値	説明
PCIE_NORMAL	正常終了
PCIE_ERR_PARAM	ハンドル異常 (ハンドルが正しいか確認して下さい)
PCIE_ERR_BLK_WRITE	ライトエラー (bar/offset/size が正しいか確認して下さい)

2.7. Pcie_ReadBlock

[説明]

MMIO 空間から 32bit アクセスで連続リードします。

```
int
Pcie_ReadBlock(
        HANDLE      hDev,
        DWORD       bar,
        DWORD       offset,
        DWORD       size,
        void        *buffer
)
```

[パラメータ]

hDev

Pcie_OpenDevice()関数で取得したデバイスハンドルを指定します。

bar

読み込みたい MMIO 空間のベースアドレス番号を指定します。

offset

読み込みたい MMIO 空間のオフセットを指定します。

size

読み込みたいサイズを BYTE 単位で指定します。(4 の倍数であること)

buffer

読み込んだデータを格納するバッファを指定します。

[戻り値]

値	説明
PCIE_NORMAL	正常終了
PCIE_ERR_PARAM	ハンドル異常 (ハンドルが正しいか確認して下さい)
PCIE_ERR_BLK_READ	リードエラー (bar/offset/size が正しいか確認して下さい)

2.8. Pcie_RegisterIntr

[説明]

割り込みを登録します。登録後は、イベントハンドルがシグナル状態になったことで割り込み発生を検知できるようになります。また、イベントハンドルは手動リセットハンドルなので、シグナル状態後に再度割り込みを検知する場合は、アプリケーション側で非シグナル状態にして下さい。(dma サンプル参照)Pcie_UnRegisterIntr()を呼び出すまでは本関数を再度呼び出すとエラーとなります。

```
int
Pcie_RegisterIntr(
        HANDLE      hDev,
        DWORD       ctrlbase,
        DWORD       ctrloffset,
        DWORD       ctrltype,
        DWORD       maskvalue,
        DWORD       statusbase,
        DWORD       statusoffset,
        DWORD       statustype,
        DWORD       intrvalue,
        HANDLE      *phEvent
)
```

[パラメータ]

hDev

Pcie_OpenDevice()関数で取得したデバイスハンドルを指定します。

ctrlbar

割り込みコントロールレジスタのベースアドレス番号を指定します。

ctrloffset

割り込みコントロールレジスタのオフセットを指定します。

ctrltype

割り込みコントロールレジスタのタイプを指定します。

(0:32bit 1:16bit 2:8bit アクセス)

maskvale

割り込みをマスクするときのコントロールレジスタへの書き込み値を指定します。

statusbar

ステータスレジスタのベースアドレス番号を指定します。

statusoffset

ステータスレジスタのオフセットを指定します。

statustype

ステータスレジスタのタイプを指定します。

(0:32bit 1:16bit 2:8bit アクセス)

intrvale

割り込みが発生したときのステータス値を指定します。

phEvent

割り込みが発生したときのイベントを格納するポインタを指定します。

[戻り値]

値	説明
PCIE_NORMAL	正常終了
PCIE_ERR_PARAM	ハンドル異常 (ハンドルが正しいか確認して下さい)
PCIE_ERR_REG_INTR	割り込み登録エラー (ctrlbar/ctrloffset/statusbar/statusoffset が正しいか確認して下さい)

[注意]

割り込みの設定を間違えると、OS がフリーズしますので、この関数を使用して動作確認を行う場合は、破損してもよい PC でご確認下さい。

割り込みコントロールレジスタの例

割り込みコントロールレジスタ(base 0 offset 0)

bit31						bit2	bit0
						PUSH	DMA

共に 1 で割り込み許可(初期値 0)

割り込みステータスレジスタ(base 0 offset 4)

bit31						bit2	bit0
						PUSH	DMA

共に 1 で割り込み発生、1で要因クリア(初期値 0)

登録方法

```
Pcie_RegisterIntr(
        hDev,           // デバイスハンドル
        0,              // コントロールレジスタベース番号
        0x00,           // コントロールレジスタ offset
        0,              // 32bit アクセス
        0,              // 割り込みをマスクする値
        0,              // ステータスレジスタベース番号
        0x04,           // ステータスレジスタ offset
        0,              // 32bit アクセス
        0x03,           // 割り込みが発生したときの値(DMA 割り込みだけの場合は 0x01)
        phEvent);       // イベントハンドルポインタ
```

割り込みマスクレジスタの例

割り込みマスクレジスタ(base 0 offset 0)

bit31						bit2	bit0
						PUSH	DMA

共に 1 で割り込み不許可(初期値 0)

割り込みステータスレジスタ(base 0 offset 4)

bit31						bit2	bit0
						PUSH	DMA

共に 1 で割り込み発生、1で要因クリア(初期値 0)

登録方法

```
Pcie_RegisterIntr(
        hDev,           // デバイスハンドル
        0,              // コントロールレジスタベース番号
        0x00,           // コントロールレジスタ offset
        0,              // 32bit アクセス
        0x03,           // 割り込みをマスクする値
        0,              // ステータスレジスタベース番号
        0x04,           // ステータスレジスタ offset
        0,              // 32bit アクセス
        0x03,           // 割り込みが発生したときの値(DMA 割り込みだけの場合は 0x01)
        phEvent);       // イベントハンドルポインタ
```

ドライバの割り込み検知処理

割り込みマスク状態を第 4 引数で指定された値と違う状態の場合を割り込み許可状態と判断し、その場合にステータスレジスタが第 10 引数で指定した値とANDをとった結果が真であった場合に割り込みありと判断し、第4引数をレジスタに書き込み割り込みをマスクします。

アプリケーションは、ステータスレジスタから割り込み要因を判断してから、要因クリアを行い、マスクを解除して下さい。

レジスタが上記のような構成になっていない場合は割り込み使用不可能です。またはドライバのカスタマイズの必要がありますので弊社までご相談下さい。

2.9. Pcie_UnRegisterIntr

[説明]

割り込みの登録を解除します。

```
int
Pcie_UnRegisterIntr(
        HANDLE          hDev,
        HANDLE          hEvent
)
```

[パラメータ]

hDev

Pcie_OpenDevice()関数で取得したデバイスハンドルを指定します。

hEvent

Pcie_RegisterIntr()関数で取得したイベントハンドルを指定します。

[戻り値]

値	説明
PCIE_NORMAL	正常終了
PCIE_ERR_PARAM	ハンドル異常 (ハンドルが正しいか確認して下さい)
PCIE_ERR_UNREG_INTR	登録解除エラー

2.10. Pcie_GetDMAMem

[説明]

DMA 転送用メモリを獲得します。Pcie_RelDMAMem()を呼び出すまでは本関数を再度呼び出すとエラーとなります。DMA 転送を行うバッファをカーネル内で取得しますので、最大 32MB 程度までしかとれません。また、メモリフラグメンテーションが発生すると、大きな領域はとれなくなりますので、OS 起動後なるべく早く取得して下さい。

```
int
Pcie_GetDMAMem(
        HANDLE      hDev,
        DWORD       size,
        void        **phy,
        void        **usr
)
```

[パラメータ]

hDev

Pcie_OpenDevice()関数で取得したデバイスハンドルを指定します。

size

獲得メモリの大きさを BYTE 単位で指定します。

phy

DMA 転送バッファの物理アドレスを格納するポインタを指定します。

usr

DMA 転送バッファのアプリケーションが使用するアドレスを格納するポインタを指定します。

[戻り値]

値	説明
PCIE_NORMAL	正常終了
PCIE_ERR_PARAM	ハンドル異常 (ハンドルが正しいか確認して下さい)
PCIE_ERR_GET_DMA	メモリ取得エラー (size を小さくして下さい)

2.11. Pcie_RelDMAMem

[説明]

DMA 転送バッファを開放します。

```
int
Pcie_RelDMAMem(
        HANDLE          hDev
)
```

[パラメータ]

hDev

Pcie_OpenDevice()関数で取得したデバイスハンドルを指定します。

[戻り値]

値	説明
PCIE_NORMAL	正常終了
PCIE_ERR_PARAM	ハンドル異常 (ハンドルが正しいか確認して下さい)
PCIE_ERR_REL_DMA	開放エラー

2.12. Pcie_WritePciConfig

[説明]

PCI コンフィグレーション空間にライトします。PCI の場合は、256BYTE までのコンフィグレーション空間、PCIExpress の場合は 4096BYTE までのコンフィグレーション空間を指定できます。

```
int
Pcie_WritePciConfig(
        HANDLE      hDev,
        DWORD       size,
        DWORD       offset,
        void        *buffer
)
```

[パラメータ]

hDev

Pcie_OpenDevice()関数で取得したデバイスハンドルを指定します。

size

書き込むサイズを BYTE 単位で指定します。

offset

書き込む offset を指定します。

buffer

書き込む値が格納されているバッファを指定します。

[戻り値]

値	説明
PCIE_NORMAL	正常終了
PCIE_ERR_PARAM	ハンドル異常 (ハンドルが正しいか確認して下さい)
PCIE_ERR_WRITE_CFG	ライトエラー (offset を確認して下さい)

2.13. Pcie_ReadPciConfig

[説明]

PCI コンフィグレーション空間からリードします。PCI の場合は、256BYTE までのコンフィグレーション空間、PCIExpress の場合は 4096BYTE までのコンフィグレーション空間を指定できます。

```
int
Pcie_ReadPciConfig(
        HANDLE      hDev,
        DWORD       size,
        DWORD       offset,
        void        *buffer
)
```

[パラメータ]

hDev

Pcie_OpenDevice()関数で取得したデバイスハンドルを指定します。

size

読み込むサイズを BYTE 単位で指定します。

offset

読み込む offset を指定します。

Buffer

読み込む値を格納するバッファを指定します。

[戻り値]

値	説明
PCIE_NORMAL	正常終了
PCIE_ERR_PARAM	ハンドル異常 (ハンドルが正しいか確認して下さい)
PCIE_ERR_READ_CFG	リードエラー (offset を確認して下さい)

2.14. Pcie_IsDMADone

[説明]

DMA 転送を正常に行えたか判断します

```
int
Pcie_IsDMADone(
        HANDLE      hDev,
        DWORD       hEvent,
        DWORD       timeout
)
```

[パラメータ]

hDev

Pcie_OpenDevice()関数で取得したデバイスハンドルを指定します。

hEvent

Pcie_RegisterIntr()関数で取得したイベントハンドルを指定します。
注:Linux では Pcie_RegisterIntr()関数が使用出来ないので、
引数として 0 を受け取っています。

timeout

タイムアウト値を指定します。

[戻り値]

値	説明
PCIE_NORMAL	正常終了
PCIE_ERR_PARAM	パラメータ異常
PCIE_ERR_DMA_DONE	DMA 転送異常