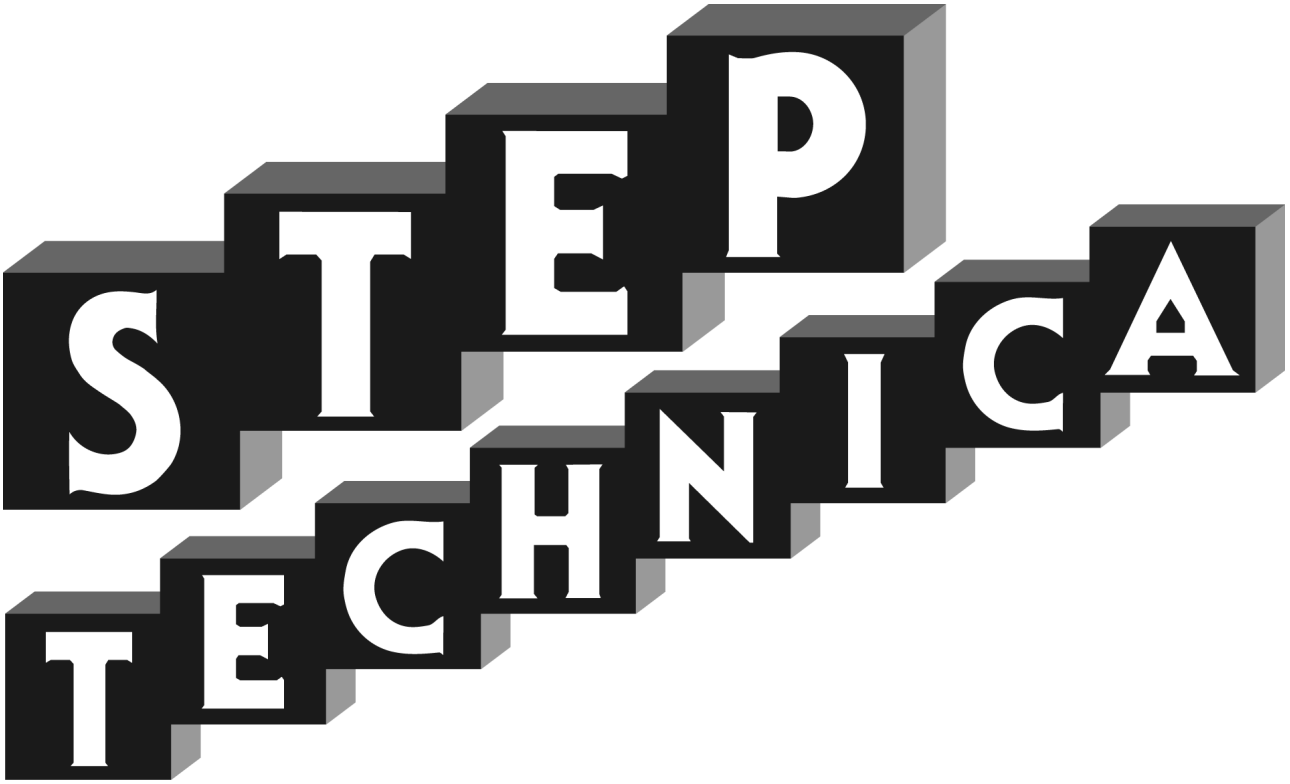
STD-HLSB36PCIEXP_V3.0J

STEP TECHNICA CO.,LTD.



HLS（MKY36）搭載 PCI Express ボード

HLSB-36PCIEXP

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.

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 Express ボードの HLSB-36PCIEXP について記述します。
HLSB-36PCIEXP 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-36PCIEXP 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
・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	2012 年 11 月	-	初版発行
Ver2.0J	2016 年 12 月	--	Windows7/8/8.1 に対応
		3-3	「3.4.3 割込み処理」の追加
		3-4	「表 3-1 API 関数」に割込み処理用の関数を追加
		3-13	割込み発生回数の取得・クリア関数を追加
		3-14	割込み発生要因情報の取得・クリア関数を追加
		3-17	割込み処理サンプルを追加
Ver3.0J	2018 年 11 月	1-1	「表 1-1 仕様」の構成を変更
		3-1	「3.1 概要」での対応 OS について変更

目　次

第１章　製品概要

第２章　ハードウェア

第３章　ソフトウェア

図　目　次

表　目　次

第 1 章　製品概要

本章は、HLSB-36PCIEXP の製品概要について記述します。

1.1 特徴

HLSB-36PCIEXP は、MKY36 搭載した PCI Express 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 API と併せて利用することにより、MKY36 の機能を簡単に利用することが可能です。

1.2 仕様

HLSB-36PCIEXP の仕様を、表 1-1 に示します。

表 1-1　仕様

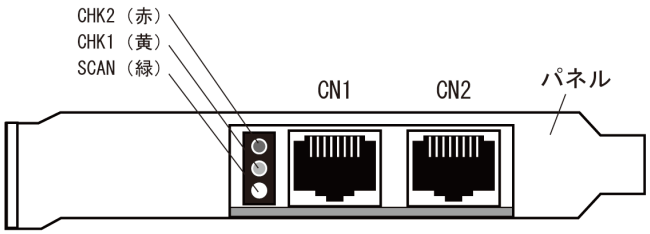
HLS デバイス	MKY36 × 1 個
HLS 通信方式	全二重 / 半二重通信
HLS 通信速度	12M/6M/3Mbps（MKY36 レジスタにて設定）
HLS 通信コネクタ	RJ45 タイプ（8pin モジュラー）x 2 個
対応バス	PCI Express x1 Gen1 準拠した拡張バス
占有リソース	16KB の連続したメモリエリア（PnP にて自動割当）
割り込み	1 ライン使用（PnP にて自動割当）
対応 OS	Windows10　（64bit/32bit） Windows8.1　（64bit/32bit） Windows8　（64bit/32bit） Windows7　（64bit/32bit）
電源	DC +3.3V
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) × 68.9mm(D) ※パネル部含まず（Low Profile 対応）
付属品	Low Profile 用ブラケット

第２章　ハードウェア

本章は、HLSB-36PCIEXP のハードウェアについて記述します。

2.1 コネクタ仕様

HLSB-36PCIEXP のパネル面とその詳細を図 2-1 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
CHK1	MKY36 の #CHK1L 端子に接続した LED（黄） CHECK-1 発生時に点灯
CHK2	MKY36 の #CHK2L 端子に接続した LED（赤） CHECK-2 発生時に点灯
CN1,CN2	HLS 通信ライン 市販の 100BASE-TX 用 CAT-5 以上のストレートケーブルが使用可能
パネル	パネルは、PC 本体の筐体と接触 PC 本体の筐体は、PC の取扱説明書に従った F.G. へ接続 基板上の PAD1 および PAD2 をショートすることにより、CN1 および CN2 の金属シェルおよび 8 番端子を、パネルへ接続可能（図 2-2、図 2-3 参照）

図 2-1　パネル外観

CN1、CN2 コネクタ周辺回路を図 2-2 に示します。

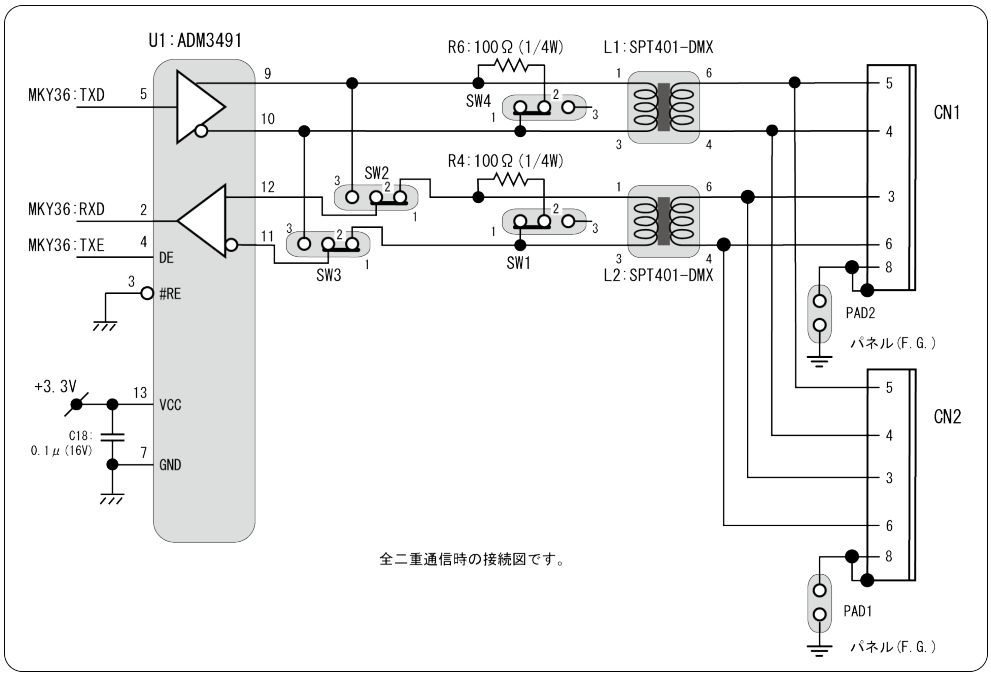


図 2-2　コネクタ周辺回路

2.2 ディップスイッチ

HLSB-36PCIEXP のディップスイッチの設定を図 2-3 に示します。

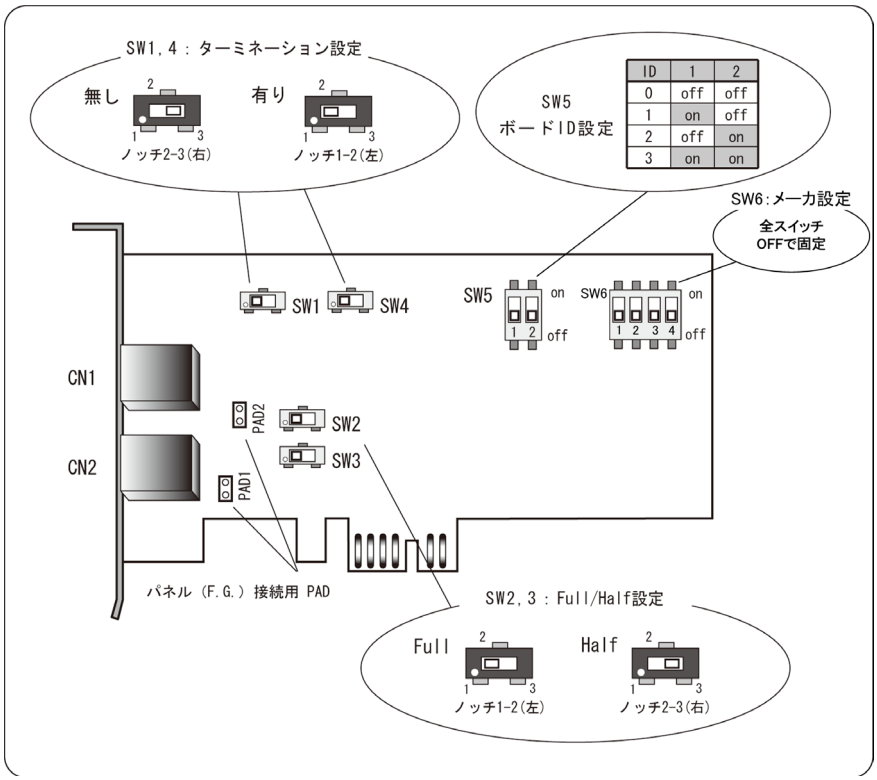


図 2-3　HLSB-36PCIEXP の設定

2.2.1　ボード ID スイッチ（SW5）

複数の HLSB-36PCIEXP を同一の機器に搭載する場合には、SW5 のボード ID を各ボード個別の番号に設定してください。
（工場出荷時設定 ボード ID：0）

2.2.2　Full/Half 設定スイッチ（SW2、SW3）

HLS の通信方式（全二重 / 半二重）を設定します。SW2 と SW3 は、必ず同じ設定にしてください。
（工場出荷時設定 通信方式：全二重通信）

2.2.3　ターミネーション設定スイッチ（SW1、SW4）

HLSB-36PCIEXP が HLS 通信ラインの末端になる場合は、ターミネーションを ON（有効）にしてください。
ターミネーションを ON に設定すると通信経路が 100 Ω で終端されます。
それ以外の場合は、OFF（無効）にしてください。OFF に設定すると終端されません。
SW1 と SW4 は、必ず同じ設定にしてください。（工場出荷時設定 ターミネーション：有効）

2.2.4　メーカ設定スイッチ（SW6）

メーカ用設定スイッチです。すべて OFF の状態で使用してください。

2.3 メモリマップ

HLSB-36PCIEXP のメモリマップを表 2-1 に示します。
メモリマップ中のアドレスは HLSB-36PCIEXP の先頭アドレスからの相対値であり、実際のアドレスは PCI BIOS から自動的に割り振られた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 2-1　メモリマップ

アドレス	概　要
000H ～ 595H	MKY36
596H ～ BFFH	未使用
C00H	Chip Reset Register
C02H ～ DFFH	未使用
E00H	Board ID Register
E02H ～ FFFH	未使用

2.3.1　MKY36

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。

2.3.2 HLSB-36PCIEXP 独自のレジスタ

表 2-1 のメモリマップに示された C00H および E00H のレジスタは、HLSB-36PCIEXP 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：C00H

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、280ns 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：E00H

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、SW5 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、"2.2 ディップスイッチ " を参照ください。

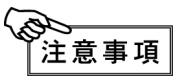
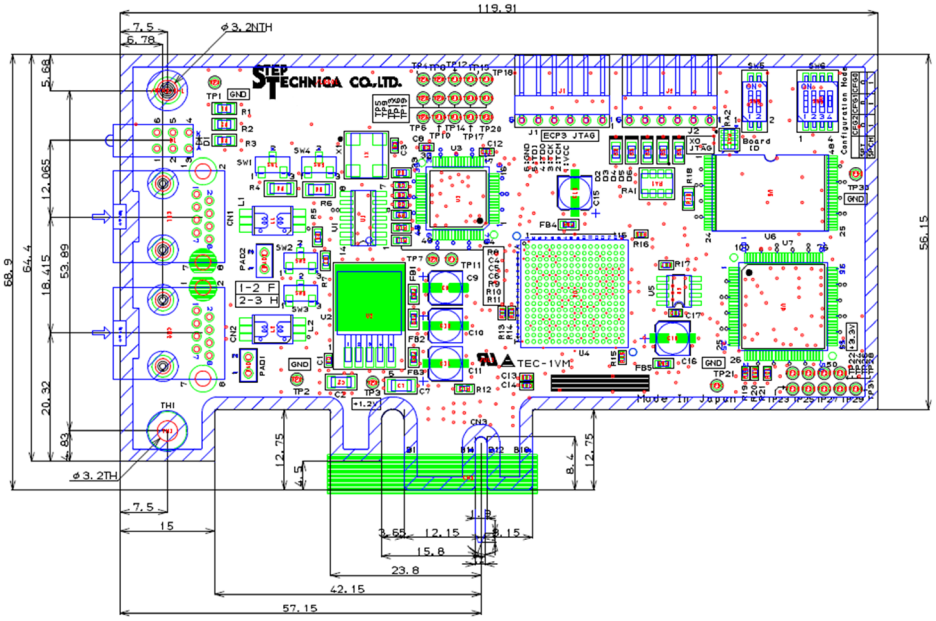


表 2-1 のメモリマップに示されている未使用領域（"596H ～ BFFH、C02H ～ DFFH、E02H ～ FFFH" ）はアクセスしないでください。システムを不安定にする可能性があります。

2.4 寸法図


119.91
7.5
6.78
φ3.2NTH
5.68
12.065
18.415
20.32
6.83
68.9
64.4
53.89
56.15
12.75
4.5
8.4
12.75
7.5
15
2.55
12.15
15.8
23.8
42.15
57.15
STEP TECHNICA CO.,LTD.
TEC-1VMF
Made in Japan

第３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。
本マニュアルは、API バージョンのメジャー番号「２」以上に基づいて記述しています。
製品のご使用にあたっては、弊社ホームページにより最新の情報をご確認してください。

3.1 概要

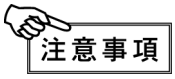
Windows 上のユーザアプリケーションからの HLSB-36PCIEXP へのアクセスするための API を用意しています。
下記ステップテクニカ社のダウンロードページより API をダウンロードできます。

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）

です。
提供している API は、Microsoft Visual Studio や VB6 などから呼び出すことが可能です。

3.2 著作権・免責



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、一切保証致しません。
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致しかねます。
⑤ バグ・不具合などを発見された方は、弊社システム開発部までご連絡ください。

3.3 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。

【hlsb36pciexp.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フォルダにコピーしてお使いください。

【hlsb36pciexp.lib】
Microsoft Visual C/C++ 用のインポートライブラリです。

【hlsb36pciexp.h】
API のヘッダファイルです。ご使用の際は、Windows.h より後ろにインクルードしてください。

3.4 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。

3.4.1 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。

3.4.2 省電力モードについて

HLSB-36PCIEXP 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。

3.4.3 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。
ドライバ内部には、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）があります。
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、MKY36 の割込みを解除します。

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（HlsGetInterrupt0Count, HlsGetInterrupt1Count）
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（HlsClearInterrupt0Count, HlsClearInterrupt1Count）
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（HlsGetInterrupt0StatusInfo, HlsGetInterrupt1StatusInfo）
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（HlsClearInterrupt0StatusInfo, HlsClearInterrupt1StatusInfo）
指定した割込み発生要因を割込み発生要因レジスタからクリアします。

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。

3.4.4 ドライバを使用しない場合のアクセス方法

ステップテクニカ社製ドライバを使用せずに HLSB-36PCIEXP 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-36PCIEXP 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-36PCIEXP 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-36PCIEXP 独自レジスタに関しても同様のアクセスが必要です。

3.5 API 関数詳細説明

表 3-1 にサポートしている API 関数の一覧を示します。
以降に説明する API 関数は、hlsb36pciexp.h に収録されている関数です。

表 3-1　API 関数

関　数	機能概要
HlsGetVersion	API のバージョン番号を取得
HlsGetLastError	API 関数の終了状態を取得
HlsOpenHandle	HLSB-36PCIEXP のハンドルを取得
HlsCloseHandle	HlsOpenHandle で取得したハンドルを閉じる
HlsCountDevice	HLSB-36PCIEXP ボードの枚数を取得
HlsSearchBoard	HLSB-36PCIEXP ボードの枚数とそのボード ID 取得
HlsResetBoard	MKY36 のリセットを指示
HlsBoardID	ボード ID を取得
HlsReadByte	指定したアドレスから 1 バイトのデータ読み込み
HlsWriteByte	指定したアドレスへ 1 バイトのデータ書き込み
HlsReadWord	指定したアドレスから 2 バイトのデータ読み込み
HlsWriteWord	指定したアドレスへ 2 バイトのデータ書き込み
HlsGetInterrupt0Count HlsGetInterrupt1Count	ドライバ内部で保持している INT0、1 割込み発生回数取得
HlsClearInterrupt0Count HlsClearInterrupt1Count	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
HlsGetInterrupt0StatusInfo HlsGetInterrupt1StatusInfo	ドライバ内部で保持している INT0、1 割込み要因情報取得
HlsClearInterrupt0StatusInfo HlsClearInterrupt1StatusInfo	指定した割込み発生要因をドライバ内部で保持している INT0、1 割込み要因情報からクリア

3.5.1 HlsGetVersion

書式

UINT HlsGetVersion(void);

機能

API のバージョン番号を取得します。

パラメータ

なし

リターンパラメータ

API のバージョン番号（BCD コード 16 進数）
（メジャー番号 + マイナー番号 + アップデート番号）

エラーコード

本関数実行後に HlsGetLastError が返すエラーコードとエラー発生要因は以下の通りです。
HLS_SUCCESS 正常終了

注釈

表 3-2 に、バージョン番号の構成を示します。バージョン番号が変更される原因は、下記のとおりです。

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変わります。
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変わります。
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変わります。

表 3-2 バージョン番号の構成

リターンパラメータ（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

3.5.2 HlsGetLastError

書式

UINT HlsGetLastError(void);

機能

最後に呼び出された API 関数の終了状態を取得します。

パラメータ

なし

リターンパラメータ

hlsb36pciexp.h で定義しているエラーコードを返します。

注釈

表 3-3 に hlsb36pciexp.h で定義しているエラーコードを記します。

表 3-3 エラーコードリスト

記号定数	値	機能概要
HLS_SUCCESS	0	正常終了
HLS_ERR_DEVICENOTEXIST	1	デバイスが存在しない
HLS_ERR_ALREADYOPENED	2	すでにオープンされている
HLS_ERR_CLOSED	3	HlsOpenHandle() が一度もコールされていない
HLS_ERR_INVALIDPARAM	4	無効なパラメータでコールされた
HLS_ERR_NORESOUCE	5	実行に必要なリソースが足りない
HLS_ERR_FAILED	6	原因不明により処理が実行できなかった
HLS_NOTCALLYET	99	まだ 1 度も API がコールされていない

3.5.3 HlsSearchBoard

書式

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

機能

パソコンに装着されている HLSB-36PCIEXP ボードの枚数とボード ID リストを返す。

パラメータ

*board_num	ボード枚数がセットされる変数へのアドレスを指定 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	ボード ID を受け取る為に、バイト型を 4 要素持つ配列のアドレスを指定します。または NULL を指定することも可能です。 NULL が指定された場合は、ボード枚数のみを返します。 セットされた値の意味は以下の通りです。 ・0x00 ～ 0x03 ： ボード ID ・0xFF ： 認識できなかった

リターンパラメータ

正常終了時は TRUE、失敗時は FALSE を返します。

エラーコード

本関数実行後に HlsGetLastError が返すエラーコードとエラー発生要因は以下の通りです。

HLS_SUCCESS	正常終了
HLS_ERR_INVALIDPARAM	*board_num に NULL が指定された
HLS_ERR_FAILED	原因不明により処理が実行できなかった

注釈

ボード ID は、SW5 により設定します。
HLSB-36PCIEXP ボードが複数枚装着している場合は、ボード ID により識別することができます。
本 API 関数では、最大 4 枚までの HLSB-36PCIEXP ボードを識別できます。
下記のようにバイト型配列をパラメータとして指定してください

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

動作例として、パソコンに 3 枚の HLSB-36PCIEXP ボードが装着されており、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。
パソコンが認識した順番が 1 枚目、3 枚目、2 枚目となっている状況で HlsSearchBoard が実行された場合

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。

3.5.4 HlsCountDevice

書式

INT HlsCountDevice(void);

機能

パソコンに装着している HLSB-36PCIEXP ボードの枚数を返します。

パラメータ

なし

リターンパラメータ

HLSB-36PCIEXP ボードの枚数を返します

- -1 ： 5 枚以上
- 0 ： 1 枚も接続されていない
- 1 ～ 4 ： 1 枚～ 4 枚

エラーコード

本関数実行後に HlsGetLastError が返すエラーコードとエラー発生要因は以下の通りです。

HLS_SUCCESS	正常終了

注釈

パソコンに 5 枚以上装着することはできません。

3.5.5 HlsBoardID

書式

INT HlsBoardID(HANDLE HLSBHandle);

機能

HLSB-36PCIEXP ボードのボード ID を取得します。

パラメータ

HANDLE HLSBHandle	HLSB-36PCIEXP のハンドル

リターンパラメータ

正常終了時はボード ID（0 ～ 3）を返します。失敗時は -1 を返します。

エラーコード

本関数実行後に HlsGetLastError が返すエラーコードとエラー発生要因は以下の通りです。

HLS_SUCCESS	正常終了
HLS_ERR_INVALIDPARAM	ハンドルが無効
HLS_ERR_FAILED	原因不明により処理が実行できなかった

注釈

パソコンに 5 枚以上装着することはできません。

3.5.6 HlsOpenHandle

書式

HANDLE HlsOpenHandle(intindex_no);

機能

HLSB-36PCIEXP へのハンドルを取得します。

パラメータ

index_no	インデックス番号 インデックス番号には、0 ～ 3 が指定できます。 HLSB-36PCIEXP ボード が 1 枚の場合は、0 を設定してください。 詳しくは " 注釈 " を参照してください。

リターンパラメータ

正常終了時は、1 以上の値を返します。失敗時は -1（INVALID_HANDLE_VALUE）を返します。

エラーコード

本関数実行後に HlsGetLastError が返すエラーコードとエラー発生要因は以下の通りです。

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

注釈

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

HLSB-36PCIEXP ボードが 1 枚のみ場合は、HlsSearchBoard を実行する必要がありません。
HLSB-36PCIEXP ボードが複数枚存在する場合は、HlsSearchBoard を先に実行し、
操作を行う対象の HLSB-36PCIEXP 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-36PCIEXP ボードが装着されており、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。ここでボード ID=2 のハンドルを取得する場合は

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

を実行した結果、
board_id_list[0] = 0、board_id_list[1] = 2、board_id_list[2] = 1、board_id_list[3] = 0xFF
になったと仮定します。
この場合、インデックス番号 1 がボード ID=2 であることが確認できます。
つまり HlsOpenHandle のパラメータであるインデックス番号は、1 になります。

3.5.7 HlsCloseHandle

書式

BOOL HlsCloseHandle(HANDLE HLSBHandle);

機能

HlsOpenHandle によって取得したハンドルを閉じます。

パラメータ

HLSBHandle	HLSB-36PCIEXP のハンドル

リターンパラメータ

正常終了時は TRUE、失敗時は FALSE を返します。

エラーコード

本関数実行後に HlsGetLastError が返すエラーコードとエラー発生要因は以下の通りです。

HLS_SUCCESS	正常終了
HLS_ERR_INVALIDPARAM	ハンドルが無効
HLS_ERR_FAILED	原因不明により処理が実行できなった

3.5.8 HlsReadByte

書式

BOOL HlsReadByte(HANDLE HLSBHandle、const ULONG Adr、BYTE *Dat);

機能

指定したアドレスから 1 バイトのデータを読み込みます。

パラメータ

HLSBHandle	HLSB-36PCIEXP のハンドル
Adr	アドレス値 入力条件は以下の通り ・入力範囲：0x0000 ～ 0x0FFF
*Dat	読み込みデータ格納先のアドレス

リターンパラメータ

正常終了時は TRUE、失敗時は FALSE を返します。

エラーコード

本関数実行後に HlsGetLastError が返すエラーコードとエラー発生要因は以下の通りです。

HLS_SUCCESS	正常終了
HLS_ERR_INVALIDPARAM	ハンドルが無効 Adr が範囲外 *Dat に NULL が指定された
HLS_ERR_FAILED	原因不明により処理が実行できなった

3.5.9 HlsWriteByte

書式

BOOL HlsWriteByte(HANDLE HLSBHandle、const ULONG Adr、const BYTE Dat);

機能

指定したアドレスへ 1 バイトのデータを書き込みます。

パラメータ

HLSBHandle	HLSB-36PCIEXP のハンドル
Adr	アドレス値 入力条件は以下の通り ・入力範囲：0x0000 ～ 0x0FFF
Dat	書き込みデータ

エラーコード

本関数実行後に HlsGetLastError が返すエラーコードとエラー発生要因は以下の通りです。

HLS_SUCCESS	正常終了
HLS_ERR_INVALIDPARAM	ハンドルが無効 Adr が範囲外
HLS_ERR_FAILED	原因不明により処理が実行できなった

3.5.10 HlsReadWord

書式

BOOL HlsReadWord(HANDLE HLSBHandle、const ULONG Adr、WORD *Dat);

機能

指定したアドレスから 2 バイトのデータを読み込みます。

パラメータ

HLSBHandle	HLSB-36PCIEXP のハンドル
Adr	アドレス値 入力条件は以下の通り ・2 の倍数 ・入力範囲：0x0000 ～ 0x0FFE
*Dat	読み込みデータ格納先のアドレス

エラーコード

本関数実行後に HlsGetLastError が返すエラーコードとエラー発生要因は以下の通りです。

HLS_SUCCESS	正常終了
HLS_ERR_INVALIDPARAM	ハンドルが無効 Adr が範囲外 Adr が 2 の倍数ではない *Dat に NULL が指定された
HLS_ERR_FAILED	原因不明により処理が実行できなった

3.5.11 HlsWriteWord

書式

BOOL HlsWriteWord(HANDLE HLSBHandle、const ULONG Adr、const WORD Dat);

機能

指定したアドレスへ 2 バイトのデータを書き込みます。

パラメータ

HLSBHandle	HLSB-36PCIEXP のハンドル
Adr	アドレス値 入力条件は以下の通り ・2 の倍数 ・入力範囲：0x0000 ～ 0x0FFE
Dat	書き込みデータ

エラーコード

本関数実行後に HlsGetLastError が返すエラーコードとエラー発生要因は以下の通りです。

HLS_SUCCESS	正常終了
HLS_ERR_INVALIDPARAM	ハンドルが無効 Adr が範囲外 Adr が 2 の倍数ではない
HLS_ERR_FAILED	原因不明により処理が実行できなった

3.5.12 HlsResetBoard

書式

BOOL HlsResetBoard(HANDLE HLSBHandle);

機能

指定された HLSB-36PCIEXP ボードの MKY36 をリセットします。

パラメータ

HLSBHandle	HLSB-36PCIEXP のハンドル

リターンパラメータ

正常終了時は TRUE、失敗時は FALSE を返します。

エラーコード

本関数実行後に HlsGetLastError が返すエラーコードとエラー発生要因は以下の通りです。

HLS_SUCCESS	正常終了
HLS_ERR_INVALIDPARAM	ハンドルが無効
HLS_ERR_FAILED	原因不明により処理が実行できなった

3.5.13 HlsGetInterrupt0Count、HlsGetInterrupt1Count

書式

BOOL HlsGetInterrupt0Count (HANDLE HLSBHandle, BYTE *int0Counter);
BOOL HlsGetInterrupt1Count (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します。
割込み発生回数は、0 から 255（0xFF）までインクリメントします。

パラメータ

HLSBHandle	HLSB-36PCIEXP のハンドル
*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アドレス

リターンパラメータ

正常終了時は TRUE、失敗時は FALSE を返します。

エラーコード

本関数実行後に HlsGetLastError が返すエラーコードとエラー発生要因は以下の通りです。

HLS_SUCCESS	正常終了
HLS_ERR_INVALIDPARAM	ハンドルが無効 *int0Counter、*int1Counter に NULL が指定された
HLS_ERR_FAILED	原因不明により処理が実行できなった

3.5.14 HlsClearInterrupt0Count、HlsClearInterrupt1Count

書式

BOOL HlsClearInterrupt0Count (HANDLE HLSBHandle);
BOOL HlsClearInterrupt1Count (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します。

パラメータ

HLSBHandle	HLSB-36PCIEXP のハンドル

リターンパラメータ

正常終了時は TRUE、失敗時は FALSE を返します。

エラーコード

本関数実行後に HlsGetLastError が返すエラーコードとエラー発生要因は以下の通りです。

HLS_SUCCESS	正常終了
HLS_ERR_INVALIDPARAM	ハンドルが無効
HLS_ERR_FAILED	原因不明により処理が実行できなった

3.5.15 HlsGetInterrupt0StatusInfo、HlsGetInterrupt1StatusInfo

書式

BOOL HlsGetInterrupt0StatusInfo (HANDLE HLSBHandle, BYTE *int0Info);
BOOL HlsGetInterrupt1StatusInfo (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します。

パラメータ

HLSBHandle	HLSB-36PCIEXP のハンドル
*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アドレス 過去に発生した割込み要因が累積されます。

リターンパラメータ

正常終了時は TRUE、失敗時は FALSE を返します。

エラーコード

本関数実行後に HlsGetLastError が返すエラーコードとエラー発生要因は以下の通りです。

HLS_SUCCESS	正常終了
HLS_ERR_INVALIDPARAM	ハンドルが無効 *int0Info、*int1Info に NULL が指定された
HLS_ERR_FAILED	原因不明により処理が実行できなった

注釈

int0Info、int1Info へセットされるパラメータの構成を表 3-4 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 3-4 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

3.5.16 HlsClearInterrupt0StatusInfo、HlsClearInterrupt1StatusInfo

書式

BOOL HlsClearInterrupt0StatusInfo (HANDLE HLSBHandle, BYTE clearInt0Info);
BOOL HlsClearInterrupt1StatusInfo (HANDLE HLSBHandle, BYTE clearInt1Info);

機能

指定した割込み要因をドライバ内部で保持している INT0、INT1 割込み発生要因情報からクリアします。

パラメータ

HLSBHandle	HLSB-36PCIEXP のハンドル
clearInt0Info、clearInt1Info	クリアする割込み発生要因を指定

リターンパラメータ

正常終了時は TRUE、失敗時は FALSE を返します。

エラーコード

本関数実行後に HlsGetLastError が返すエラーコードとエラー発生要因は以下の通りです。

HLS_SUCCESS	正常終了
HLS_ERR_INVALIDPARAM	ハンドルが無効
HLS_ERR_FAILED	原因不明により処理が実行できなった

注釈

割込み発生要因と設定値の構成を表 3-5 に記します。
割込み発生要因に対応した設定値を clearInt0Info、clearInt1Info へセットしてください。
複数の割込み発生要因をクリアする場合は、各設定値の論理和を求めてセットしてください。

表 3-5 クリアする割込み発生要因と設定値

割込み発生要因	設定値
スキャンの停止による割込みをクリア	0x80
CHECK-2 の発生による割込みをクリア	0x40
CHECK-1 の発生による割込みをクリア	0x20
サテライト IC から DREQ が新規発生したことによる割込みをクリア	0x10
1 周期のスキャン終了時期において割込みをクリア	0x08
DR2 機能による割込みをクリア	0x04
DR1 機能による割込みをクリア	0x02
DR0 機能による割込み発生	0x01

3.6 サンプルプログラム

3.6.1 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-36PCIEXP 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) {
		HlsWriteWord(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。
	HlsWriteWord(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 Satellite) を書き込む */
	HlsWriteWord(HLSBHandle, 0x580, 0x003F);

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

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

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

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

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

	return 0;
}
```

3.6.2 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-36PCIEXP 用のハンドル生成 */
	HLSBHandle = HlsOpenHandle(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 Satellite) を書き込む */
	HlsWriteWord(HLSBHandle, 0x580, 0x003F);

	/* 割込み発生要因レジスタをクリア */
	HlsClearInterrupt0StatusInfo(HLSBHandle, 0xFF);

	/* 割込み発生回数レジスタをクリア */
	HlsClearInterrupt0Count(HLSBHandle);

	/* 変数をクリア */
	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 */
	HlsWriteWord(HLSBHandle, 0x586, 0x2000);

	while (1) {
		/* 割込み発生回数レジスタの情報を取得 */
		HlsGetInterrupt0Count(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 */
			HlsGetInterrupt0StatusInfo(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 & 0x20) == 0x20) {
				/* --- CHECK-1 が発生したときの処理を書きます --- */

				/* 割込み発生要因レジスタから割込み発生要因 CHECK-1 をクリア */
				HlsClearInterrupt0StatusInfo(HLSBHandle, 0x20);
			}
		}
	}
	/* 生成したハンドルを閉じます */
	HlsCloseHandle(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 Express ボード
HLSB-36PCIEXP
ユーザーズマニュアル

ドキュメント No.：STD_HLSB36PCIEXP_V3.0J
発行年月日：2018 年 11 月