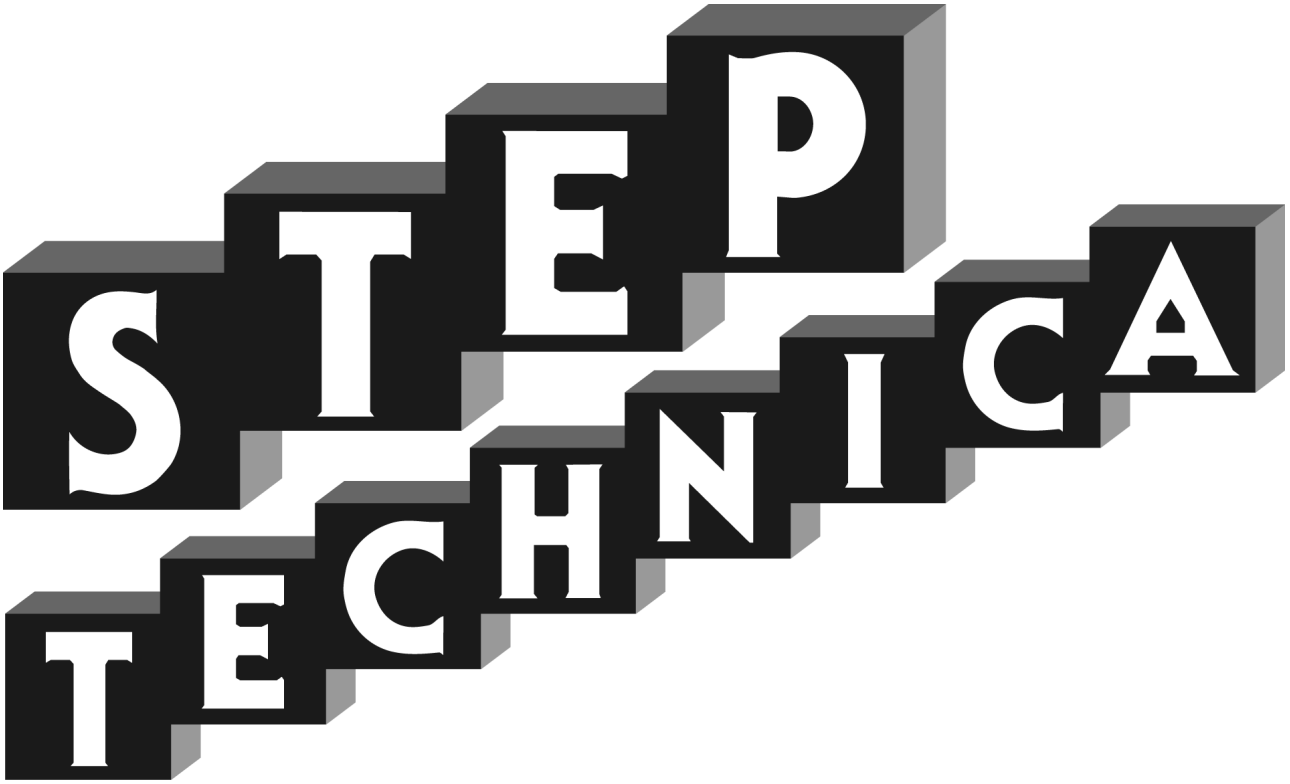
STD_HLSB36PCI1_V1.0E

STEP TECHNICA CO.,LTD.



HLS（MKY36）PCI Board

HLSB-36PCI1

User's Manual

Notes

1. The information in this document is subject to change without prior notice.
   Before using this product, please confirm that this is the latest version of document.
2. Technical information in this document, such as explanations and circuit examples, are just for references to use this product in a proper way.
   When actually using this product, always fully evaluate the entire system according to the design purpose based on considerations of peripheral circuits and environment.
   We assume no responsibility for any incompatibility between this product and your system.
3. We assume no responsibility whatsoever for any losses or damages arising from the use of the information, products, and circuits in this document, or for infringement of patents and any other rights of a third party.
4. When using this product and the information and circuits in this document, we do not guarantee the right to use any property rights, intellectual property rights, and any other rights of a third party.
5. This product is not designed for use in critical applications, such as life support systems.
   Contact us when considering such applications.
6.

Revision history

Date	Version	Content	Note
AUG 2018	1.0E	Issued the first edition	

Preface

This manual describes HLSB-36PCI1, PCI board with MKY36 which is a kind of HLS family IC.
Be sure to read "HLS Introduction Guide" in advance to use HLSB-36PCI1 and understand this manual.

● Target readers
・Those who first build an HLS
・Those who first use StepTechnica's HLSB-36PCI1 to build an HLS

● Prerequisites
This manual assumes that you are familiar with :
・Network technology
・Semiconductor products (especially microcontrollers and memory)

● Related manuals
・HLS Introduction Guide
・HLS Technical Guide (For Network)
・MKY36 User's Manual

【Note】
Some terms in this manual are different from those that used in our website or product brochures. The brochure uses ordinary terms to help many people in various industries understand our products.
Expertise in HLS family, please understand technical information based on technical documents (manuals).

Table of Contents

Chapter 1 Hardware

Chapter 2 Software

Figures

Tables

Chapter 1 Hardware

This chapter describes hardware of HLSB-36PCI1.

1.1 Features

HLSB-36PCI1 is a PCI expansion bus supported HLS communication board with MKY36 chip. This product is designed to help easy operation of MKY36 functions with StepTechnica's Library for Windows.

HLSB-36PCI1 is adopting 8-pin modular connector to evaluate operation with commercial CAT-5 or more greater straight-through cable for 100BASE-TX.
Experience of using HLSB-36PCI1 will be a clue for microcontroller embedded system with MKY36.

1.2 Specifications

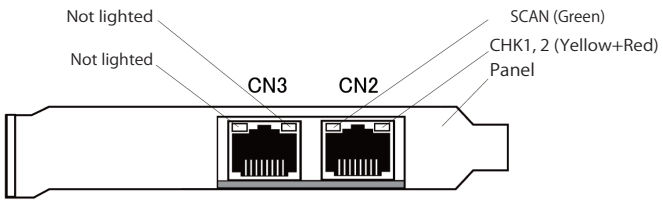
The specifications of HLSB-36PCI1 are shown in Table 1-1.

Table 1-1 Board specifications

Type	HLSB-36PCI1
Type of IC	MKY36 × 1pc
Communication method	HLS communication method（full-duplex, half-duplex）
Baud rate	3Mbps/6Mbps/12Mbps
Supported bus	PCI Ver2.2 supported, 32bit / 33MHz expansion bus 5V / 3.3V supported（Low profile supported）
Owned resource	16KB serial memory area（Automatically allocated by PnP）
Interrupt	1 line used（Automatically allocated by PnP）
Connector	RJ-45 modular connector：TM11R-5M2-88-LP
Power supply	DC +5V
Consumption current	500mA or less
Atmospheric conditions	Temperature 0 to 50 ℃ Humidity 20 to 90%（With no condensation）
Size	119.9mm（W）× 64.4mm（D）※ Not including panel

1.3 Connector specifications

The panel view and its details are shown in Fig.1-1.

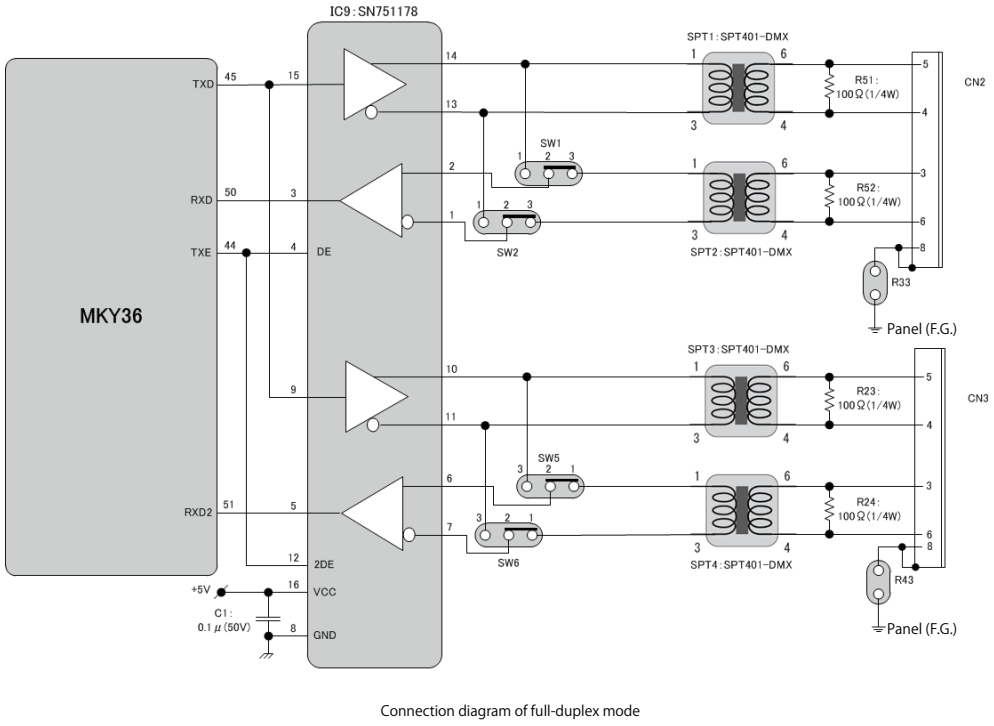


Pin#	Function	
	Full-duplex	Half-duplex
3	RX+	--
4	TX-	TRX-
5	TX+	TRX+
6	RX-	--
8	Shield	Shield
1,2,7	Unused	Unused

SCAN	LED is connected to #SCANL pin of MKY36, lit green when MKY36 is scanning.
CHK1, 2	LED is connected to #CHK1L and #CHK2L pin of MKY36. Lit yellow when CHECK-1 is occurring, lit red when CHECK-2 is occurring. LED lit red if CHECK-1 and CHECK-2 are occurring at the same time.
CN2, CN3	HLS communication line Commercial CAT-5 or more greater straight-through cable for 100BASE-TX can be used. Be sure that this is not for an Ethernet board.
Panel	Panel is connected to a PC case front panel. Connect a PC case to F.G. in accordance with the PC manual. Metal shell of CN2 / CN3, and No.8 pin can be connected to the panel by making R33 PAD and R43 PAD on the board shorted. (Refer to Fig.1-2, Fig.1-3.)

Fig.1-1 Panel view

CN2 / CN3 connector peripheral circuit is shown in Fig.1-2.



Connection diagram of full-duplex mode

Fig.1-2 Connector peripheral circuit

1.4 DIP switches

The settings of DIP switches of HLSB-36PCI1 are shown in Fig.1-3.

If two or more HLSB-36PCI1 devices are connected to one PC, set SW9 board IDs to individual number of each boards so that you can distinguish the boards using software. (Factory setting board ID : 0)
HLSB-36PCI1 supports both of full-duplex and half-duplex mode.

Full-duplex (Factory setting)
Set SW1 and SW2 to 2-3, and SW5 and SW6 to 1-2.

Half-duplex
Set SW1 and SW2 to 1-2, SW5 and SW6 to 2-3.

HLSB-36PCI1 is always at a termination of multi-drop connection (an end of network cable).
For this reason, termination is always being enabled.

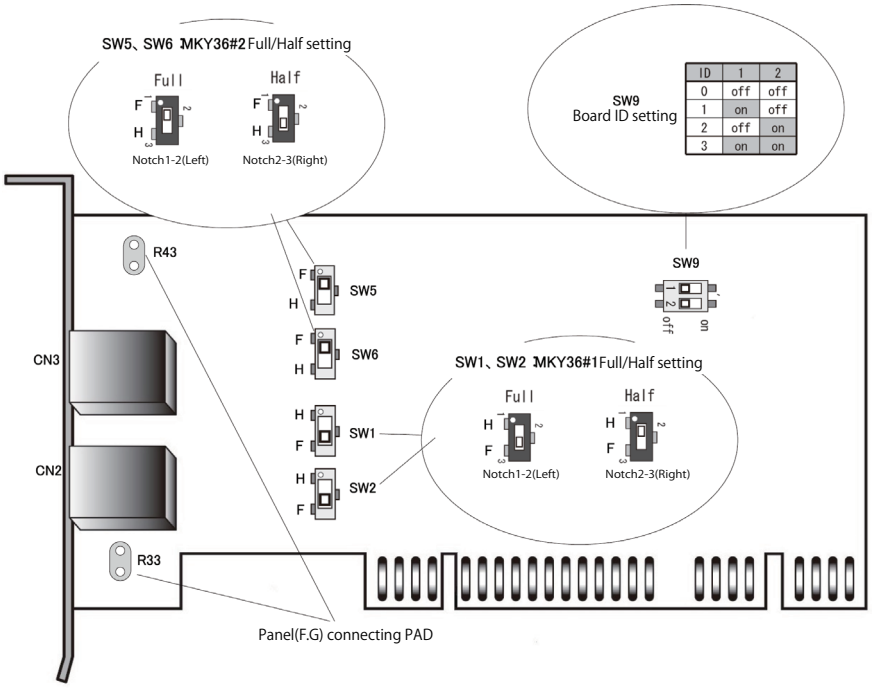


Fig.1-3 Settings of HLSB-36PCI1 board

1.5 Memory map

Table 1-2 describes memory map of HLSB-36PCI1.
An address in memory map is the relative value from starting address of HLSB-36PCI1, and actual address is the value which is added the starting address of the board.

Table 1-2 Memory map

Address	Content
000H ～ 595H	MKY36
596H ～ EFFH	Unused
F00H	Chip Reset Register
F02H	Board ID Register
F04H ～ FFFH	Unused

1.5.1 MKY36

MKY36 on the board is mapped to 000H to 595H as shown in Table 1-2.
For the details of memory map of MKY36, refer to "2.1 Memory map" in MKY36 User's Manual.

1.5.2 Unique register of HLSB-36PCI1

F00H and F02H registers shown in Table 1-2 Memory map are unique registers of HLSB-36PCI1. The details of these registers are described in the following.

Chip Reset Register Address：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
Function	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	CRST0

[Function] By writing "1" to CRST0（Chip ReSet 0）, reset signal can be applied to #RST pin of MKY36.
A reset term to #RST pin is 100ms. This register is write-only, so data will be undefined when reading the register.

Board ID Register Address：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
Function	--	--	--	--	--	--	--	--	--	--	--	--	--	--	BID1	BID0

[Function] The value of board ID set in SW9 can be obtained by reading BID0,1（Board ID）.
For the details, refer to "1.4 DIP switches".

Caution

Do not access to unused area（596H to EFFH）shown in Table 1-2 Memory map.
It can make a whole system unstable.

1.6 The access without driver software

When you directly access to HLSB-36PCI1 without StepTechnica's driver software, note the following point.

Always use 32bit access to access to HLSB-36PCI1. At that time, lower 16 bit data will be used and upper 16 bit will be unused.
For this reason, address to access needs to be specified 2 times longer than memory map mentioned above.
For example, in order to read 200H address of MKY36, make the lower 16bit in 400H of HLSB-36PCI1 obtain 2 bytes data in 200H of MKY36 by executing 32bit Read.
This access method applies to HLSB-36PCI1 unique register.

Chapter 2 Software

This chapter describes API provided by StepTechnica.

2.1 Outline

StepTechnica provides DLL to optimize the access to HLSB-36PCI1 from user application on Windows.
You can download the DLL from our website below.

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

The supported operating systems are as follows.
・Windows 8 （64bit/32bit）
・Windows 7 （64bit/32bit）

Provided DLL can be called from Microsoft Visual Studio and VB6.
Description of API functions recorded in DLL is the following.

For the details of supported OS and the latest software information, refer to our website.
（http://www.steptechnica.com/en/index.html）

2.2 Copyright and disclaimer

① StepTechnica Co., Ltd. assume no responsibility for any results caused by using the attached driver disk or all software downloaded from our website.

② Use API in proper ways with its instructions.

③ All specifications and contents are subject to change without prior notice.
We do not guarantee for forward compatibility.

④ We can not support for an inquiry regarding operating systems and development environment.

⑤ If you have found any errors and failures, contact our R&D department.

2.3 File structure

Files stored in "DLL" folder are the following.

【hlsb36pci1.dll】

DLL body :
Use this within Windows system folder or user program directory using this DLL.

【hlsb36pci1.lib】

Import library for Microsoft Visual C/C++, which is created using VisualStudio2013

【hlsb36pci1.h】

Header file for API :
If you use HLSB-36PCI1 as a new HLS master board, use this header file.
hlsb36pcilp.h is also available for a user who replaced HLSB-36PCI-LP/EXP with this product.

【hlsb36pcilp.h】

Header file for HLSB-36PCI-LP/EXP:
Include this after than Windows.h when using.

2.4 Restrictions

This chapter describes the restrictions at building an application using this API.

2.4.1 Multi-thread

API in DLL can not be used from other threads at the same time.
In the case that an application has multithreading structure, be sure not to be called from other thread at the same time.

2.4.2 Interrupt handling

INT0R and INT1R registers enable MKY36 to set enable/disable of interrupt and to check the status of interrupt occurrence. The internal driver has registers called interrupt factor register which retains the information in lower 8 bit of INT0R and INT1R at interrupt occurrence and interrupt count register which retains the interrupt occurrence count of each INT0R and INT1R.
The internal driver process the following procedure using these registers at interrupt occurrence.
(For instance, below describes when interrupt occurred at INT0.)

① Set the interrupt factor information in interrupt factor register.
(Previous interrupt factor remains until be cleared by interrupt factor register from user application.)
② Increment the value of interrupt count register.
③ Clear interrupt factor by writing "1" to the bit which is set "1" in 0-7 bit of INT0R.

An API function is provided to obtain and clear the information from interrupt factor register and interrupt count register.

(1) A function which returns the value of interrupt count register (HlsbGetInt0Counter, HlsbGetInt1Counter)
The internal driver retains interrupt count of each INT0, INT1 registers from MKY36. This function returns the count value.

(2) Clear function of interrupt count register (HlsbClearInt0Counter, HlsbClearInt1Counter)

Clears the interrupt count register

(3) A function which returns the value of interrupt factor register (HlsbGetInt0StatusInfo, HlsbGetInt1StatusInfo)

The internal driver retains interrupt factor of each INT0, INT1 registers from MKY36.
This function returns the information from interrupt factor register.

(4) Clear function of interrupt factor register (HlsbClearInt0StatusInfo, HlsbClearInt1StatusInfo)
Clears the interrupt factor register
On user application, check the interrupt count from MKY36 and interrupt factor using these functions.

2.5 API functions

The supported API functions are listed in Table 2-1.
For the contents of API functions recorded in hlsb36pcilp.h, refer to HLSB-36PCI-LP or HLSB-36PCIEXP User's manual.
API functions recorded in hlsb36pci1.h are shown below.

Table 2-1 API functions

Function	Description
HlsbGetVersion	Obtains API version number
HlsbGetLastError	Obtains the termination status of API functions
HlsbSearchBoard	Obtains the number of HLSB-36PCI1 board connected to PC and its board ID
HlsbOpenHandle	Obtains the handle value of HLSB-36PCI1
HlsbCloseHandle	Closes the handle obtained by HlsbOpenHandle
HlsbReadWord	Read access to HLSB-36PCI1
HlsbWriteWord	Write access to HLSB-36PCI1
HlsbReadData	Specified word length read access to HLSB-36PCI1
HlsbWriteData	Specified word length write access to HLSB-36PCI1
HlsbResetBoard	Resets MKY36
HlsbGetInt0Counter HlsbGetInt1Counter	Obtains INT0, INT1 interrupt count retained in the internal driver
HlsbClearInt0Counter HlsbClearInt1Counter	Clears INT0, INT1 interrupt count retained in the internal driver
HlsbGetInt0StatusInfo HlsbGetInt1StatusInfo	Obtains INT0, INT1 interrupt factor retained in the internal driver
HlsbClearInt0StatusInfo HlsbClearInt1StatusInfo	Clears specified INT0, INT1 interrupt factor retained in the internal driver

2.5.1 HlsbGetVersion

Format

UINT HlsbGetVersion（void）;

Function

Returns API version number

Parameter

None

Return value

Unsigned integer value which describes the version number of API

Error code

The error code and error factor returned by HlsbGetLastError after executing this function is as follows.

HLSB_SUCCESS	Terminated normally

[Note]

You can ensure the security by checking compatibility between DLL with user application with hlsb36pci1.dll (not necessarily).
In this addendum, the word 'security' means to avoid an unexpected shutdown by checking incompatible function call in advance.
The version number returned by this API is shown in Table 2-2.
The reasons for updating the version number are as follows.

Major Number : The revision with no backward compatibility such as API specification change.
Minor Number : The revision with backward compatibility such as an addition of API function.
Update Number : The revision with no specification change such as bug fixes.

The updates of Minor or Update Number do not influence on compatibility. When the value of Major Number has been changed, it is better not to call API.
This compatibility check needs to be done before initializing.

Table 2-2 Version numbering

Return value (Example)	Major Number (Bit 15 - 8)	Minor Number (Bit 7 - 4)	Update Number (Bit 3 - 0)
0x0102	1	0	2
0x1398	13	9	8

2.5.2 HlsbGetLastError

Format

UINT HlsbGetLastError（void）;

Function

Obtains the termination state of the API called last time

Parameter

None

Return value

The error codes supported by this API is shown in Table 2-3, which are defined in hlsb36pci1.h.

Table 2-3 Error code list

Character constant	Value	Description
HLSB_SUCCESS	0	Terminated normally
HLSB_ERR_DEVICE_NOT_EXIST	1	Device does not exist.
HLSB_ERR_ALREADY_OPENED	2	Handle has already opened.
HLSB_ERR_CLOSED	3	HlsbOpenHandle() has never been called.
HLSB_ERR_INVALID_HANDLE	4	Invalid handle value
HLSB_ERR_INVALID_PARAM	5	Called with invalid parameter
HLSB_ERR_NO_RESOUCE	6	No resource to execute the process
HLSB_ERR_FAILED	7	The process failed due to unknown reason.
HLSB_NOT_CALLYET	99	API function has never been called.

2.5.3 HlsbSearchBoard

Format

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

Function

Returns the number of HLSB-36PCI1 board connected to PC and its board ID list.
No more than five units can be identified in one PC.

Parameter

*board_num	A pointer to the variable in which to set the number of boards The meanings of set values are as follows. ・0 : Not connected ・1 to 4 : Number of boards identified ・-1 : Five or more
*board_id_list	To receive the board ID, specify a address which has an unsigned char array of four elements (bytes). It is also possible to specify NULL. If NULL has been specified, only the number of boards are counted. The meanings of set values are as follows. ・0 to 3 : Board ID information ・0xFF : Not identified

Return value

Succeeded : TRUE is returned. Failed : FALSE is returned.
You can check the details of error factor by executing HlsbGetLastError.

Error code

HLSB_SUCCESS	Terminated normally
HLSB_ERR_INVALID_PARAM	NULL has been specified to *board_num.
HLSB_ERR_FAILED	The process failed due to unknown reason.

[Note]

The board ID can be set using SW9 on HLSB-36PCI1.
If two or more HLSB-36PCI1 are connected to a PC, each unit can be distinguished by board IDs.
This API can distinguish up to four HLSB-36PCI1 boards.
When using API, specify the byte type array as a parameter as shown below.

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

As an example, three HLSB-36PCI1 units are connected to a PC, and each board IDs are set in sequence ;
1st board ID = 0, 2nd board ID = 1, 3rd board ID = 2.
If the units have been identified by the PC in sequence with first, third, and second, and run HlsbSearchBoard, board number and its IDs are returned as follows.

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

2.5.4 HlsbOpenHandle

Format

HANDLE HlsbOpenHandle（int index_no）

Function

Obtains the handle value of HLSB-36PCI1

Parameter

index_no	Index number You can select the second or later if two or more HLSB-36PCI1 are connected to a PC. If just one HLSB-36PCI1 is connected to PC, set 0. For more information, see "Note".

Return value

Succeeded : 1 or greater value is returned to the specified index number.
Failed : -1 is returned.
You can check the details of error factor by executing HlsbGetLastError.

Error code

HLSB_SUCCESS	Terminated normally
HLSB_ERR_DEVICE_NOT_EXIST	Device does not exist.
HLSB_ERR_FAILED	The process failed due to unknown reason.

[Note]

If only one HLSB-36PCI1 board is connected to a PC, you can simply execute HlsbOpenHandle (0); not HlsbSearchBoard.

When two or more HLSB-36PCI1 are connected to a PC, execute "HlsbSearchBoard" in advance to check which HLSB-36PCI1 to manipuate.
As an example, three HLSB-36PCI1 units are connected to a PC, and each board IDs are set in sequence ;
1st board ID = 0, 2nd board ID = 1, 3rd board ID = 2.
To obtain the handle value of Board ID = 2 via user application, operate as follows.

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

Check if the Board ID=2 (the board to manipulate) is set in board_id_list[4].
board_id_list[0] = 0, board_id_list[1] = 2, board_id_list[2] =1, board_id_list[3] = 0xFF

In this case, you see that the board ID to manipulate is set in “board_id_list[1]”.
That means 1 is the index number to set as a parameter of HlsbOpenHandle.

```
HlsbOpenHandle(1);
```

Close the handle by HlsbCloseHandle at finishing a program.

2.5.5 HlsbCloseHandle

Format

BOOL HlsbCloseHandle（HANDLE hlsbHandle）;

Function

Closes handle obtained by HlsbOpenHandle

Parameter

hlsbHandle	Handle value of HLSB-36PCI1

Return value

Succeeded : TRUE is returned.
Failed : FALSE is returned.
You can check the details of error factor by executing HlsbGetLastError.

Error code

HLSB_SUCCESS	Terminated normally
HLSB_ERR_INVALID_HANDLE	Invalid hlsbHandle has been specified.
HLSB_ERR_FAILED	The process failed due to unknown reason.

2.5.6 HlsbReadWord

Format

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

Function

Reads word data from the specified address in memory map of HLSB-36PCI1.

Parameter

hlsbHandle	Handle value of HLSB-36PCI1
addr	Offset address from the starting address Input conditions are the following ・Specify an address value which is multiples of 2 ・Input range : 0x0000 - 0x0FFE
*data	Pointer to the storage word area of read value

Return value

Succeeded : TRUE is returned.
Failed : FALSE is returned.
You can check the details of error factor by executing HlsbGetLastError.

Error code

HLSB_SUCCESS	Terminated normally
HLSB_ERR_INVALID_HANDLE	Invalid hlsbHandle has been specified.
HLSB_ERR_INVALID_PARAM	addr is out of range. Set value is not multiple of 2. NULL has been specified to *data.
HLSB_ERR_FAILED	The process failed due to unknown reason.

2.5.7 HlsbWriteWord

Format

BOOL HlsbWriteWord(HANDLE hlsbHandle, const ULONG addr, const WORD*data)

Function

Writes specified word data from the specified address in memory map of HLSB-36PCI1

Parameter

hlsbHandle	Handle value of HLSB-36PCI1
addr	Offset address from the starting address of the board Input conditions are the following. ・Specify an address value which is multiples of 2 ・Input range : 0x0000 to 0x0FFE
data	Write data

Return value

Succeeded : TRUE is returned. Failed : FALSE is returned.

You can check the details of error factor by executing HlsbGetLastError.

Error code

HLSB_SUCCESS	Terminated normally
HLSB_ERR_INVALID_HANDLE	Invalid hlsbHandle has been specified.
HLSB_ERR_INVALID_PARAM	addr is out of range. Set value is not multiple of 2. NULL has been specified to *data
HLSB_ERR_FAILED	The process failed due to unknown reason.

2.5.8 HlsbReadData

Format

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

Function

Reads specified word length data to the specified address of HLSB-36PCI1

Parameter

hlsbHandle	Handle value of HLSB-36PCI1
addr	Offset address from the starting address of the board Input conditions are the following. ・Specify an address value which is multiples of 2 ・Input range : 0x0000 - 0x0FFE
wordLen	Specified word length : Input conditions are the following. ・Input range : 0x0001 - 0x0800 ・Access range does not exceed 0x1000.
*data	The storage address of read data

Return value

Succeeded : TRUE is returned. Failed : FALSE is returned

You can check the details of error factor by executing HlsbGetLastError.

Error code

HLSB_SUCCESS	Terminated normally
HLSB_ERR_INVALID_HANDLE	Invalid hlsbHandle has been specified.
HLSB_ERR_INVALID_PARAM	addr is out of range. Set value is not multiple of 2. wordLen is out of range. Access range exceeds 0x1000. NULL has been specified to *data.
HLSB_ERR_FAILED	The process failed due to unknown reason.

2.5.9 HlsbWriteData

Format

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

Function

Writes specified word length data from the specified address of HLSB-36PCI1

Parameter

hlsbHandle	Handle value of HLSB-36PCI1
addr	Offset address from the starting address of the board Input conditions are the following. ・Specify an address value which is multiples of 2 ・Input range : 0x0000 to 0x0FFE
wordLen	Specified word length : Input conditions are the following. ・Input range : 0x0001 to 0x0800 ・Access range does not exceed 0x1000.
*data	The storage address of read data

Return value

Succeeded : TRUE is returned. Failed : FALSE is returned

You can check the details of error factor by executing HlsbGetLastError.

Error code

HLSB_SUCCESS	Terminated normally
HLSB_ERR_INVALID_HANDLE	Invalid hlsbHandle has been specified.
HLSB_ERR_INVALID_PARAM	addr is out of range. Set value is not multiple of 2. wordLen is out of range. Access range exceeds 0x1000. NULL has been specified to *data.
HLSB_ERR_FAILED	The process failed due to unknown reason.

2.5.10 HlsbResetBoard

Format

BOOL HlsbResetBoard(HANDLE hlsbHandle);

Function

Resets MKY36.

Parameter

hlsbHandle	Handle value of HLSB-36PCI1

Return value

Succeeded : TRUE is returned. Failed : FALSE is returned

You can check the details of error factor by executing HlsbGetLastError.

Error code

HLSB_SUCCESS	Terminated normally
HLSB_ERR_INVALID_HANDLE	Invalid hlsbHandle has been specified.
HLSB_ERR_FAILED	The process failed due to unknown reason.

[Note]

Wait more than 100ms to access MKY36 after reset.

2.5.11 HlsbGetInt0Counter、HlsbGetInt1Counter

Format

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

Function

Obtains the information of INT0, 1 interrupt count register retained in internal driver.
Interrupt count is incremented from 0 to 255（0xFF）and returns to 0.

Parameter

hlsbHandle	Handle value of HLSB-36PCI1
*int0Counter,*int1Counter	Pointer to the storage byte area of obtained interrupt count

Return value

Succeeded : TRUE is returned. Failed : FALSE is returned
You can check the details of error factor by executing HlsbGetLastError.

Error code

HLSB_SUCCESS	Terminated normally
HLSB_ERR_INVALID_HANDLE	Invalid hlsbHandle has been specified.
HLSB_ERR_INVALID_PARAM	NULL has been specified to *int0Counter, *int1Counter.
HLSB_ERR_FAILED	The process failed due to unknown reason.

2.5.12 HlsbClearInt0Counter、HlsbClearInt1Counter

Format

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

Function

Clears INT0, INT1 interrupt counter register retained in the internal driver

Parameter

hlsbHandle	Handle value of HLSB-36PCI1

Return value

Succeeded : TRUE is returned. Failed : FALSE is returned
You can check the details of error factor by executing HlsbGetLastError.

Error code

HLSB_SUCCESS	Terminated normally
HLSB_ERR_INVALID_HANDLE	Invalid hlsbHandle has been specified.
HLSB_ERR_FAILED	The process failed due to unknown reason.

2.5.13 HlsbGetInt0StatusInfo、HlsbGetInt1StatusInfo

Format

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

Function

Obtains the information of INT0, INT1 interrupt factor retained in th internal driver

Parameter

hlsbHandle	Handle of HLSB-36PCI1
*int0Info、*int1Info	Pointer to the storage byte area of obtained interrupt factor Information of interrupt factors which has been occurred are accumulated.

Return value

Succeeded : TRUE is returned. Failed : FALSE is returned
You can check the details of error factor by executing HlsbGetLastError.

Error code

HLSB_SUCCESS	Terminated normally
HLSB_ERR_INVALID_HANDLE	Invalid hlsbHandle has been specified.
HLSB_ERR_INVALID_PARAM	NULL has been specified to *int0Info、*int1Info.
HLSB_ERR_FAILED	The process failed due to unknown reason.

[Note]

The configuration of parameters set to int0Info, int1Info is shown in Table 2-4.
When interrupt has occurred, the bit which corresponds to its factor turned to "1".
The arrangements of interrupt factors are same as the lower 8bit in INT0R and INT1R of MKY36.

Table 2-4 Internal configuration of int0Info and int1Info

bit	Interrupt factor
7	An interrupt occurs when scanning stops.
6	An interrupt occurs when CHECK-2 occurs.
5	An interrupt occurs when CHECK-1 occurs.
4	An interrupt occurs when DREQ is generated from satellite IC.
3	An interrupt occurs when a period of scan finished.
2	An interrupt occurs due to DR2 function.
1	An interrupt occurs due to DR1 function.
0	An interrupt occurs due to DR0 function.

2.5.14 HlsbClearInt0StatusInfo、HlsbClearInt1StatusInfo

Format

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

Function

Obtains the information of INT0, INT1 interrupt factor retained in the internal driver

Parameter

hlsbHandle	Handle value of HLSB-36PCI1

Return value

Succeeded : TRUE is returned. Failed : FALSE is returned
You can check the details of error factor by executing HlsbGetLastError.

Error code

HLSB_SUCCESS	Terminated normally
HLSB_ERR_INVALID_HANDLE	Invalid hlsbHandle has been specified.
HLSB_ERR_FAILED	The process failed due to unknown reason.

2.6 Sample Program

2.6.1 Access sample to MKY36

The following describes a sample program to initialize, set HLS communication mode, change Do and obtain the information of Di using this API.

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

        /* Generating handle for HLSB-36PCI1 */
        hlsbHandle = HlsbOpenHandle(0);
        /* Checking the generated handle */
        if (hlsbHandle == INVALID_HANDLE_VALUE) {
                exit(1);   /* FALSE : end of program*/
        }

        /* Initializing MKY36 */
        // (1) Write 0x00 to 0x000 - 0x57F in memory map of MKY36.
        for (i=0;i<0x580;i+=2) {
                HlsbWriteWord(hlsbHandle, i, 0);
        }

        // (2) Setting HLS scanning conditions to BCR
        //     Set FH=0(Half-duplex), BPS1,0=2(6Mbps)
        HlsbWriteWord(hlsbHandle, 0x58E, 0x0002);

        // (3) Write Do output status (initial value) to Do area (0x80 - 0xFF) if necessary.
        //     In this sample program, it is processed without specifying the initial value.

        /* Write FS(Final Satelite) to SCR.*/
        HlsbWriteWord(hlsbHandle, 0x580, 0x003F);

        /* Obtain Di information of SA1. */
        HlsbReadWord(hlsbHandle, 0x0102, &sa1_di);

        /* Obtain Di information of SA63. */
        HlsbReadWord(hlsbHandle, 0x017E, &sa63_di);

        /* Change Do information of SA1 */
        HlsbWriteWord(hlsbHandle, 0x0082, 0xFF00);

        /* Change Do information of SA63 */
        HlsbWriteWord(hlsbHandle, 0x00FE, 0x00FF);

        /* Close the generated handle. */
        HlsbCloseHandle(hlsbHandle);

        return 0;
}
```

2.6.2 Sample of interrupt handling

The following describes the sample method to check the interrupt from MKY36 using this API.

```
int main（int argc, char *argv[]）
{
        HANDLE hlsbHandle;
        BYTE int0_current_numOfOccurr;          // Current INT0 interrupt count
        BYTE int0_lastTime_numOfOccurr;         // Previous INT0 interrupt count
        BYTE int0_factor;                       // INT0 interrupt factor

        /* Generating handle for HLSB-36PCI1 */
        hlsbHandle = HlsbOpenHandle（0）;

        /* Checking the generated handle */
        if (hlsbHandle == INVALID_HANDLE_VALUE) {
                exit（1）;          /* FALSE : end of program*/
        }

        /* Write FS(Final Satelite) to SCR */
        HlsbWriteWord（hlsbHandle, 0x580, 0x003F）;

        /* Clear the interrupt factor register */
        HlsbClearInt0StatusInfo（hlsbHandle）;

        /* Clear the interrupt count register */
        HlsbClearInt0Counter（hlsbHandle）;

        /* Interrupt count is 0 */
        int0_lastTime_numOfOccurr = 0;

        /* Set the interrupt factor. Generate INT0 interrupt when CHECK-1 is occurring*/
        HlsbWriteWord（hlsbHandle, 0x586, 0x2000）;

        while（1）{
                /* Obtain the information of interrupt count register */
                HlsbGetInt0Counter（hlsbHandle, &int0_current_numOfOccurr）;

                /* An interrupt is occurring if it does not match with previous interrupt count. */
                if (int0_lastTime_numOfOccurr != int0_current_numOfOccurr) {
                        /* Copy current value to the previous value */
                        int0_lastTime_numOfOccurr = int0_current_numOfOccurr;
                        /* Obtain the information of an interrupt factor register */
                        HlsbGetInt0StatusInfo（hlsbHandle, &int0_factor）;

                        /* Make sure the interrupt factor is CHECK-1.  */
                        if ((int0_factor & 0x0020) == 0x0020) {
                                /* --- Here descrices the process when CHECK-1 has occured.  --- */

                                /* Clear INT0 interrupt factor register */
                                HlsbClearInt0StatusInfo（hlsbHandle）;
                        }
                }
        }
        /*Close the generated handle */
        HlsbCloseHandle（hlsbHandle）;
        return 0;
}
```

■ Developed and manufactured by
StepTechnica Co., Ltd.
757-3, Shimofujisawa, Iruma, Saitama
http://www.steptechnica.com/en/index.html
info@steptechnica.com

HLS（MKY36）PCI Board

HLSB-36PCI1

User's Manual

Document No.：STD_HLSB36PCI1_V1.0E
Issued：August 2018