特電 EZ-USB FX2
インタフェースコア Ver2
説明書

Rev 1.1

平成 22 年 1 月 10 日

特殊電子回路株式会社

1．はじめに

特電 EZ-USB FX2 インタフェースコア Ver2（以下、本コアと略す）は、Cypress Semiconductor 社の EZ-USB FX2 と、FPGA 内のユーザロジックをインタフェースするためのコアです。

ユーザは、EZ-USB FX2 の難しい設定や仕様を気にすることなく、簡単かつ高速にデータ通信を行うことができます。

EZ-USB FX2 は、その名前とは裏腹に、デバイスメーカーから提供されたサンプルの範囲を超えた使い方をしようとすると非常に難易度が高いものになります。EZ-USB FX2 の性能を最大限引き出すためには SlaveFIFO という機能を使わなければなりませんが、SlaveFIFO と FPGA 内のユーザ回路を接続するには、非常に難解な設定やロジック設計、そしてソフトウェアの設計が必要になります。

本コアは、そのような面倒で難しい側面をすべて隠蔽して、EZ-USB FX2 を単純なインタフェースとして使うことができるようにするものです。本コアが FPGA 内のユーザロジックに対して提供するインタフェースは、FIFO のような単純なものです。

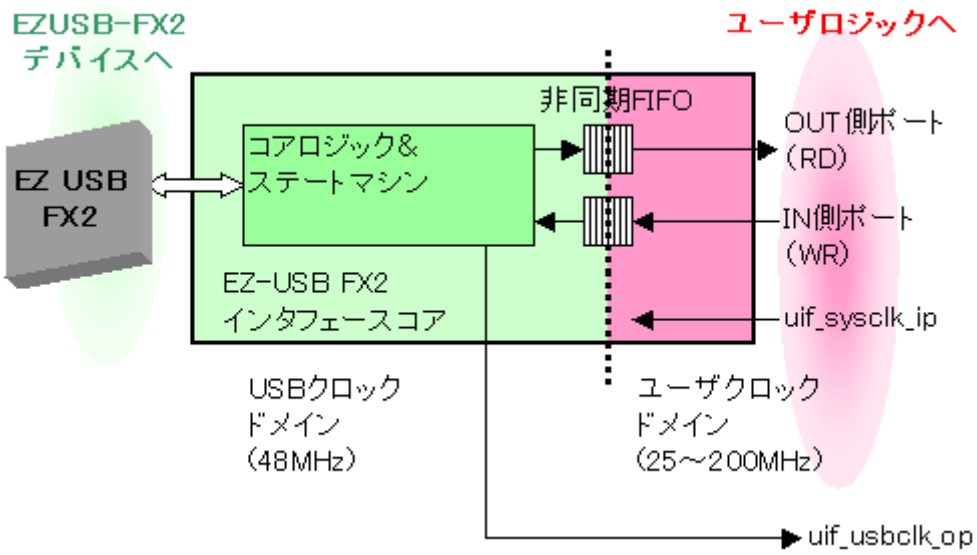

図 1　本コアの構成

なお、本コアは当社の Spartan-6 評価ボードでのみ動作します。他のボードでの動作は一切保障いたしませんのでご注意ください。

２．シンボル

図２に、本コアのシンボルを示します。

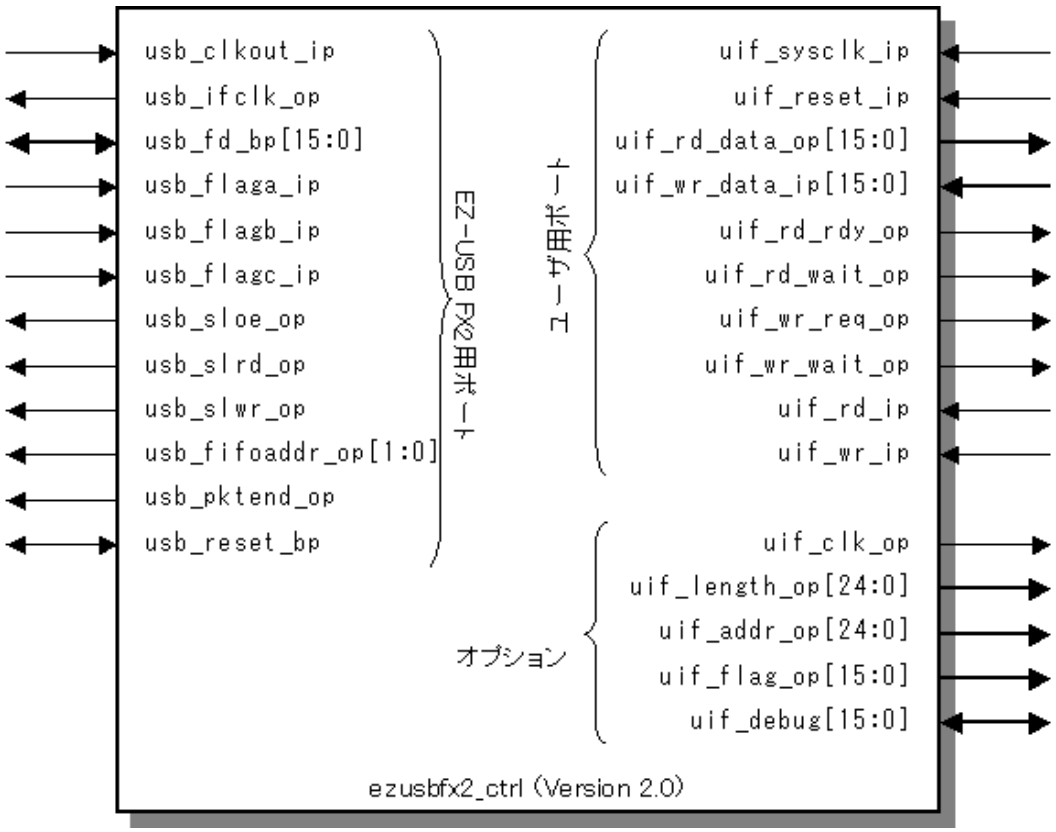

図 2 本コアのシンボル

本コアには大きく分けて、EZ-USB FX2 デバイスと接続するためのポートと、ユーザロジックと接続するためのポートがあります。

図 2 で、左側に描かれたポートの信号は EZ-USB FX2 デバイスと接続するためのポートです。そのまま FPGA の外に取り出し、EZ-USB FX2 に直結してください。

図 2 で、右上に描かれたポートは、ユーザロジックを接続するためのポートです。ここにはユーザの作成したロジックを接続します。

図 2 で、右下に描かれたポートはオプション用の信号出力ポートです。

3．IP コアを構成するファイル

本 IP コアは、次のファイルから構成されます。

- ezusbfx2_ctrl.ngc
  本 IP コアの本体です。
- afifo.ngc
  非同期 FIFO です。

これら 2 つのファイルを、ISE プロジェクトのあるディレクトリに格納してください。Add File などでプロジェクトに追加する必要はありません。次の図 3 のように、？マークが出ていても問題ありません。

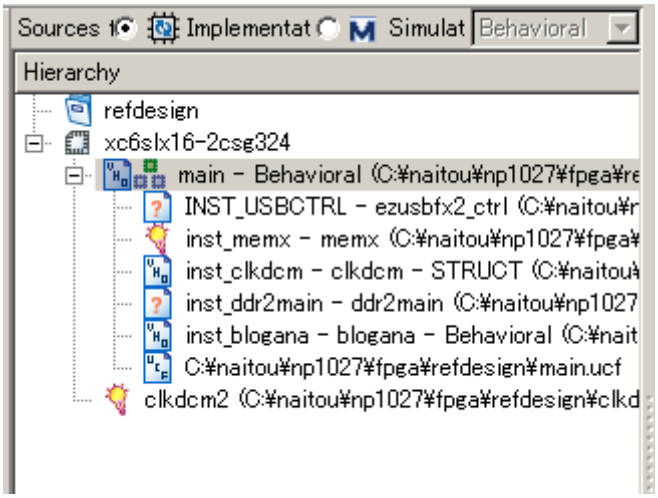

図 3　ISE でのプロジェクトツリー

４．入出力ポート

４．１　ポートの概要

EZ-USB FX2 を接続用ポートの機能を、次の表 1 に示します。

これらの信号は EZ-USB FX2 の対応するポートへ接続してください。

表 1　EZ-USB FX2 用入出力ポート

信号名	方向	機能
usb_clkout_ip	入力	48MHz のクロック入力
usb_ifclk_op	出力	48MHz のクロック出力
usb_fd_bp [15:0]	双方向	データバス
usb_flaga_ip	入力	予約
usb_flagb_ip	入力	データ書き込み用フラグ
usb_flagc_ip	入力	データ読み出し用フラグ
usb_sloe_op	出力	スレーブ FIFO 出力イネーブル
usb_slrd_op	出力	スレーブ FIFO リードイネーブル
usb_slwr_op	出力	スレーブ FIFO ライトイネーブル
usb_fifoaddr_op [1:0]	出力	スレーブ FIFO アドレス選択
usb_pktend_op	出力	短いパケットを送信
usb_reset_bp	双方向	リセット信号

ユーザロジック接続用ポートの機能を、次の表 2 に示します。

表 2　ユーザロジック接続用入出力ポート

信号名	方向	機能
uif_sysclk_ip	入力	ユーザ回路クロック入力
uif_reset_ip	入力	ユーザリセット入力
uif_rd_data_op [15:0]	出力	読み出しデータバス
uif_wr_data_ip [15:0]	入力	書き込みデータバス
uif_rd_rdy_op	出力	USB 読み出しデータあり
uif_rd_wait_op	出力	USB 読み出し一時停止要求
uif_wr_req_op	出力	USB データ書き込み要求
uif_wr_wait_op	出力	USB 書き込み一時停止要求
uif_rd_ip	入力	データ読み出しストローブ
uif_wr_ip	入力	データ書き込みストローブ

オプション用ポートの機能を、次の表 3 に示します。

表 3　オプション用入出力ポート

信号名	方向	機能
uif_usbclk_op	出力	USB クロック出力(48MHz)
uif_length_op[24:0]	入力	データ長
uif_addr_op[24:0]	入力	転送開始アドレス
uif_flag_op[15:0]	入力	転送フラグ
uif_debug[15:0]	双方向	デバッグ用

- uif_reset_ip を'1'にすると、USB インタフェース IP コアの内部レジスタがリセットされると共に、usb_reset_bp から'0'が出力され、EZ-USB FX2 にもリセットがかかります。
- uif_usbclk_op は EZ-USB FX2 デバイスから受信したクロック(48MHz)が出力されます。USB クロック(48MHz)に同期させて動作させたい場合には、ユーザ回路用のシステムクロックに uif_usbclk_op を使ってください。
- uif_length_op[15:0]は、送受信データサイズを示します。単位はワード（2Bytes）単位です。なお、この信号を使わなくても、転送終了のタイミングを知ることができるので、必要がなければ使わなくても構いません。
- uif_addr_op[24:0]は、現在のコアでは使用しません。オープンのままにしておいてください。
- uif_flag_op[15:0]は、転送の際に PC 上のソフトウェアから FPGA に付加的な情報を送るためのフラグです。この信号は、ユーザ回路内でのデータの転送先や転送モードなどを示すなど、自由に使用することができます。使用しない場合はオープンにしておいてください。
- uif_debug[15:0]信号はオープンのままにしておいてください。

４．２　データ受信（OUT 転送）と関連する信号

① PC 上のソフトウェアで**データ書き込み関数**がコールされると、本コア内の FIFO に先頭の数百バイトのデータが格納され、uif_rd_rdy_op 出力が'1'になります。uif_rd_rdy_op は USB からの読み出しデータが存在することを示します。
この時点では、まだデータは出力されていません。

② FIFO に十分な量のデータが溜まると、uif_rd_wait_op が'0'になって、データが読み出し可能になります。FIFO の残量が十分でないと uif_rd_wait_op が'1'になって、取り出せるデータがないか少ないことを示します。
ユーザロジックは、(uif_rd_rdy_op = '1' and uif_rd_wait_op = '0') という条件が満たされた場合に、uif_rd_ip を'1'にするようにしてください。

③ ユーザ回路がデータを取り込んだら、uif_rd_ip を'1'にしてコアに伝えます。コアは、uif_sysclk_ip クロックの立ち上がりで次のデータを出力します。

④ 最後のデータが読み出されたら、uif_rd_rdy_op は'0'に戻ります。

図 4 は、データ読み出し波形の例です。

SYSCLK が 50MHz 程度の場合、FIFO に溜まる速度と読み出す速度がほぼ一致します。したがって、USB の 1 パケットに収まるサイズの読み出し時には、ほとんど WAIT は発生しません。

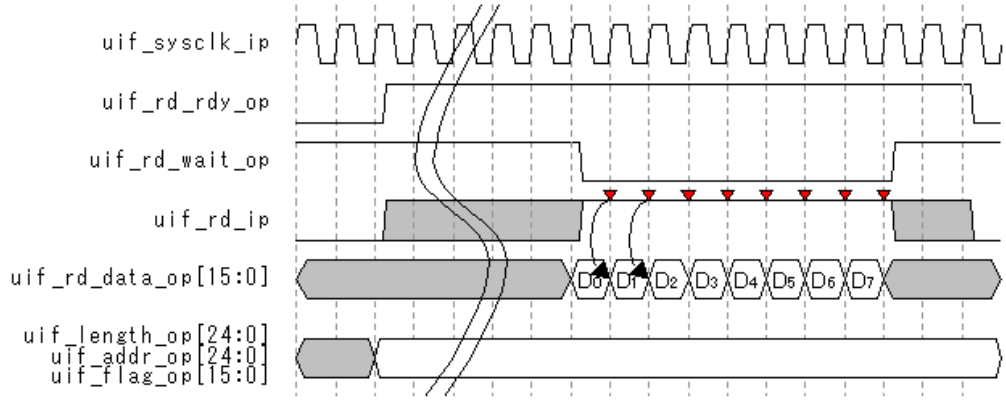

図 4　USB からのデータ読み出し波形（$f_{USBCLK} \fallingdotseq f_{SYSCLK}$）

図 5 は、SYSCLK が USB クロックよりも速い場合の波形の例です。この場合、FIFO に溜まる速さよりも読み出す速さのほうが速いため、WAIT が発生します。

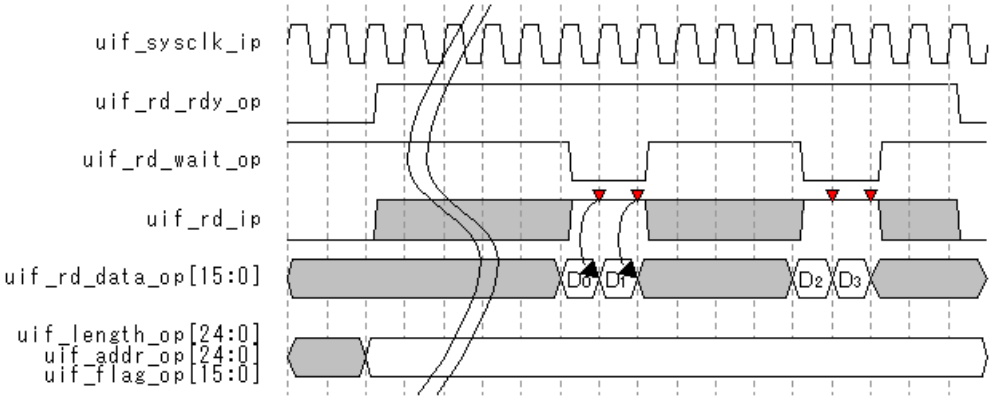

図 5　USB からのデータ読み出し波形（$f_{USBCLK} << f_{SYSCLK}$）

補足説明

- usb_rd_wait_op が'1'になって WAIT が発生した場合でも、usb_rd_ip を'0'に戻す必要はありません。ただし、実際のリード動作は保留されます。
- USB からの受信がはじまって、usb_rd_rdy_op が最初に'0'から'1'に変化した時点では、FIFO にはまだ十分な量のデータが溜まっていません。そのため、usb_rd_wait_op は'1'の状態からスタートします。
- SYSCLK の速度が USBCLK(48MHz)よりも十分に速い場合、WAIT が頻繁に発生します。
- FIFO の容量は 2048 バイトです。（将来的に変更の可能性あり）
- PC 上の**データ書き込み関数**は、オプションの引数で flag という 16 ビットの値をとることができます。この関数で指定した flag は、uif_flag_op[15:0]から出力されます。
- uif_length_op [24:0], uif_addr_op[24:0], uif_flag_op[15:0] のオプション信号は、uif_rd_rdy_op が有効になった時点で有効になっています。

４．３　データ送信（IN 転送）と関連する信号

① PC 上のソフトウェアで**データ読み出し関数**がコールされると、uif_wr_req_op 出力が'1'になります。この信号は、PC 上のソフトウェアが FPGA にデータの送信を要求していることを示すフラグです。

② ユーザ回路は、uif_wr_data_ip[15:0]に書き込みたいデータを与え、uif_wr_ip を'1'にします。すると、uif_sysclk_ip の立ち上がりでサンプリングされ、FIFO に書き込まれます。

③ uif_wr_ip を'0'にすることで、任意のタイミングで書き込みを一時停止することができます。

④ FIFO の空き容量が十分でないと uif_wr_wait_op が'1'になります。その場合、ユーザロジックは次のタイミングで uif_wr_ip を'0'にして、データの書き込みを一時的に停止してください。

⑤ 最後のデータを書き込むと、uif_wr_req_op は'0'に戻ります。

図 6 は、データ書き込み波形の例です。

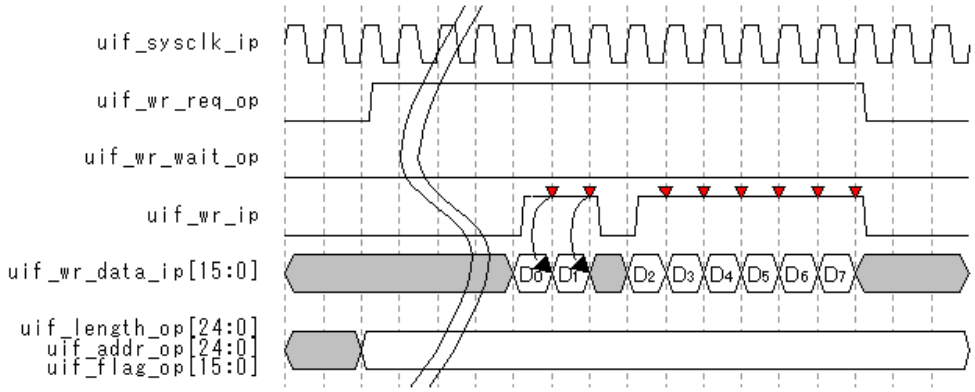

図 6　USB へのデータ書き込み波形（f_{USBCLK} ≒ f_{SYSCLK}）

図 7 は、SYSCLK が USB クロックよりも速い場合の波形の例です。この場合、FIFO から読み出す速さよりも書き込む速さのほうが速いため、WAIT が発生します。ただし、FIFO は 2048 バイト（1024 ワード）の深さがあるため、短いデータ書き込みの場合は WAIT は発生しません。

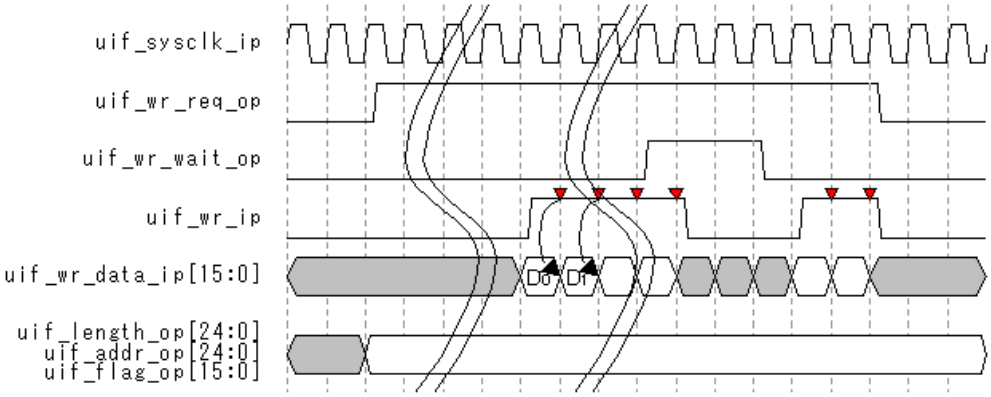

図 7　USB からのデータ読み出し波形（$f_{USBCLK} << f_{SYSCLK}$）

補足事項

- WAIT 発生時は、必ず uif_wr_ip を'0'にして転送を停止してください。
- usb_wr_wait_op はフラグが立ってからも 2 個程度の余裕があり、すぐに FIFO が溢れることはありませんが、できるだけ速やかに次のデータの書き込みを停止してください。
- FIFO が溢れた場合の動作およびデータ内容は保障されません。
- FIFO の容量は 2048 バイトです。（将来的に変更の可能性あり）
- PC 上の**データ読み出し関数**は、オプションの引数で flag という 16 ビットの値をとることができます。この関数で指定した flag は、uif_flag_op[15:0] から出力されます。
- uif_length_op [24:0], uif_addr_op[24:0], uif_flag_op[15:0] のオプション信号は、uif_rd_rdy_op が有効になった時点で有効になっています。

５．ソフトウェア

５．１　ファイルの構成

本 IP コアを操作するソフトウェアは表 1 に示すファイルから構成されます。

表 1　IP コアを操作するためのファイル

種類	ファイル名
デバイスドライバ	tkdnsp6.sys
ユーザ・インタフェース用 DLL	tkusb.dll
DLL 用インクルードファイル	tkusb.h
インポートライブラリ（Visual C++用）	tkusb_vc.h
インポートライブラリ（Borland C++用）	tkusb_bcc.lib
モジュール定義ファイル	tkusb.def

ヘッダファイルとライブラリファイルは、ユーザプログラムのビルド時に使用します。tkusb.dll は、実行するアプリケーション・ソフトウェアの実行ファイルと同じディレクトリに格納してください。tkdnsp6.sys は、インストーラを実行した際に、Windows のシステムディレクトリにインストールされます。

モジュール定義ファイルは、DLL でエクスポートされる関数を定義したファイルです。

５．２　API 関数の概要

ユーザ向けに公開されている関数は、ボード情報取得・操作、USB 基本操作、USB-JTAG 操作の 3 種類に分けられます。

次節では、これらの関数の使用方法を説明します。

５．３　API 関数の詳細

５．３．１　ボード情報取得・操作関数

表 2 に、ボード情報取得・操作関数の一覧を示します。これらの関数は、ドライバやファームウェアのバージョン管理、ボード上に記録されたユーザ情報の設定や取得、ボード上の EEPROM の読み書きなどを行います。

表 2 ボード情報取得・操作関数の一覧

```
int WINAPI USBEnumerate ();
BOOL WINAPI USBGetBoardInfo(USBBoardInfo *result);
BOOL WINAPI USBSetBoardUserId(unsigned char id[32]);
unsigned long WINAPI USBGetDriverVersion(void);
unsigned long WINAPI USBGetFirmwareVersion(void);
BOOL WINAPI USBReadEEPROM(unsigned short addr,unsigned char *data,int len);
BOOL WINAPI USBWriteEEPROM(unsigned short addr,unsigned char *data,int len);
BOOL WINAPI USBUnlockEEPROMFirmArea(const char *keyword);
```

本ボードは、EZ-USB FX2 デバイスのファームウェア格納用に 8k バイトの EEPROM を接続しています。この EEPROM の前半 4k バイトはファームウェアが使用し、後半 4k バイトはユーザ用に解放しています。特に、ユーザ情報エリアのうち、0x1FE0 から 0x1FFF までの 32 バイトは、ボードの**識別情報**エリアとして利用されています。

USBReadEEPROM 関数や USBWriteEEPROM 関数を使うと、この EEPROM にアクセスすることができます。また、USBSetBoardUserId 関数や USBGetBoardInfo 関数を使うと、**識別情報**を読み出したり、設定したりすることができます。

以下、個々の関数について詳細に説明します。

関数名	USBEnumerate
凡例	int WINAPI USBEnumerate();
説明	システムに接続されたボードの数を返します。
引数	なし
戻り値	USB ドライバのインスタンスの数を返す。 0 なら、ボードは接続されていない(エラーではない) -1 なら、ドライバや DLL で何らかのエラーが発生している 正数なら、発見されたボードの数
注意事項	頻繁に USB ケーブルを抜き差しすると、1 台のボードを複数回カウントしてしまうことがあります。その場合、実際のボードの数は本関数の戻り値よりも少なくなります。 **この関数はまだ実装されておらず、将来において仕様が変更される可能性があります。**

関数名	USBGetBoardInfo
凡例	BOOL WINAPI USBGetBoardInfo(USBBoardInfo *result);
説明	現在開いている USB-FPGA ボードの情報を返します。
引数	結果が格納される USBBoardInfo 型構造体のアドレス
戻り値	取得が成功したら TRUE、失敗したら FALSE

注意事項	USBBoardInfo 型構造体の定義を以下に示します
	
	typedef struct USBBoardInfo {
	  int id; // USBOpenById()関数で指定するべき ID
	  WCHAR descriptor[32]; // USB の String Descriptor
	  unsigned char option[32]; // 各ボードに書き込まれた識別情報
	} USBBoardInfo;
	
	descriptor は TKDN-SP6-BRD という文字列がセットされています。
	option は、ボードの**識別情報**です。

関数名	USBSetBoardUserId
凡例	BOOL WINAPI USBSetBoardUserId(unsigned char id[32]);
説明	現在オープンしているボードに対して識別情報を設定します。
引数	id　識別情報が格納されたバッファ
戻り値	成功したら TRUE を返す 失敗したら FALSE を返す
注意事項	

関数名	USBGetDriverVersion
凡例	unsigned long WINAPI USBGetDriverVersion(void);
説明	デバイスドライバ(SYS ファイル)のバージョンを返します。
引数	なし
戻り値	ドライバのバージョン。 [31:16]はメジャーバージョン。[15:0]はマイナーバージョン。 現在のバージョンは 1.x であるので、0x0001XXXX を返す。
注意事項	**この関数はまだ実装されていません。**

関数名	USBGetFirmwareVersion
凡例	unsigned long WINAPI USBGetFirmwareVersion(void);
説明	ファームウェア(EZ-USB FX2 のマイコン上の実行ファイル)のバージョンを返します。
引数	なし
戻り値	ファームウェアのバージョン。 [31:16]はメジャーバージョン。[15:0]はマイナーバージョン。 現在のバージョンは 1.x であるので、0x0001XXXX を返す。
注意事項	**この関数はまだ実装されていません。不定な値を返します。**

関数名	USBReadEEPROM
凡例	BOOL WINAPI USBReadEEPROM(unsigned short addr, unsigned char *data,int len);
説明	EEPROM のデータを読み出します
引数	addr　読み出したい EEPROM のアドレスを指定します data　読み出したいデータが格納されたバッファを指定します len　読み出したい長さを指定します
戻り値	成功したら TRUE を返し、data で示されたバッファに読み出し結果を格納する。 失敗したら FALSE を返す
注意事項	アドレスとデータは 32 バイト単位で指定してください。

関数名	USBWriteEEPROM
凡例	BOOL WINAPI USBWriteEEPROM (unsigned short addr, unsigned char *data,int len);
説明	EEPROM にデータを書き込みます
引数	addr　書き込みたい EEPROM のアドレスを指定します data　書き込みたいデータが格納されたバッファを指定します len　書き込みたい長さを指定します
戻り値	成功したら TRUE を返す 失敗したら FALSE を返す
注意事項	アドレスとデータは 32 バイト単位で指定してください。 デフォルトではファームウェア領域 (0x0000～0x0FFF)は書き換えることができません。ファームウェア領域を書き換えるには次の USBGetFirmwareVersion 関数を使用します。

関数名	USBUnlockEEPROMFirmArea
凡例	BOOL WINAPI USBUnlockEEPROMFirmArea(const char *keyword);
説明	WriteEEPROM でファームウェアを書き換えられるようにする。
引数	keyword　"REWRITE"と指定する。
戻り値	成功したら TRUE を返す 失敗したら FALSE を返す
注意事項	

５．３．２　USB 基本操作関数

表 3 に、USB 基本操作関数の一覧を示します。これらの関数は、Spartan-6 ボードの USB をオープン/クローズと、データの読み書きを行います。

表 3 USB 基本操作関数

```
BOOL WINAPI USBOpen();
BOOL WINAPI USBOpenByID(int id);
void WINAPI USBClose();
BOOL WINAPI USBWriteData(unsigned char *buffer, int len, unsigned short flag);
BOOL WINAPI USBReadData(unsigned char *buffer, int len, unsigned short flag);
BOOL WINAPI USBWriteSDRAM(unsigned long addr,unsigned char *buffer,int len,
                          unsigned short attr);
BOOL WINAPI USBReadSDRAM(unsigned long addr,unsigned char *buffer,int len,
                          unsigned short attr);
```

関数名	USBOpen
凡例	BOOL WINAPI USBOpen();
説明	USB ドライバをオープンします。
引数	なし
戻り値	オープンに成功すると TRUE を返し、失敗すると FALSE を返します。

関数名	USBOpenByID
凡例	BOOL WINAPI USBOpenByID(int id);
説明	ID を指定して、USB ドライバをオープンします。 複数のボードが接続されている場合に、特定のボードを指定してオープンする場合に使用します。
引数	id は 0～255 までの数値。PC に接続された後、PnP で見つかった順に 0 から割り当てられています。
戻り値	オープンに成功すると TRUE を返し、失敗すると FALSE を返します。

関数名	USBClose
凡例	void WINAPI USBClose();
説明	USB ドライバをクローズします。
引数	なし
戻り値	なし

関数名	USBWriteData
凡例	BOOL WINAPI USBWriteData(unsigned char *buffer, int len, unsigned short flag);
説明	**データ送信関数です。**USB に一連のデータを送信します。
引数	buffer に送信したいデータをセットします。 len には送信した長さをセットします。範囲は 1～33553919 です。 flag にはオプションで送信したい値をセットします。
戻り値	成功すると TRUE を返し、失敗すると FALSE を返します。

関数名	USBReadData
凡例	BOOL WINAPI USBReadData(unsigned char *buffer, int len, unsigned short flag);
説明	**データ受信関数**です。USB からデータを受信します。この関数は受信が完了するか、エラーが発生するまで、制御を戻しません。
引数	buffer には受信したデータが入るバッファへのポインタを指定します。 len には受信したい長さをセットします。範囲は 1～33553919 です。 flag にはオプションで送信したい値をセットします。
戻り値	成功すると TRUE を返し、失敗すると FALSE を返します。

関数名	USBWriteSDRAM
凡例	BOOL WINAPI USBWriteSDRAM(unsigned long addr,unsigned char *buffer, int len,unsigned short attr);
説明	SDRAM へのデータ送信関数です。
戻り値	常に FALSE を返します
注意	**この関数はまだ実装されていません。**

関数名	USBReadSDRAM
凡例	BOOL WINAPI USBReadSDRAM(unsigned long addr,unsigned char *buffer, int len,unsigned short attr);
説明	SDRAM からのデータ受信関数です。
戻り値	常に FALSE を返します
注意	**この関数はまだ実装されていません。**

５．３．３　USB-JTAG 操作関数

表 4 に、USB-JTAG 操作関数の一覧を示します。これらの関数は、Spartan-6 ボード上の FPGA に対して JTAG 経由でのコンフィギュレーションを行うためのものです。

表 4 USB-JTAG 操作関数

```
BOOL WINAPI USBSetSmajMode(BOOL enable);
BOOL WINAPI USBGetSmajMode(BOOL *result);
BOOL WINAPI USBConfigFPGA(unsigned char *bitStream)
```

本ボードに搭載された EZ-USB FX2 は、通常はユーザ・アプリケーション用のデータを交換するモードで動作していますが、**SmartJTAG モード**に切り替えることによって FPGA をコンフィギュレーションすることができるようになります。なお、SmartJTAG モードでは、ユーザ・アプリケーション用の通信はできません。

関数名	USBSetSmajMode
凡例	BOOL WINAPI USBSetSmajMode(BOOL enable);
説明	USB の動作モードを切り替えます。
引数	1 にすると、USB-JTAG モード。 0 にすると、通常動作モード。
戻り値	成功すると TRUE を返し、失敗すると FALSE を返します。

関数名	USBGetSmajMode
凡例	BOOL WINAPI USBGetSmajMode(BOOL *result);
説明	現在、SmartJTAG モードで動作しているかどうかを調べます。
引数	結果が格納されるバッファを result に指定します。
戻り値	成功すると TRUE を返し、失敗すると FALSE を返します。

関数名	USBConfigFPGA
凡例	BOOL WINAPI USBConfigFPGA(unsigned char *bitStream);
説明	FPGA を再コンフィギュレーションします。
引数	bitStream には、BitSteam ファイルのデータが格納されたバッファを指定してください。
戻り値	成功すると TRUE を返し、失敗すると FALSE を返します。
注意	**この関数はまだ実装されていません。**

５．４　API 関数の使用方法

最初に USBOpen 関数を呼び出します。

この関数が FALSE を返した場合は、PC にターゲットボードが接続されていないか、デバイスドライバが動作していないなどの理由により、EZ-USB FX2 と接続できない状況になっています。

USBOpen 関数が FALSE を返した場合は、ユーザプログラムでは、エラーメッセージを表示してください。

ドライバのオープンに成功したら、最初に SetSmajMode(FALSE)を呼び出し、通常動作モードに切り替えます。

その後、USBWriteData または USBReadData を呼び出し、アプリケーション・ソフトウェアと FPGA 間で通信を行います。

USBReadData 関数を呼び出すと、FPGA が必要な数のデータを送信して、PC が受信するまで制御を戻しませんのでご注意ください。

インタフェースの使用を終了する場合は、USBClose 関数を呼び出します。