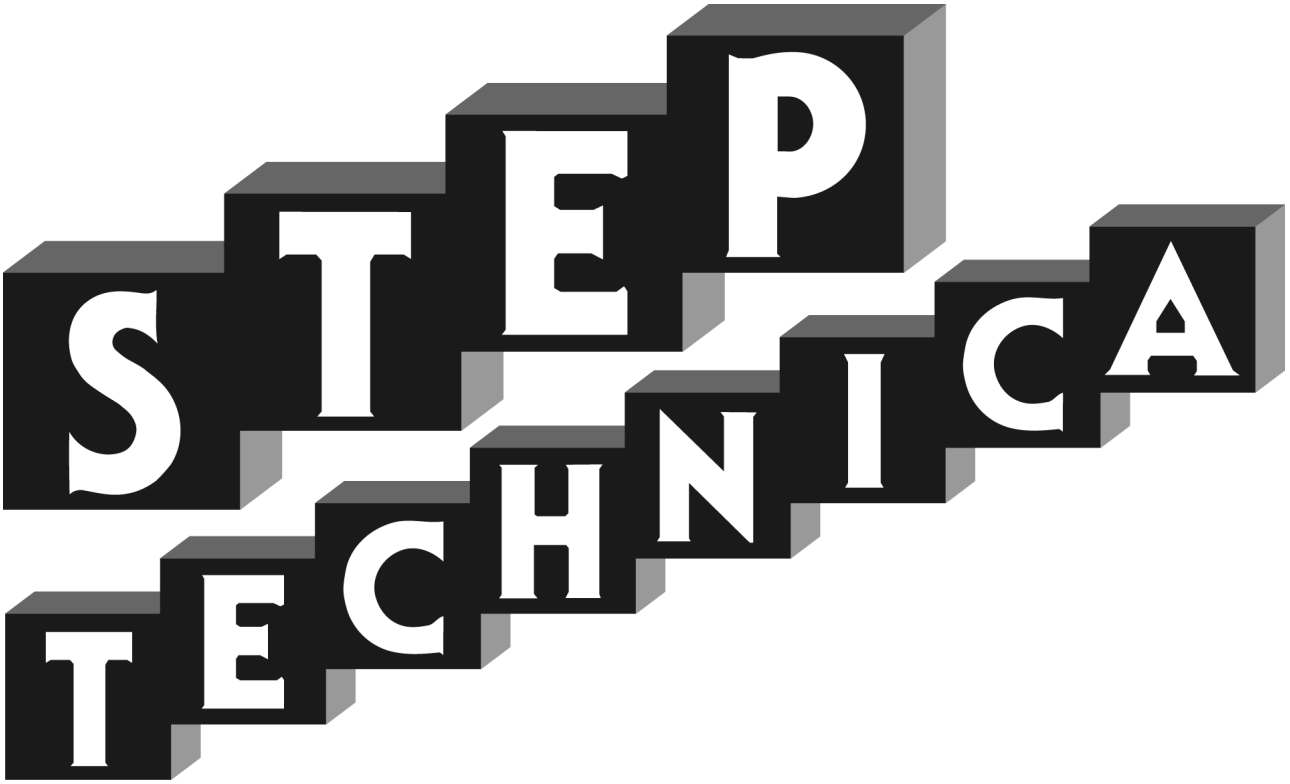
STD-HLSB36PCI1_V2.0J

STEP TECHNICA CO.,LTD.



HLS（MKY36）搭載 PCI ボード

HLSB-36PCI1

ユーザーズマニュアル

ご注意

1. 本書に記載された内容は、将来予告なしに変更する場合があります。本製品をご使用になる際には、本書が最新の版数であるかをご確認ください。

2. 本書において記載されている説明や回路例などの技術情報は、お客様が用途に応じて本製品を適切にご利用をいただくための参考資料です。実際に本製品をご使用になる際には、基板上における本製品の周辺回路条件や環境を考慮の上、お客様の責任においてシステム全体を十分に評価し、お客様の目的に適合するようシステムを設計してください。当社は、お客様のシステムと本製品との適合可否に対する責任を負いません。

3. 本書に記載された情報、製品および回路等の使用に起因する損害または特許権その他権利の侵害に関して、当社は一切その責任を負いません。

4. 本製品および本書の情報や回路などをご使用になる際、当社は第三者の工業所有権、知的所有権およびその他権利に対する保証または実施権を許諾致しません。

5. 本製品は、人命に関わる装置用としては開発されておりません。人命に関わる用途への採用をご検討の際は、当社までご相談ください。

6. 本書の一部または全部を、当社に無断で転載および複製することを禁じます。

はじめに

本マニュアルは、HLS 専用 IC の一品種である MKY36 を搭載した PCI ボードの HLSB-36PCI1 について記述します。
HLSB-36PCI1 の利用および本マニュアルの理解に先駆けて、“HLS 導入ガイド ” を必ずお読みください。

●対象読者
・HLS を初めて構築する方
・HLS を構築するために、弊社の HLSB-36PCI1 を初めてご利用になる方

●読者が必要とする知識
・ネットワーク技術に関する標準的な知識
・半導体製品（特にマイクロコントローラおよびメモリ）に関する標準的な知識

●関連マニュアル
・HLS　導入ガイド
・HLS　テクニカルガイド
・HLS　MKY36 ユーザーズマニュアル

【注意事項】
本書において記載されている一部の用語は、弊社の Web および営業用ツール（総合カタログ等）において記載されている用語とは異なっています。営業用ツールにおいては、様々な業界において弊社製品をご理解いただけるよう、一般的用語を用いています。
HLS ファミリに関する専門知識は、技術ドキュメント（マニュアル等）を基にご理解ください。

改訂履歴

Ver	日付	改訂内容	
		ページ	説明
Ver1.0J	2015 年 9 月	-	初版発行
Ver2.0J	2018 年 11 月	1-1	「表 1-1 ボード仕様」の内容を変更
		2-1	「2.1 概要」にて記載している対応 OS を変更

目　次

第１章　ハードウェア

第２章　ソフトウェア

図　目　次

表　目　次

第 1 章　ハードウェア

本章は、HLSB-36PCI1 のハードウェアについて記述します。

1.1 特徴

HLSB-36PCI1 は、MKY36 チップを搭載した PCI 拡張バス対応の HLS 通信ボードです。ステップテクニカ提供の Windows 用のライブラリと併せて利用することにより、MKY36 の機能を簡単に利用しやすいように設計されています。HLSB-36PCI1 は、MKY36 の評価および学習にご利用ください。

HLSB-36PCI1 には 8pin のモジュラコネクタを採用しており、100BASE-TX 用市販の CAT5 以上のストレートケーブルで動作を評価することができます。HLSB-36PCI1 の利用経験は、MKY36 を搭載したマイコンシステムにも活かす事ができます。

1.2 仕様

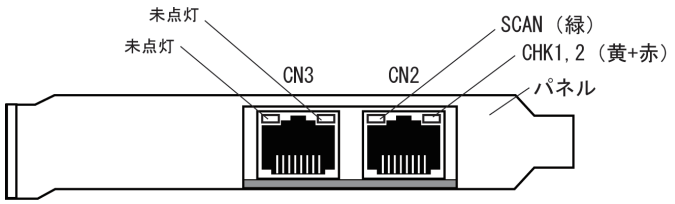
HLSB-36PCI1 の仕様を、表 1-1 に示します。

表 1-1　ボード仕様

HLS デバイス	MKY36　1 個
HLS 通信方式	全二重 / 半二重通信
HLS 通信速度	12M/6M/3Mbps（MKY36 レジスタにて設定）
HLS 通信コネクタ	RJ45 タイプ（8pin モジュラー）x 2 個
対応バス	PCI Ver2.2 準拠した、32 ビット・33MHz 拡張バス 5.0V/3.3V 対応
占有リソース	16KB の連続したメモリエリア（PnP にて自動割当）
割り込み	1 ライン使用（PnP にて自動割当）
対応 OS	Windows10　（64bit/32bit） Windows8.1　（64bit/32bit） Windows8　（64bit/32bit） Windows7　（64bit/32bit）
電源	DC +5.0V
消費電流	500mA 以下
動作環境	温度 0 ～ 50℃　湿度 20 ～ 90%（非結露）
保存環境	温度 0 ～ 80℃　湿度　0 ～ 90%（非結露）
外形寸法	119.9mm(W) × 64.4mm(D) ※パネル部含まず（Low Profile 対応）
付属品	Low Profile 用ブラケット

1.3 コネクタ仕様

HLSB-36PCI1 のパネル面とその詳細を図 1-1 に示します。



ピン番号	機能	
	全二重	半二重
3	RX+	--
4	TX-	TRX-
5	TX+	TRX+
6	RX-	--
8	シールド	シールド
1,2,7	未使用	未使用

SCAN	MKY36 の #SCANL 端子に接続した LED（緑）。MKY36 がスキャン中である時点灯します。
CHK1、2	MKY36 の #CHK1L 端子に接続した LED（黄）または #CHK2L 端子に 接続した LED（赤）が表示されます。CHECK-1 または CHECK-2 発生時に点灯します。 ただし、CHECK-1 と CHECK-2 が同時に発生した場合は LED（赤）が点灯します。
CN2、CN3	HLS 通信ライン。市販の 100BASE-TX 用 CAT5 以上のストレートケーブルが使用可能です。 Ethernet ボードとの誤接続に注意して下さい。
パネル	パネルは、PC 本体の筐体と接触します。PC 本体の筐体は PC 取扱説明書に従って F.G. へ接続してください。基板上の R33 パットおよび R43 パットをショートさせることにより、CN2 および CN3 の金属シェルおよび 8 番端子をパネルへ接続することができます。（図 1-2、図 1-3 参照）

図 1-1　パネル概観

CN2、CN3 コネクタ周辺回路を図 1-2 に示します。

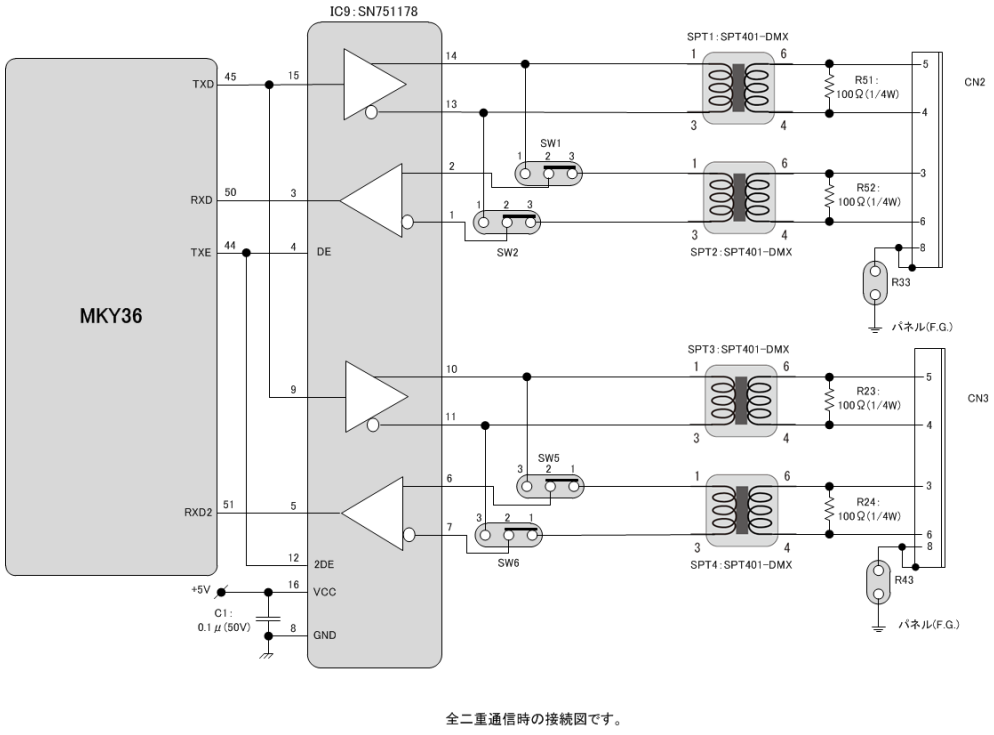


図 1-2　コネクタ周辺回路

1.4 ディップスイッチ

HLSB-36PCI1 のディップスイッチの設定を図 1-3 に示します。

複数の HLSB-36PCI1 を同一の機器に搭載する場合には､SW9 のボード ID を設定してください。このボード ID によって、ソフトウェアから特定の HLSB-36PCI1 を見つけることができます。（工場出荷時設定 ID：0）
HLSB-36PCI1 は、全二重通信、半二重通信の両方に対応します。

全二重通信（工場出荷時設定）
SW1、SW2 を 2-3 に SW5、SW6 を 1-2 にしてください。

半二重通信
SW1、SW2 を 1-2 に SW5、SW6 を 2-3 にしてください。

HLSB-36PCI1 は常にマルチドロップ接続の終端位置（通信ケーブルの端）になります。
その為、ターミネーションは常に有効な状態です。

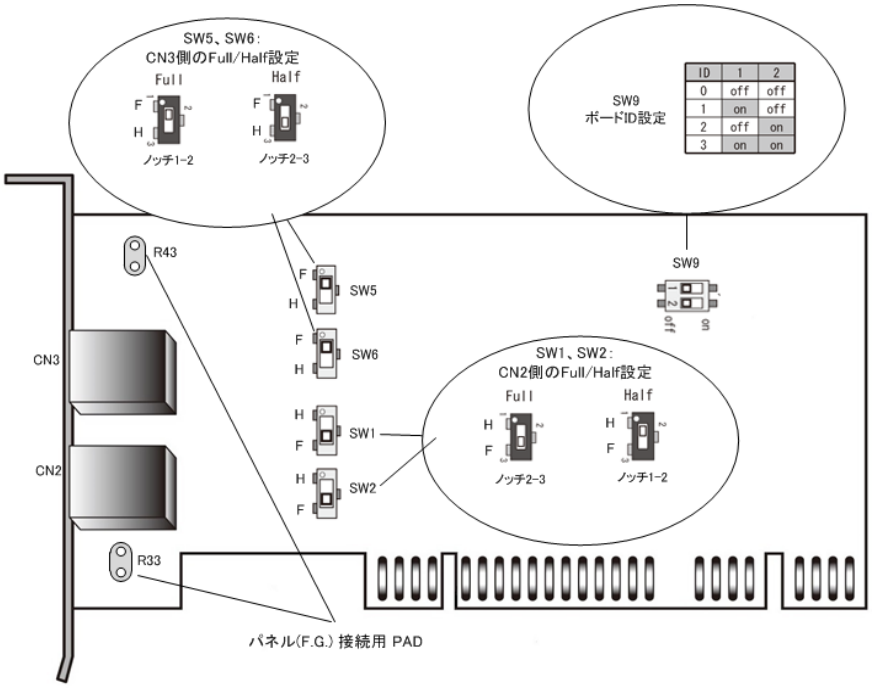


図 1-3　HLSB-36PCI1 ボードの設定

1.5 メモリマップ

HLSB-36PCI1 のメモリマップを表 1-2 に示します。
メモリマップ中のアドレスは HLSB-36PCI1 の先頭アドレスからの相対値であり、実際のアドレスはボードの先頭アドレス値を加算したアドレスになります。

表 1-2　メモリマップ

アドレス	概　要
000H ～ 595H	MKY36
596H ～ EFFH	未使用
F00H	Chip Reset Register
F02H	Board ID Register
F04H ～ FFFH	未使用

1.5.1　MKY36

本ボード上の MKY36 は表 1-2 に示す通り、000H ～ 595H にマッピングしています。
MKY36 のメモリマップについては「MKY36 ユーザーズマニュアル」の「第 2 章 MKY36 のソフトウエア」、「2.1 メモリマップ」をご参照ください。

1.5.2 HLSB-36PCI1 独自のレジスタ

表 1-2 のメモリマップに示された F00H および F02H のレジスタは、HLSB-36PCI1 独自のレジスタです。
以下に、そのレジスタの詳細を記載します。

Chip Reset Register　　アドレス：F00H

bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R/W	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	W
機能	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	CRST0

[機能説明] CRST0（Chip ReSeT 0）へ ”1” をライトすることにより、MKY36 の RST 端子へリセット信号を印加することができます。RST 端子へのリセット期間は、100ms です。また、本レジスタは書き込み専用レジスタの為、読み込みを行った場合のデータは不定です。

Board ID Register　　アドレス：F02H

bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R/W	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
機能	--	--	--	--	--	--	--	--	--	--	--	--	--	--	BID1	BID0

[機能説明] BID0,1（Board ID）ビットをリードすることにより、SW9 によって設定されたボード ID の値を取得することができます。詳細については、”1.4 ディップスイッチ ” を参照ください。

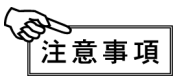
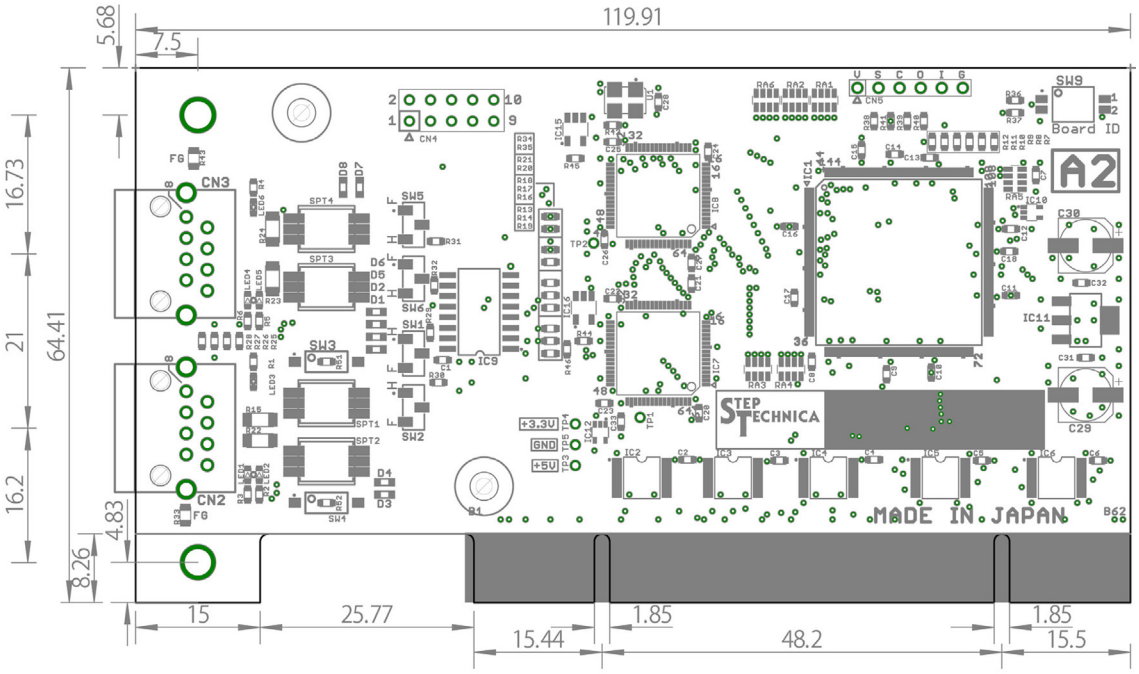


表 1-2 のメモリマップに示されている未使用領域（"596H ～ EFFH" 、"F04H ～ FFFH"）はアクセスしないでください。システムを不安定にする可能性があります。

1.6 寸法図


119.91
5.68
7.5
16.73
21
64.41
16.2
4.83
8.26
15
25.77
15.44
1.85
48.2
1.85
15.5
CN3
CN2
CN1
CN4
CN5
SW9
Board ID
A2
MADE IN JAPAN
+3.3V
GND
+5V

第２章　ソフトウェア

本章は、ステップテクニカ社提供の API について記述します。

2.1 概要

Windows 上のユーザアプリケーションからの HLSB-36PCI1 へのアクセスを簡略化するために DLL を用意しています。
下記ステップテクニカ社のダウンロードページより DLL をダウンロードできます。

URL：http://www.steptechnica.com/jp/download/index.html

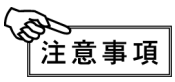
対応 OS は
・Windows10（64bit/32bit）
・Windows8.1（64bit/32bit）
・Windows 8 （64bit/32bit）
・Windows 7 （64bit/32bit）

になります。
提供している DLL は、Microsoft Visual Studio や VB6 などから呼び出すことが可能となっています。
以下から DLL に収録されている API 関数の説明を記します。

対応 OS や、最新のソフトウェア情報は、弊社 web サイト（http://www.steptechnica.com）をご確認ください。

2.2 著作権・免責

ステップテクニカ社が提供している、全てのドキュメント・プログラム・プログラムソースの著作権は、株式会社ステップテクニカが所有しています。株式会社ステップテクニカは、以下の注意事項を了承された個人・法人、または、その他の団体が弊社製品 HLSB-36PCI1 を利用する場合に限り、これら著作物の複製・利用をする権利をライセンスするものであり、株式会社ステップテクニカに断り無く、これら著作物の一部または全部を改訂・再配布を行ったり、上記以外の目的のために複製・利用することはできません。



① 弊社 web ページより入手した全てのソフトウェアの使用による、いかなる結果に対しても弊社は一切責任を負いません。
② API 関数は、説明に従って正しくお使いください。
③ 仕様・内容は、将来予告無く変更になる場合があります。弊社は、将来への互換性について、一切保証いたしません。
④ 弊社製品以外の OS や開発環境等に関するお問い合わせはサポートいたしかねます。
⑤ バグ・不具合などを発見された方は、弊社技術部までご連絡ください。

2.3 ファイル構成

“DLL” フォルダに収められたファイルは以下のとおりです。

【hlsb36pci1.dll】

DLL 本体です。Windows のシステムフォルダか、本 DLL を使用するユーザプログラムと同じディレクトリにコピーしてお使いください。

【hlsb36pci1.lib】

Microsoft Visual C++ 用のインポートライブラリです。VisualStudio2013 によって作成されています。

【hlsb36pci1.h】

API のヘッダファイルです。新規に HLS マスターボードを使用する場合は、こちらのヘッダファイルを使用してください。HLSB-36PCI-LP/EXP からの乗り換えユーザ用に hlsb36pcilp.h も用意しています。

【hlsb36pcilp.h】

HLSB-36PCI-LP/EXP 用互換ヘッダファイルです。ご使用の際には、Windows.h により後ろにインクルードしてください。

2.4 制限事項

ここでは、本 API を使用してアプリケーションを作成する際の制限事項について記します。

2.4.1 マルチスレッド

DLL 内の API は複数スレッドから同時に使用することはできません。
アプリケーションをマルチスレッド構成にする場合、同時呼び出しが起こらないように配慮して下さい。

2.4.2 省電力モードについて

HLSB-36PCI1 は、省電力モード機能に対応しておりません。
OS のスリープ機能を停止したうえでご使用ください。スリープに入った場合には、搭載されている MKY36 への電源供給が遮断され、通信が停止します。
また、省電力モードからの復帰時には、リセットがかかる為、各レジスタは初期化され、コントロール、Do、Di、C1 ～ C7、DRC 領域は不定の状態になりますのでご注意ください。

2.4.3 割り込み処理

MKY36 では、INT0R と INT1R レジスタにより割込み有効・無効の設定と割込み発生状況の確認が行えます。
ドライバ内部では、割込み発生時の INT0R、INT1R の下位 8bit の情報を保持するレジスタ（割込み発生要因レジスタ）と INT0R、INT1R それぞれで割込みが発生した回数を保持するレジスタ（割込み発生回数レジスタ）を管理しています。
ドライバ内部では、割込み発生時にこれらのレジスタを使用して次の処理を行います。
（ここでは、INT0 での割込み発生した場合の説明を記します。）

① 割込み発生要因レジスタに割込み発生要因情報をセットします。
（ユーザアプリケーションから割込み発生要因レジスタのクリア指示があるまで過去の割込み発生要因が残った状態でセットされます。）

② 割込み発生回数レジスタの値をインクリメントする。

③ INT0R の 0 から 7bit の内で ”1” となっている箇所に ”1” をライトし、割込み発生要因のクリアを行います。

割込み発生要因レジスタと割込み発生回数レジスタから情報の取得とクリアを行う為の API 関数を用意しています。

(1) 割込み発生回数レジスタの値を返す関数（HlsbGetInt0Counter, HlsbGetInt1Counter）
ドライバ内で MKY36 からの INT0、INT1 それぞれの割込み発生回数を割込み発生回数レジスタでカウントしており、そのカウント値を返します。

(2) 割込み発生回数レジスタのクリア関数（HlsbClearInt0Counter, HlsbClearInt1Counter）
割込み発生回数レジスタをクリアします。

(3) 割込み発生要因レジスタの値を返す関数（HlsbGetInt0StatusInfo, HlsbGetInt1StatusInfo）
ドライバ内部で MKY36 からの INT0、INT1 それぞれで割込みが発生された際に割込み要因を割込み発生要因レジスタで保持しており、その割込み発生要因レジスタの情報を返します。

(4) 割込み発生要因レジスタのクリア関数（HlsbClearInt0StatusInfo, HlsbClearInt1StatusInfo）
割込み発生要因レジスタをクリアします。
ユーザアプリケーションでは、これらの関数を使用して MKY36 からの割込み発生回数と割込み発生要因の確認を行ってください。

2.4.4 ドライバを使用しない場合のアクセス方法

弊社提供ドライバを使用せずにせずに HLSB-36PCI1 へ直接アクセスする場合には、以下の点について注意が必要です。

HLSB-36PCI1 には、常に 32bit アクセスを行ってください。その時、下位 16bit データが有効となり、上位 16bit は使用されません。その為にアクセスするアドレスは上記メモリマップの 2 倍を指定する必要があります。
例えば、MKY36 のアドレス 200H を Read する場合、HLSB-36PCI1 の 400H を 32bit Read することで、その下位 16bit に MKY36 の 200H の 2 バイトデータが取得できます。HLSB-36PCI1 独自レジスタに関しても同様のアクセスが必要です。

2.5 API 関数詳細説明

表 2-1 にサポートしている API 関数の一覧を示します。
hlsb36pcilp.h 内に収録されている API 関数の内容については、HLSB-36PCI-LP または HLSB-36PCIEXP のユーザーズマニュアルを参照してください。
以降に説明する API 関数は、hlsb36pci1.h に収録されている関数です。

表 2-1　API 関数

関　数	機能概要
HlsbGetVersion	API のバージョン取得
HlsbGetLastError	API 関数の終了状態取得
HlsbSearchBoard	HLSB-36PCI1 の枚数と各ボード ID 取得
HlsbOpenHandle	HLSB-36PCI1 へのハンドル・オープン
HlsbCloseHandle	HLSB-36PCI1 へのハンドル・クローズ
HlsbReadWord	HLSB-36PCI1 へのリード・アクセス
HlsbWriteWord	HLSB-36PCI1 へのライト・アクセス
HlsbReadData	HLSB-36PCI1 への指定ワード長リード・アクセス
HlsbWriteData	HLSB-36PCI1 への指定ワード長リード・アクセス
HlsbResetBoard	HLSB-36PCI1 に搭載 MKY36 へのリセット指示
HlsbGetInt0Counter HlsbGetInt1Counter	ドライバ内部で保持している INT0、1 割込み発生回数取得
HlsbClearInt0Counter HlsbClearInt1Counter	ドライバ内部で保持している INT0、1 割込み発生回数クリア
HlsbGetInt0StatusInfo HlsbGetInt1StatusInfo	ドライバ内部で保持している INT0、1 割込み要因情報取得
HlsbClearInt0StatusInfo HlsbClearInt1StatusInfo	ドライバ内部で保持している INT0、1 割込み要因情報クリア

2.5.1 HlsbGetVersion

書式

UINT HlsbGetVersion(void);

機能

API のバージョン番号を返します。

引数

なし

戻り値

API のバージョンを表す符号なしの整数値

エラーコード

本関数実行後に HlsbGetLastError が返すエラーコード

HLSB_SUCCESS	正常終了

[補足]

必須ではありませんが、hlsb36pci1.dll を利用するユーザアプリケーションで、DLL に対する互換性チェックを行うことで安全性を高めることができます。
ここでいう安全性とは、互換性の無い関数コールをあらかじめ避けることでプログラムの強制終了などを避けることを意味します。
本 API が返すバージョン番号は表 2-2 のような構成になっています。それぞれの番号がアップされる原因は以下の通りです。

メジャー番号：API の仕様変更など、互換性を保てなかった変更がなされた時にかわります。
マイナー番号：API の追加など、下位互換を保ったままの変更がなされた時にかわります。
アップデート番号：バグ修正など、仕様上の変更が全くない変更がなされた時にかわります。

マイナー番号やアップデート番号の数字は互換上無視してもかまいませんがメジャー番号の値が変わっている場合は、API をコールしないことをお奨めします。この互換性チェックは、初期化処理よりも先に行う必要があります。

表 2-2 バージョン番号の構成

戻り値（例）	メジャー番号（ビット 15 ～ 8）	マイナー番号（ビット 7 ～ 4）	アップデート番号（ビット 3 ～ 0）
0x0102	1	0	2
0x1398	13	9	8

2.5.2 HlsbGetLastError

書式

UINT HlsbGetLastError(void);

機能

プロセスが最後に呼び出した API 関数の終了状態をエラーコードとして返します。

引数

なし

戻り値

本 API がサポートしているエラーコードを表 2-3 に示します。
表で説明しているエラーコードは hlsb36pci1.h にて定義されています。

表 2-3　エラーコード一覧

記号定数	値	説　明
HLSB_SUCCESS	0	正常終了
HLSB_ERR_DEVICE_NOT_EXIST	1	デバイスが存在しない
HLSB_ERR_ALREADY_OPENED	2	すでにオープンされている
HLSB_ERR_CLOSED	3	HlsbOpenHandle() が一度もコールされていない
HLSB_ERR_INVALID_HANDLE	4	ハンドル値が無効
HLSB_ERR_INVALID_PARAM	5	無効なパラメータでコールされた
HLSB_ERR_NO_RESOUCE	6	実行に必要なリソースが足りない
HLSB_ERR_FAILED	7	原因不明により処理が遂行されなかった
HLSB_NOT_CALLYET	99	まだ 1 度も API がコールされていない

2.5.3 HlsbSearchBoard

書式

BOOL HlsbSearchBoard(BYTE *board_num , BYTE *board_id_list);

機能

PC 上に存在する HLSB-36PCI1 ボードの枚数とボード ID リストを返す。
同一 PC 上に 5 枚以上の枚数を認識できません。

引数

*board_num	ボード枚数がセットされる変数へのポインタ セットされた値の意味は以下の通りです。 ・0 ： 1 枚もない ・1 ～ 4 ： 認識したボード枚数 ・-1 ： 5 枚以上確認された
*board_id_list	unsigned char 型の要素数 4 つの配列へのアドレスまたは NULL を指定 NULL が指定された場合は、ボード枚数のみを数えます。 セットされた値の意味は以下の通りです。 ・0 ～ 3 ： ボード ID 情報 ・0xFF ： 認識できなかった

戻り値

正常終了時は TRUE、エラー発生時は FALSE を返します。
詳しいエラー発生要因は、HlsbGetLastError を実行することにより確認ができます。

エラーコード

HLSB_SUCCESS	正常終了
HLSB_ERR_INVALID_PARAM	*board_num に NULL が指定された
HLSB_ERR_FAILED	原因不明により処理が実行できなった

[補足]

HLSB-36PCI1 には、SW9 にてボード ID を指定できます。ボード ID を設定することにより PC 上に
複数の HLSB-36PCI1 が搭載されている場合に個別に識別することが可能です。
本 API では、最大 4 台までの HLSB-36PCI1 を識別します。API 使用時には、

```
BYTE board_num;
BYTE board_id_list[4];
HlsbSearchBoard(&board_num, &board_id_list[0]);
```

と構造体の配列を宣言してパラメータとしてセットして下さい。
例として、PC 上に 3 枚の HLSB-36PCI1 が存在しており、各 HLSB-36PCI1 のボード ID が
1 枚目ボード ID=0、2 枚目ボード ID=1、3 枚目ボード ID=2
Windows 上で認識されている順番が 1、3，2 となっている状況で HlsbSearchBoard が実行された場合は

board_num = 3;
board_id_list [0] = 0、board_id_list [1] = 2、board_id_list [2] = 1、board_id_list [3] = 0xFF

と返します。

2.5.4 HlsbOpenHandle

書式

HANDLE HlsbOpenHandle(int index_no)

機能

HLSB-36PCI1 へのハンドル値を取得

引数

index_no	インデックス番号 HLSB-36PCI1 が複数枚あるときは、2 枚目以降を選択できます。 HLSB-36PCI1 が 1 枚しかないときは、0 を設定してください。 詳しくは " 補足 " を参照してください。

戻り値

指定されたインデックス番号へハンドル値を返し、処理に失敗した場合は -1 を返します。
詳しいエラー発生要因は、HlsbGetLastError を実行することにより確認ができます。

エラーコード

HLSB_SUCCESS	正常終了
HLSB_ERR_DEVICE_NOT_EXIST	デバイスが存在しない
HLSB_ERR_FAILED	原因不明により処理が実行できなった

[補足]

HLSB-36PCI1 ボードが 1 枚のみ場合は、HlsbSearchBoard を実行せずに

```
HlsbOpenHandle (0);
```

としても問題ありません。
HLSB-36PCI1 が複数枚存在する場合は、"HlsbSearchBoard" を先に実行し、操作を行う HLSB-36PCI1 を確認しておく必要があります。
例として、PC 上に 3 枚の HLSB-36PCI1 が存在しており、それぞれのボード ID が

1 枚目ボード ID=0、2 枚目ボード ID=1、3 枚目ボード ID=2

と設定されています。アプリケーションでボード ID=2 のハンドル値を取得する場合は

```
BYTE board_num;
BYTE board_id_list[4];
HlsbSearchBoard(&board_num, &board_id_list[0]);
```

と実行し、board_id_list[4] に 2(操作したいボード ID) がセットされている事を確認します。

board_id_list [0] = 0、board_id_list [1] = 2、board_id_list [2] = 1、board_id_list [3] = 0xFF

"board_id_list[1]" に操作するボード ID がセットされているので
HlsbOpenHandle のパラメータには、ボード ID2 が保持している配列のインデックス番号 "1" をセットします。

```
HlsbOpenHandle(1);
```

プログラム終了時、HlsbCloseHandle によりハンドルをクローズしてください。

2.5.5 HlsbCloseHandle

書式

BOOL HlsbCloseHandle(HANDLE hlsbHandle);

機能

HlsbOpenHandle で取得したハンドルをクローズ

引数

hlsbHandle	HLSB-36PCI1 へのハンドル値

戻り値

正常終了時は TRUE、エラー発生時は FALSE を返します。
詳しいエラー発生要因は、HlsbGetLastError を実行することにより確認ができます。

エラーコード

HLSB_SUCCESS	正常終了
HLSB_ERR_INVALID_ HANDLE	無効な hlsbHandle が指定された
HLSB_ERR_FAILED	原因不明により処理が実行できなった

2.5.6 HlsbReadWord

書式

BOOL HlsbReadWord(HANDLE hlsbHandle、const ULONG addr、WORD *data)

機能

HLSB-36PCI1 のメモリマップ内の指定アドレスからワードデータを読み込む

引数

hlsbHandle	HLSB-36PCI1 へのハンドル値
addr	ボード先頭からのオフセットアドレス 入力条件は以下の通りです。 ・2 の倍数のアドレス値を指定 ・入力範囲は 0x0000 ～ 0x0FFE
*data	読み込んだ値を格納するワード領域へのポインタ

戻り値

正常終了時は TRUE、エラー発生時は FALSE を返します。
詳しいエラー発生要因は、HlsbGetLastError を実行することにより確認ができます。

エラーコード

HLSB_SUCCESS	正常終了
HLSB_ERR_INVALID_ HANDLE	無効な hlsbHandle が指定された
HLSB_ERR_INVALID_PARAM	addr が範囲外 2 の倍数値ではない *data に NULL が指定された
HLSB_ERR_FAILED	原因不明により処理が実行できなった

2.5.7 HlsbWriteWord

書式

BOOL HlsbWriteWord(HANDLE hlsbHandle、const ULONG addr、const WORD data)

機能

HLSB-36PCI1 のメモリマップ内の指定アドレスへワードデータを書き込む

引数

hlsbHandle	HLSB-36PCI1 へのハンドル値
addr	ボード先頭からのオフセットアドレス 入力条件は以下の通りです。 ・2 の倍数のアドレス値を指定 ・入力範囲は 0x0000 ～ 0x0FFE
data	書き込みデータ

戻り値

正常終了時は TRUE、エラー発生時は FALSE を返します。
詳しいエラー発生要因は、HlsbGetLastError を実行することにより確認ができます。

エラーコード

HLSB_SUCCESS	正常終了
HLSB_ERR_INVALID_ HANDLE	無効な hlsbHandle が指定された
HLSB_ERR_INVALID_PARAM	addr が範囲外 2 の倍数値ではない *data に NULL が指定された
HLSB_ERR_FAILED	原因不明により処理が実行できなった

2.5.8 HlsbReadData

書式

BOOL HlsbReadData(HANDLE hlsbHandle、const ULONG addr、const ULONG wordLen、void *data);

機能

HLSB-36PCI1 の指定アドレスから指定ワード長のデータ読み込み

引数

hlsbHandle	HLSB-36PCI1 へのハンドル値
addr	ボード先頭からのオフセットアドレス 入力条件は以下の通りです。 ・2 の倍数のアドレス値を指定 ・入力範囲は 0x0000 ～ 0x0FFE
wordLen	ワード長を指定。入力条件は以下の通り ・入力範囲は 0x0001 ～ 0x0800 ・アクセス範囲が 0x1000 を超えない
*data	読み込みデータ格納先へのアドレス

戻り値

正常終了時は TRUE、エラー発生時は FALSE を返します。
詳しいエラー発生要因は、HlsbGetLastError を実行することにより確認ができます。

エラーコード

HLSB_SUCCESS	正常終了
HLSB_ERR_INVALID_ HANDLE	無効な hlsbHandle が指定された
HLSB_ERR_INVALID_PARAM	addr が範囲外 2 の倍数値ではない wordLen が範囲外 アクセス範囲が 0x1000 を超えている *data に NULL が指定された
HLSB_ERR_FAILED	原因不明により処理が実行できなった

2.5.9 HlsbWriteData

書式

BOOL HlsbWriteData(HANDLE hlsbHandle、const ULONG addr、const ULONG wordLen、void *data);

機能

HLSB-36PCI1 の指定アドレスから指定ワード長のデータ書き込み

引数

hlsbHandle	HLSB-36PCI1 へのハンドル値
addr	ボード先頭からのオフセットアドレス 入力条件は以下の通りです。 ・2 の倍数のアドレス値を指定 ・入力範囲は 0x0000 ～ 0x0FFE
wordLen	ワード長を指定。入力条件は以下の通り ・入力範囲は 0x0001 ～ 0x0800 ・アクセス範囲が 0x1000 を超えない
*data	読み込みデータ格納先へのアドレス

戻り値

正常終了時は TRUE、エラー発生時は FALSE を返します。
詳しいエラー発生要因は、HlsbGetLastError を実行することにより確認ができます。

エラーコード

HLSB_SUCCESS	正常終了
HLSB_ERR_INVALID_ HANDLE	無効な hlsbHandle が指定された
HLSB_ERR_INVALID_PARAM	addr が範囲外 2 の倍数値ではない wordLen が範囲外 アクセス範囲が 0x1000 を超えている *data に NULL が指定された
HLSB_ERR_FAILED	原因不明により処理が実行できなった

2.5.10 HlsbResetBoard

書式

BOOL HlsbResetBoard(HANDLE hlsbHandle);

機能

MKY36 をリセット

引数

hlsbHandle	HLSB-36PCI1 へのハンドル値

戻り値

処理結果を返します。正常終了時は TRUE、エラー発生時は FALSE を返します。
詳しいエラー発生要因は、HlsbGetLastError を実行することにより確認ができます。

エラーコード

HLSB_SUCCESS	正常終了
HLSB_ERR_INVALID_ HANDLE	無効な hlsbHandle が指定された
HLSB_ERR_FAILED	原因不明により処理が実行できなった

[補足]

リセット後、100ms 以上待ってから MKY36 へのアクセスを行ってください。

2.5.11 HlsbGetInt0Counter、HlsbGetInt1Counter

書式

BOOL HlsbGetInt0Counter(HANDLE hlsbHandle, BYTE *int0Counter)
BOOL HlsbGetInt1Counter(HANDLE hlsbHandle, BYTE *int1Counter)

機能

ドライバ内部で保持している INT0、1 割込み発生回数レジスタ情報を取得。
割込み発生回数は、0 から 255(0xFF) までインクリメントし、0 に戻ります。

引数

hlsbHandle	HLSB-36PCI1 へのハンドル値
*int0Counter,*int1Counter	取得した割込み発生回数を格納するバイト領域へのポインタ

戻り値

正常終了時は TRUE、エラー発生時は FALSE を返します。
詳しいエラー発生要因は、HlsbGetLastError を実行することにより確認ができます。

エラーコード

HLSB_SUCCESS	正常終了
HLSB_ERR_INVALID_ HANDLE	無効な hlsbHandle が指定された
HLSB_ERR_INVALID_PARAM	*int0Counter、*int1Counter に NULL が指定された
HLSB_ERR_FAILED	原因不明により処理が実行できなった

2.5.12 HlsbClearInt0Counter、HlsbClearInt1Counter

書式

BOOL HlsbClearInt0Counter(HANDLE hlsbHandle)
BOOL HlsbClearInt1Counter(HANDLE hlsbHandle)

機能

ドライバ内部で保持している INT0、1 割込み発生回数レジスタをクリア

引数

hlsbHandle	HLSB-36PCI1 へのハンドル値

戻り値

正常終了時は TRUE、エラー発生時は FALSE を返します。
詳しいエラー発生要因は、HlsbGetLastError を実行することにより確認ができます。

エラーコード

HLSB_SUCCESS	正常終了
HLSB_ERR_INVALID_ HANDLE	無効な hlsbHandle が指定された
HLSB_ERR_FAILED	原因不明により処理が実行できなった

2.5.13 HlsbGetInt0StatusInfo、HlsbGetInt1StatusInfo

書式

BOOL HlsbGetInt0StatusInfo(HANDLE hlsbHandle, BYTE *int0Info)
BOOL HlsbGetInt1StatusInfo(HANDLE hlsbHandle, BYTE *int1Info)

機能

ドライバ内部で保持している累積された INT0、1 割込み発生要因情報を取得

引数

hlsbHandle	HLSB-36PCI1 へのハンドル値
*int0Info、*int1Info	取得した割込み発生要因情報を格納するバイト領域へのポインタ。 過去に発生した割込み要因が累積されます。

戻り値

正常終了時は TRUE、エラー発生時は FALSE を返します。
詳しいエラー発生要因は、HlsbGetLastError を実行することにより確認ができます。

エラーコード

HLSB_SUCCESS	正常終了
HLSB_ERR_INVALID_ HANDLE	無効な hlsbHandle が指定された
HLSB_ERR_INVALID_PARAM	*int0Info、*int1Info に NULL が指定された
HLSB_ERR_FAILED	原因不明により処理が実行できなった

[補足]

int0Info、int1Info へセットされるパラメータの構成を表 2-4 に記します。
割込みが発生した場合、発生要因に対応した bit が "1" となります。
割込み発生要因の配置は、MKY36 の INT0R、INT1R の下位 8bit と同等です。

表 2-4 Int0Info、Int1Info の内部構成

bit	割込み発生要因
7	スキャンの停止による割込み発生
6	CHECK-2 の発生による割込み発生
5	CHECK-1 の発生による割込み発生
4	サテライト IC から DREQ が新規発生したことによる割込み発生
3	1 周期のスキャン終了時期において割込み発生
2	DR2 機能による割込み発生
1	DR1 機能による割込み発生
0	DR0 機能による割込み発生

2.5.14 HlsbClearInt0StatusInfo、HlsbClearInt1StatusInfo

書式

BOOL HlsbClearInt0StatusInfo(HANDLE hlsbHandle)
BOOL HlsbClearInt1StatusInfo(HANDLE hlsbHandle)

機能

ドライバ内部で保持している累積された INT0、1 割込み発生要因をクリア

引数

hlsbHandle	HLSB-36PCI1 へのハンドル値

戻り値

正常終了時は TRUE、エラー発生時は FALSE を返します。
詳しいエラー発生要因は、HlsbGetLastError を実行することにより確認ができます。

エラーコード

HLSB_SUCCESS	正常終了
HLSB_ERR_INVALID_ HANDLE	無効な hlsbHandle が指定された
HLSB_ERR_FAILED	原因不明により処理が実行できなった

2.6 サンプルプログラム

2.6.1 MKY36 へのアクセスサンプル

本 API を使用しての MKY36 への初期化、HLS 通信設定、Do 変更、Di 情報の取得のサンプルプログラムを記します。

```
int main(int argc, char *argv[])
{
	HANDLE hlsbHandle;
	WORD   sa1_di, sa63_di;
	int    i;

	/* HLSB-36PCI1 用のハンドル生成 */
	hlsbHandle = HlsbOpenHandle(0);
	/* 生成されたハンドルをチェック */
	if (hlsbHandle == INVALID_HANDLE_VALUE) {
		exit(1);   /* FALSE : end of program*/
	}

	/* MKY36 を初期化 */
	// (1) MKY36 メモリマップ内の 0x000 ～ 0x57F を 0x00 でライト
	for (i=0;i<0x580;i+=2) {
		HlsbWriteWord(hlsbHandle, i, 0);
	}

	// (2) BCR へ HLS のスキャン稼働条件を設定
	//    FH=0( ハーフデュプレックス )、BPS1,0=2(6Mbps) と設定します。
	HlsbWriteWord(hlsbHandle, 0x58E, 0x0002);

	// (3) 必要があれば Do 領域 (0x80 ～ 0xFF) へ Do 出力状態 ( 初期値 ) をライト
	//    サンプルプログラムでは特に初期値を指定せずに進めます。

	/* SCR へ FS(Final Satelite) を書き込む */
	HlsbWriteWord(hlsbHandle, 0x580, 0x003F);

	/* SA1 の Di 情報を取得 */
	HlsbReadWord(hlsbHandle, 0x0102, &sa1_di);

	/* SA63 の Di 情報を取得 */
	HlsbReadWord(hlsbHandle, 0x017E, &sa63_di);

	/* SA1 の Do 情報を変更 */
	HlsbWriteWord(hlsbHandle, 0x0082, 0xFF00);

	/* SA63 の Do 情報を変更 */
	HlsbWriteWord(hlsbHandle, 0x00FE, 0x00FF);

	/* 生成したハンドルを閉じる */
	HlsbCloseHandle(hlsbHandle);

	return 0;
}
```

2.6.2 割込み処理サンプル

本 API を使用しての MKY36 からの割込み確認方法のサンプルを記します。

```
int main(int argc, char *argv[])
{
	HANDLE hlsbHandle;
	BYTE int0_current_numOfOccurr;		// 現在の INT0 割込み発生回数
	BYTE int0_lastTime_numOfOccurr;		// 前回の INT0 割込み発生回数
	BYTE int0_factor;			// INT0 割込み発生要因

	/* HLSB-36PCI1 用のハンドル生成 */
	hlsbHandle = HlsbOpenHandle(0);

	/* 生成されたハンドルをチェック */
	if (hlsbHandle == INVALID_HANDLE_VALUE) {
		exit(1);   /* FALSE : end of program*/
	}

	/* SCR へ FS(Final Satelite) を書き込む */
	HlsbWriteWord(hlsbHandle, 0x580, 0x003F);

	/* 割込み発生要因レジスタをクリア */
	HlsbClearInt0StatusInfo(hlsbHandle);

	/* 割込み発生回数レジスタをクリア */
	HlsbClearInt0Counter(hlsbHandle);

	/* 割込み発生回数は 0 です */
	int0_lastTime_numOfOccurr = 0;

	/* 割込み発生要因をセット CHECK-1 発生時に INT0 割込みを発生させます */
	HlsbWriteWord(hlsbHandle, 0x586, 0x2000);

	while (1) {
		/* 割込み発生回数レジスタの情報を取得 */
		HlsbGetInt0Counter(hlsbHandle, &int0_current_numOfOccurr);

		/* 前回の割込み発生回数と比較し一致しなければ割込みが発生しています */
		if (int0_lastTime_numOfOccurr != int0_current_numOfOccurr) {
			/* 現在値を前回値にコピー */
			int0_lastTime_numOfOccurr = int0_current_numOfOccurr;
			/* 割込み発生要因レジスタの情報を取得 */
			HlsbGetInt0StatusInfo(hlsbHandle, &int0_factor);

			/* 割込み発生要因が CHECK-1 であることを確認する */
			if ((int0_factor & 0x0020) == 0x0020) {
				/* --- CHECK-1 が発生したときの処理を書きます --- */

				/* INT0 割込み発生要因レジスタをクリア */
				HlsbClearInt0StatusInfo(hlsbHandle);
			}
		}
	}
	/* 生成したハンドルを閉じます */
	HlsbCloseHandle(hlsbHandle);
	return 0;
}
```

■開発・製造
株式会社ステップテクニカ
〒 358-0011 埼玉県入間市下藤沢 757-3
TEL: 04-2964-8804
http://www.steptechnica.com/
info@steptechnica.com

HLS（MKY36）搭載 PCI ボード
HLSB-36PCI1
ユーザーズマニュアル

ドキュメント No.：STD_HLSB36PCI1_V2.0J
発行年月日：2018 年 11 月