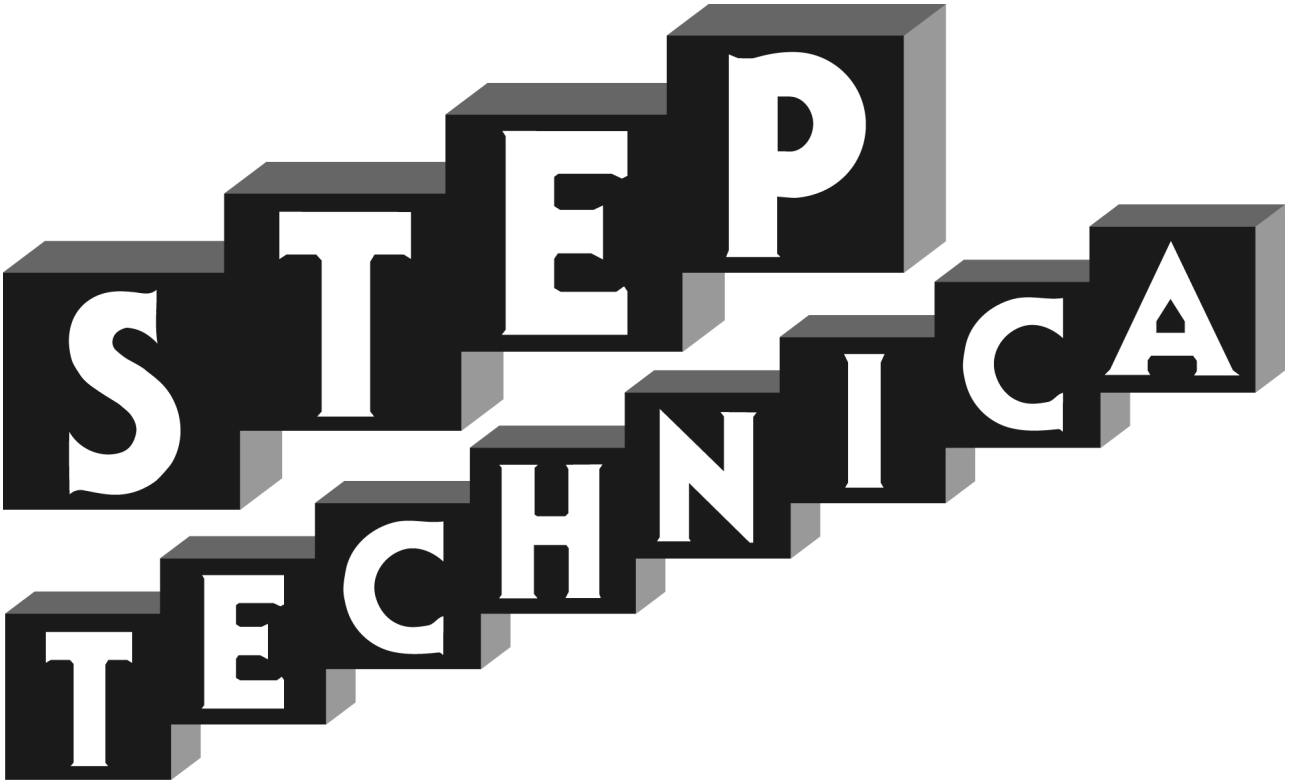
STD-HLSB36PCIEXP_V2.0E

STEP TECHNICA CO.,LTD.



HLS(MKY36) PCI Express Board

HLSB-36PCIEXP

User's Manual

Note

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 this document.
2. Technical information in this document, such as explanations and circuit examples, are references for this product. When actually using this product, always fully evaluate the entire system according to the design purpose based on considerations of peripheral circuits and the PC board 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.

Preface

This manual describes the HLSB-36PCIEXP, PCI Express board which MKY36 is a kind of center IC in the Hi-speed Link System, is mounted.
Be sure to read “Hi-speed Link System Introduction Guide” before understanding this manual and the HLSB-36PCIEXP.
In this manual, the Hi-speed Link System is abbreviated as “HLS”.

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

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

● Related Manuals
・Hi-speed Link System Introduction Guide
・Hi-speed Link System Technical Guide
・Hi-speed Link System Center IC MKY36 User’s Manual

【Caution】
Some terms in this manual are different from those used on our website and in our product brochures. The brochure uses ordinary terms to help many people in various industries understand our products.
Please understand technical information on HLS Family based on technical documents (manuals).

CONTENTS

Chapter 1 Product Summary

Chapter 2 Hardware

Chapter 3 Software

Figure Table of Contents

List of Tables

Chapter 1 Product Summary

This chapter describes the HLSB-36PCIEXP product summary.

1.1 Features

HLSB-36PCIEXP is the HLS communication board. The board which Step Technica's MKY36 IC is mounted, have expansion bus compatible with PCIExpress. HLSB-36PCIEXP is designed for more easy operation of the MKY36 using with the attached API for Windows.

1.2 Specifications

The Specifications of HLSB-36PCIEXP is given in Table 1-1.

Table 1-1 Specifications

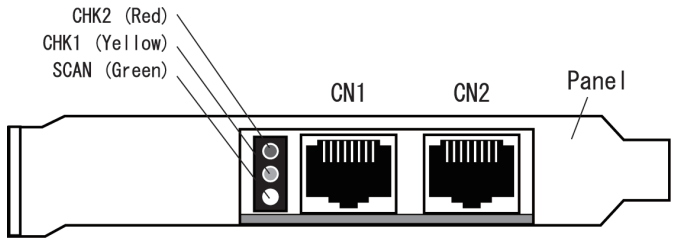
Type	HLSB-36PCIEXP
Type of IC mounted	MKY36 × 1
HLS communication method	Full / half duplex
HLS communication speed	12M/6M/3Mbps(Set by MKY36 register)
Connector	HLS communication connector(RJ-45 type) × 2pcs
Compatible bus	Expansion bus compliant with PCI Express x1 Gen1
Resources to be occupied	16KB, consecutive memory area (automatically allocated by PnP function)
Interrupt	1 line used (automatically allocated by PnP function)
Maximum number of simultaneous use	4
Compliant OS	Windows8.1 (64bit/32bit) Windows8 (64bit/32bit) Windows7 (64bit/32bit)
Power supply	DC +3.3V
Consumption current	Less than 500mA
Conditions of use	Temperature 0 － 50℃ Humidity 20 － 90% (with no condensation)
Board Size	119.9mm(W) x 68.9mm(H) *Panel side of the board is excluded
Accessories	Low profile bracket
Provided software	Driver for Windows API HLSB-36PCI-LP/EXP Editor

Chapter 2 Hardware

This chapter describes the HLSB-36PCIEXP hardware.

2.1 Connector

The panel side of HLSB-36PCIEXP is shown in Fig.2-1.



Pin No.	function	
	full	half
3	RX+	--
4	TX-	TRX-
5	TX+	TRX+
6	RX-	--
8	shield	shield
1, 2, 7	Not used	Not used

SCAN	A green LED connected to #SCANL pin of the MKY36. Led is lighted when occurrence communication.
CHK1	A yellow LED connected to #CHK1L pin of the MKY36. Led is lighted when occurrence CHECK-1.
CHK2	A red LED connected to #CHK2L pin of the MKY36. Led is lighted when occurrence CHECK-2.
CN1,CN2	HLS communication lines. A straight cable for 100BASE-TX, CAT-5 or upper is applicable to these lines.
Panel	The panel part is touching with the PC cabinet. Connect the PC cabinet to FG with the instruction manual of the PC. To connect metal shell and No. 8 pin of CN1 / CN2 to the panel. PAD1 and PAD2 on the board need to be short-circuited. (fig.2-2 , fig.2-3)

Fig.2-1 Panel side view

The peripheral circuit of CN1, CN2 connectors is shown in Fig.2-2.

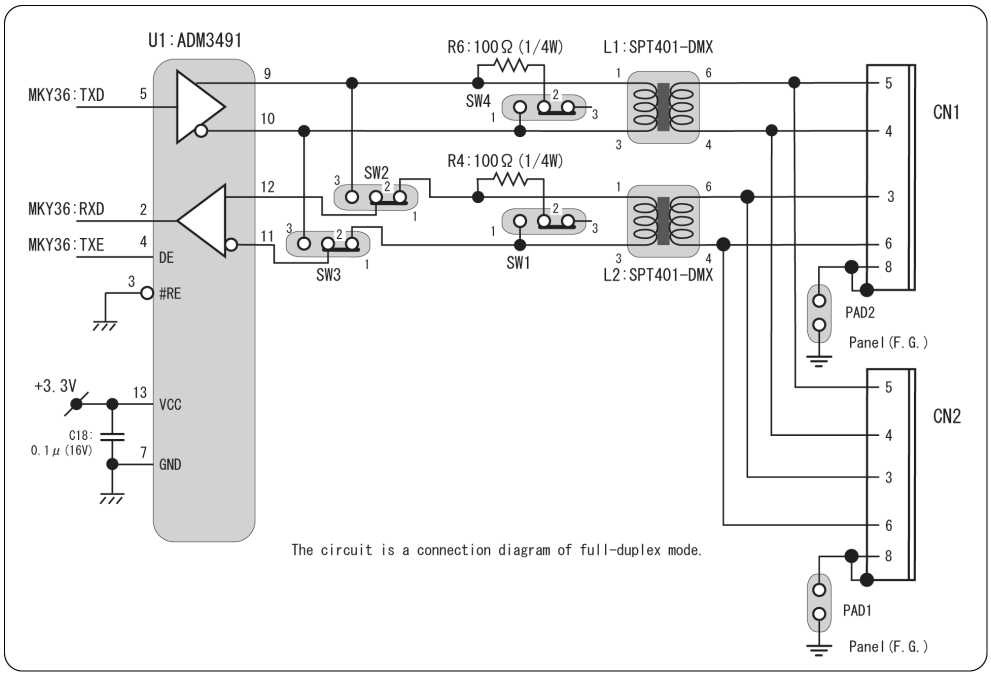


Fig.2-2 Connector peripheral circuit

2.2 DIP switches

The setting dip switches of HLSB-36PCIEXP are shown in Fig.2-3.

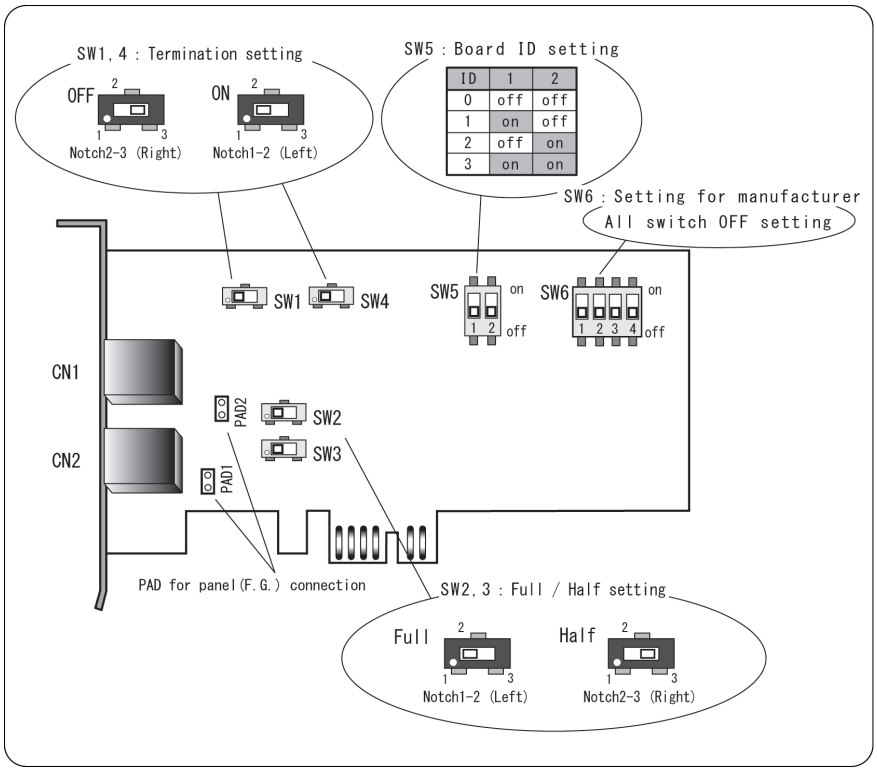


Fig.2-3 Settings of HLSB-36PCIEXP board

2.2.1 Board ID switch (SW5)

To embed multiple HLSB-36PCIEXP boards in one system platform, set SW5 board ID.
(Factory-setting is Board ID : 0)

2.2.2 Full / Half setting switch (SW2 , SW3)

Setting for HLS communication method(Full / Half -duplex). Make sure SW2 and SW3 are same setting.
(Factory-setting is communication method : Full-duplex)

2.2.3 Termination setting switch (SW1 , SW4)

If HLSB-36PCIEXP is at the terminal position of multi-drop connection, set switch “ON” for termination.
When termination is set to ON, the communication path is terminated with 100 Ω .
In all other cases, setting it off (disable). If it is set to OFF, it will not be terminated.
Be sure to set SW1 and SW4 to the same setting. (Factory-setting is termination : ON (enable))

2.2.4 Setting for manufacturer (SW6)

Setting switch for manufacturer. Please use it with all OFF.

2.3 Memory map

HLSB-36PCIEXP memory map are listed in Table 2-1.
Address value in memory map is relative with a starting address of HLSB-36PCIEXP, and actual address has the value that added a starting address of the board.

Table 2-1 Memory map

Address	Function
000H ～ 595H	MKY36
596H ～ BFFH	reserved
C00H	Chip Reset Register
C02H ～ DFFH	reserved
E00H	Board ID Register
E02H ～ FFFH	reserved

2.3.1 Access to MKY36

For details of MKY36, refer to "Chapter 2 MKY36 Software" - "2.1 Memory Map" of “MKY36 User’s Manual".

2.3.2 Registers unique to HLSB-36PCIEXP

C00H and E00H address registers are listed in Table 2-1 Memory map is unique to HLSB-36PCIEXP. Details of C00H and E00H are given below.

Chip Reset Register address : 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
Function	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	CRST0

[Function] Writing "1" to CRST0 (Chip ReSeT0) bit (bit 0) enables to apply reset signal to RST pin of MKY36. Reset period to the RST pin of MKY36 is 280ns. This register is write-only. If you read undefined data.

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

[Function] Reading BID0, 1 (Board ID) bit enables to obtain Board ID value that is set to SW5.

Do not access to addresses (596H to BFFH , C02H to DFFH and E02H to FFFH), Refer to "Table 2-1 Memory map".
To do so, it may cause unstable system operation.

2.4 Access without attached driver

For direct access to HLSB-36PCIEXP without Step Technica's attached driver, please note the following points.

When accessing HLSB-36PCIEXP, always use 32bit. Data in the lower 16bits is valid and data in the upper 16bits is not used. Therefore, address value to access need to be the value which multiplied the actual address.
For example, to read address 200H, read address 400H of HLSB-36PCIEXP with 32bit acccess and obtain word data in address 200H of MKY36 in the lower 16bits out of 32bit data that is read. The same method is necessary to make read or write acccess to registers that are unique to HLSB-36PCIEXP.

Chapter 3 Software

This chapter describes the API provided by Step Technica.
This manual is described based on the API major version number "2" or more than "2".
Please confirm the latest information by our company website when using the product.

3.1 Outline

The API for accessing HLSB-36PCIEXP has available on StepTechnica's website.
Please download the API from StepTechnica download site.

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

Supported OS is
· Windows 8.1 (32bit, 64bit)
· Windows 8 (32bit, 64bit)
· Windows 7 (32bit, 64bit)

This API can be called from Microsoft Visual Studio and VB6 etc..

3.2 Copyright and disclaimer

① Step Technica Co., Ltd. assume no responsibility for any results caused by using the attached driver disc or all softwares downloaded from our website.
② Use API in proper ways with its instructions.
③ All specifications and contents is subject to change without prior notice.
About the compatibility of the future, we do not guarantee absolutely.
④ We can not support for inquiries about OS and development environment and the like.
⑤ If you have found error, please contact our system development department.

3.3 File organization

The files in “DLL” folder are given below.

【hlsb36pciexp.dll】
DLL body. Copy it to the system folder of Windows or the directory in which there is the user program using this DLL before use.

【hlsb36pciexp.lib】
import library for Microsoft Visual C / C ++.

【hlsb36pciexp.h】
API header file. Please include after than Windows.h.

3.4 Limitations

This section describes restrictions on creating applications using this API.

3.4.1 Multi thread

The API can not be used simultaneously from multiple threads.
Consider not to generate a collective call if the application has multithreaded structure.

3.4.2 Power saving mode

HLSB-36 PCIEXP does not support the power saving mode function.
Please stop using the OS's sleep function.When going to sleep mode, electricity to the installed MKY36 Source supply is cut off and communication stops.

3.4.3 Interrupt processing

Can enable, disable or confirm the MKY36's interrupt by setting the INT0R INT1R.
In the driver, manage register that holds the number of times the interrupt occurs in each INT0R and INT1R(“Interrupt generation count register”), holds the information of the lower 8 bit of INT0R and INT1R when an interrupt occurred(“Interrupt generation factor register”).
And do the processing the following using these registers when an interrupt occurs.
(Here is a description of when the interrupt INT0 has occurred.)

① Set the Interrupt factor information to "Interrupt generation factor register".
(Until there is a CLR instruction of the "Interrupt generation factor register" from the user application, the past interrupt factor remain.)
② Increment the value of the "Interrupt generation count register".
③ Writing "1" to the place where "1" is set among 0 to 7 bits of INT0 R to cancel MKY36 interrupt.

Can get information or clear "Interrupt generation count register" and "Interrupt generation factor register " by the API function.

(1) Function to read the value of the interrupt generation count register.
(HlsGetInterrupt0Count , HlsGetInterrupt1Count)
Driver count the interrupts from the MKY36(to each INT0 and INT1).
This function will return this count value.
(2) Function to clear "Interrupt generation count register".
(HlsClearInterrupt0Count , HlsClearInterrupt1Count)
Clear the "Interrupt generation count register".
(3) Function to read a value of "Interrupt generation factor register".
(HlsGetInterrupt0StatusInfo , HlsGetInterrupt1StatusInfo)
Driver holds the interrupts factor from the MKY36(to each INT0 and INT1) .
This information is holds in "Interrupt generation factor register".
This function will return information of "Interrupt generation factor register".
(4) Function of clear "Interrupt factor register".
(HlsClearInterrupt0StatusInfo , HlsClearInterrupt1StatusInfo)
Clear the "Interrupt generation factor registe".

In user application, using these functions to access "Interrupt generation count register" and "Interrupt generation factor register".

3.5 API specification

A list of API functions is shown in Table 3-1.
The API functions described below are the functions contained in hlsb36pciexp.h.

Table 3-1 API function

Function	Description
HlsGetVersion	Obtain the version number of API
HlsGetLastError	Obtain the termination status of API function
HlsOpenHandle	Obtain handles to HLSB-36PCIEXP
HlsCloseHandle	Close handle obtained by HlsOpenHandle
HlsCountDevice	Obtain the number of HLSB-36PCIEXP connected to PC
HlsSearchBoard	Obtain the number of HLSB-36PCIEXP connected to PC and obtain the Board ID
HlsResetBoard	Reset to MKY36
HlsBoardID	obtain board ID
HlsReadByte	Read 1 byte of data from the specified address
HlsWriteByte	Write 1 byte of data to the specified address
HlsReadWord	Read 2 bytes of data from the specified address
HlsWriteWord	Write 2 bytes of data to the specified address
HlsGetInterrupt0Count HlsGetInterrupt1Count	Obtain the INT0, 1 interrupt occurrence count register information inside the driver
HlsClearInterrupt0Count HlsClearInterrupt1Count	Clear the INT0, 1 interrupt occurrence count register information inside the driver
HlsGetInterrupt0StatusInfo HlsGetInterrupt1StatusInfo	Obtain INT0, 1 interrupt cause information inside the driver
HlsClearInterrupt0StatusInfo HlsClearInterrupt1StatusInfo	Clear the interrupt factor of information INT0, 1 that is held inside the driver

3.5.1 HlsGetVersion

Format

UINT HlsGetVersion(void);

Function

Obtain the version number of API.

Parameter

None

Return value

Version number of API(BCD code Hexadecimal)
(Major number + Minor number + Update number)

Error code

After executing this function, get the error code by call HlsGetLastError and the cause of error occurrence are as follows.

HLS_SUCCESS Terminated normally

Notes

The configuration of the version number is shown in Table 3-2. The cause of the upgrade of version number is as follows.

Major Number：The revision no compatibility such as API specification change.
Minor number：The revision with backword compatibility such as API addetion.
Update number：The revision with no specification change such as bug fixes.

Table 3-2 Structure of version number

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

3.5.2 HlsGetLastError

Format

UINT HlsGetLastError(void);

Function

Obtain the termination state of the API function called at last.

Parameter

None

Return value

Returns the error code defined in hls36pciexp.h.

Notes

Table 3-3 lists the error codes defined in hls36pciexp.h.

Table 3-3 List of Error code

Symbol constant	Value	Function
HLS_SUCCESS	0	Terminated normally
HLS_ERR_DEVICENOTEXIST	1	Device does not exist
HLS_ERR_ALREADYOPENED	2	Handle is already opened
HLS_ERR_CLOSED	3	'HlsOpenHandle' has never been called
HLS_ERR_INVALIDPARAM	4	Called with invalid parameter
HLS_ERR_NORESOUCE	5	No resource to execute the process
HLS_ERR_FAILED	6	The process failed due to unknown reason
HLS_NOTCALLYET	99	API has never been called yet

3.5.3 HlsSearchBoard

Format

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

Function

Get the number of HLSB-36 PCIEXP boards inserted in PC and the board ID list.

Parameter

*board_num	Specify the address to the byte type variable in which the number of boards are set. The meanings of the set values are as follows. ・0 : no inserted ・1 ～ 4 : Number of boards recognized ・-1 : 5 or more
*board_id_list	To receive the board ID, specify a pointer to an array with four byte types. It is also possible to specify NULL.If NULL is specified, only the number of boards is returned. The meanings of the set values are as follows. ・0x00 ～ 0x03 : Board ID ・0xFF : Could not recognize

Return value

Succeeded : TRUE is return.
Failed : FALSE is return.

Error code

After executing this function, get the error code by call HlsGetLastError and the cause of error occurrence are as follows.

HLS_SUCCESS	Terminated normally
HLS_ERR_INVALIDPARAM	* board_num is NULL
HLS_ERR_FAILED	The process failed due to unknown reason

Notes

The board ID is set by SW5.
If insert more than one HLSB-36PCIEXP boards, it can be identified by board ID.
This API function can identify up to four HLSB-36PCIEXP.
Specify the byte type array as a parameter as shown below.

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

As example, Three HLS-36PCIEXPs are insert to the PC, and each board ID is 1st board ID = 0, 2nd board ID = 1, 3rd board board ID = 2 is set.
The order recognized by the PC is the first, third, and second. At that time, run HlsSearchBoard.

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

3.5.4 HlsCountDevice

Format

INT HlsCountDevice(void);

Function

Get the number of HLSB-36PCIEXP inserted to PC.

Parameter

None

Return value

Returns the number of HLSB-36PCIEXP.

- -1 : 5 or more
- 0 : Not inserted
- 1 ～ 4 : 1 to 4

Error code

After executing this function, get the error code by call HlsGetLastError and the cause of error occurrence are as follows.

HLS_SUCCESS	Terminated normally

Notes

Cannot insert more than 5 units on one PC.

3.5.5 HlsBoardID

Format

INT HlsBoardID(HANDLE HLSBHandle);

Function

Obtain board ID of HLSB-36PCIEXP.

Parameter

HANDLE HLSBHandle	Handle value of HLSB-36PCIEXP

Return value

At normal termination, board ID (0 to 3) is returned. On failure, -1 is returned.

Error code

After executing this function, get the error code by call HlsGetLastError and the cause of error occurrence are as follows.

HLS_SUCCESS	Terminated normally
HLS_ERR_INVALIDPARAM	Invalid HLSBHandle specified
HLS_ERR_FAILED	The process failed due to unknown reason

Notes

Cannot insert more than 5 units on one PC.

3.5.6 HlsOpenHandle

Format

HANDLE HlsOpenHandle(intindex_no);

Function

Open handles to HLSB-36PCIEXP board

Parameter

index_no	Index number The index number is 0 to 3. When HLSB-36PCIEXP board is 1 unit, set to 0. See "Notes" for details.

Return value

At normal termination, 1 or more is returned. On failure, -1(INVALID_HANDLE_VALUE) is returned.

Error code

After executing this function, get the error code by call HlsGetLastError and the cause of error occurrence are as follows.

HLS_SUCCESS	Terminated normally
HLS_ERR_DEVICENOTEXIST	Device does not exist
HLS_ERR_FAILED	The process failed due to unknown reason

Notes

Close the handle with "HlsCloseHandle" at program termination.

If only have one HLSB-36PCIEXP board, not necessary to execute "HlsSearchBoard".
If multiple HLSB-36PCIEXP boards, execute "HlsSearchBoard" first, It is necessary to check the HLSB-36PCIEXP to be operated on.
As example, Three HLS-36PCIEXPs are insert to the PC, and each board ID is 1st board ID = 0, 2nd board ID = 1, 3rd board board ID = 2 is set. In order to obtain a handle with Board ID = 2.

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

Result of executing,
board_id_list[0] = 0, board_id_list[1] = 2, board_id_list[2] = 1, board_id_list[3] = 0xFF
That way, assume.
In this case, you can confirm that the index number 1 is board ID = 2.
In other words, The index number which is a parameter of "HlsOpenHandle" is 1.

3.5.7 HlsCloseHandle

Format

BOOL HlsCloseHandle(HANDLE HLSBHandle);

Function

This API closes the handle which is obtained with "HlsOpenHandle".

Parameter

HLSBHandle	Handle value of HLSB-36PCIEXP

Return value

Succeeded : TRUE is return.
Failed : FALSE is return.

Error code

After executing this function, get the error code by call HlsGetLastError and the cause of error occurrence are as follows.

HLS_SUCCESS	Terminated normally
HLS_ERR_INVALIDPARAM	Invalid HLSBHandle specified
HLS_ERR_FAILED	The process failed due to unknown reason

3.5.8 HlsReadByte

Format

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

Function

Read one byte of data from the specified address.

Parameter

HLSBHandle	Handle value of HLSB-36PCIEXP
Adr	Address Input conditions are as follows ・Input range：0x0000 ～ 0x0FFF
*Dat	Address of read data storage destination

Return value

Succeeded : TRUE is return.
Failed : FALSE is return.

Error code

After executing this function, get the error code by call HlsGetLastError and the cause of error occurrence are as follows.

HLS_SUCCESS	Terminated normally
HLS_ERR_INVALIDPARAM	Invalid HLSBHandle specified Address out of range NULL was specified for * Dat
HLS_ERR_FAILED	The process failed due to unknown reason

3.5.9 HlsWriteByte

Format

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

Function

Write one byte of data to the specified address.

Parameter

HLSBHandle	Handle value of HLSB-36PCIEXP
Adr	Address Input conditions are as follows ・Input range：0x0000 ～ 0x0FFF
Dat	write data

Error code

After executing this function, get the error code by call HlsGetLastError and the cause of error occurrence are as follows.

HLS_SUCCESS	Terminated normally
HLS_ERR_INVALIDPARAM	Invalid HLSBHandle specified Address out of range
HLS_ERR_FAILED	The process failed due to unknown reason

3.5.10 HlsReadWord

Format

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

Function

Read two bytes of data from the specified address.

Parameter

HLSBHandle	Handle value of HLSB-36PCIEXP
Adr	Address Input conditions are as follows ・Multiples of 2 ・Input range：0x0000 ～ 0x0FFE
*Dat	Address of read data storage destination

Error code

After executing this function, get the error code by call HlsGetLastError and the cause of error occurrence are as follows.

HLS_SUCCESS	Terminated normally
HLS_ERR_INVALIDPARAM	Invalid HLSBHandle specified Address out of range Address is not a multiple of 2 NULL was specified for * Dat
HLS_ERR_FAILED	The process failed due to unknown reason

3.5.11 HlsWriteWord

Format

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

Function

Write two bytes of data to the specified address.

Parameter

HLSBHandle	Handle value of HLSB-36PCIEXP
Adr	Address Input conditions are as follows ・Multiples of 2 ・Input range：0x0000 ～ 0x0FFE
Dat	write data

Error code

After executing this function, get the error code by call HlsGetLastError and the cause of error occurrence are as follows.

HLS_SUCCESS	Terminated normally
HLS_ERR_INVALIDPARAM	Invalid HLSBHandle specified Address out of range Address is not a multiple of 2
HLS_ERR_FAILED	The process failed due to unknown reason

3.5.12 HlsResetBoard

Format

BOOL HlsResetBoard(HANDLE HLSBHandle);

Function

Reset for MKY36 on the specified HLSB-36PCIEXP board.

Parameter

HLSBHandle	Handle value of HLSB-36PCIEXP

Return value

Succeeded : TRUE is return.
Failed : FALSE is return.

Error code

After executing this function, get the error code by call HlsGetLastError and the cause of error occurrence are as follows.

HLS_SUCCESS	Terminated normally
HLS_ERR_INVALIDPARAM	Invalid HLSBHandle specified
HLS_ERR_FAILED	The process failed due to unknown reason

3.5.13 HlsGetInterrupt0Count , HlsGetInterrupt1Count

Format

BOOL HlsGetInterrupt0Count (HANDLE HLSBHandle, BYTE *int0Counter);
BOOL HlsGetInterrupt1Count (HANDLE HLSBHandle, BYTE *int1Counter);

Function

Obtain the INT0, 1 interrupt occurrence count register information inside the driver.
The number of interrupt occurrences is incremented from 0 to 255 (0xFF).

Parameter

HLSBHandle	Handle value of HLSB-36PCIEXP
*int0Counter, *int1Counter	An address to the byte area that stores the acquired interrupt occurrence count.

Return value

Succeeded : TRUE is return.
Failed : FALSE is return.

Error code

After executing this function, get the error code by call HlsGetLastError and the cause of error occurrence are as follows.

HLS_SUCCESS	Terminated normally
HLS_ERR_INVALIDPARAM	Invalid HLSBHandle specified NULL was specified for *int0Counter or *int1Counter
HLS_ERR_FAILED	The process failed due to unknown reason

3.5.14 HlsClearInterrupt0Count , HlsClearInterrupt1Count

Format

BOOL HlsClearInterrupt0Count (HANDLE HLSBHandle);
BOOL HlsClearInterrupt1Count (HANDLE HLSBHandle);

Function

Clear the INT0, 1 interrupt occurrence count register information inside the driver.

Parameter

HLSBHandle	Handle value of HLSB-36PCIEXP

Return value

Succeeded : TRUE is return.
Failed : FALSE is return.

Error code

After executing this function, get the error code by call HlsGetLastError and the cause of error occurrence are as follows.

HLS_SUCCESS	Terminated normally
HLS_ERR_INVALIDPARAM	Invalid HLSBHandle specified
HLS_ERR_FAILED	The process failed due to unknown reason

3.5.15 HlsGetInterrupt0StatusInfo , HlsGetInterrupt1StatusInfo

Format

BOOL HlsGetInterrupt0StatusInfo (HANDLE HLSBHandle, BYTE *int0Info);
BOOL HlsGetInterrupt1StatusInfo (HANDLE HLSBHandle, BYTE *int1Info);

Function

Obtain INT0, 1 interrupt factor information inside the driver.

Parameter

HLSBHandle	Handle value of HLSB-36PCIEXP
*int0Info, , *int1Info	An address to the byte area that stores the acquired interrupt factor count. Interrupt factors that occurred in the past are accumulated.

Return value

Succeeded : TRUE is return.
Failed : FALSE is return.

Error code

After executing this function, get the error code by call HlsGetLastError and the cause of error occurrence are as follows.

HLS_SUCCESS	Terminated normally
HLS_ERR_INVALIDPARAM	Invalid HLSBHandle specified NULL was specified for *int0Info or *int1Info
HLS_ERR_FAILED	The process failed due to unknown reason

Notes

The composition of Int0Info and Int1Info is shown in Table 3-4.
When an interrupt occurs, the bit corresponding to the cause is "1".
The placement of the interrupt factor is same to the lower 8 bits of INT0R and INT1R of MKY36.

Table 3-4 Internal configuration of int0Info , int1Info

bit	Interrupt source
7	Generates an interrupt when scanning is stopped
6	Generates an interrupt when CHECK-2 occurs
5	Generates an interrupt when CHECK-1 occurs
4	Generates an interrupt when a DREQ is issued from the satellite IC
3	Generates an interrupt when a scan is completed
2	Generates an interrupt by DR2 function
1	Generates an interrupt by DR1 function
0	Generates an interrupt by DR0 function

3.5.16 HlsClearInterrupt0StatusInfo , HlsClearInterrupt1StatusInfo

Format

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

Function

Clear the interrupt factor of information INT0, 1 that is held inside the driver.

Parameter

HLSBHandle	Handle value of HLSB-36PCIEXP
clearInt0Info, clearInt1Info	Interrupt generation factor to clear

Return value

Succeeded : TRUE is return.
Failed : FALSE is return.

Error code

After executing this function, get the error code by call HlsGetLastError and the cause of error occurrence are as follows.

HLS_SUCCESS	Terminated normally
HLS_ERR_INVALIDPARAM	Invalid HLSBHandle specified
HLS_ERR_FAILED	The process failed due to unknown reason

Notes

Table 3-5 shows the configuration of interrupt generation factors and setting values.
Set the set value corresponding to the cause of interrupt generation to clearInt0Info, clearInt1Info.
To clear multiple interrupt generation factors, set the logical sum of each set value.

Table 3-5 Interrupt generation factor to be cleared and set value

Interrupt generation factor	Set value
Clear interrupt by scanning stop	0x80
Clear interrupt by CHECK-2 occurs	0x40
Clear interrupt by CHECK-1 occurs	0x20
Clear interrupt by DREQ occurs.	0x10
Clear interrupt by single-scan complete	0x08
Clear interrupt by DR2 function	0x04
Clear interrupt by DR1 function	0x02
Clear interrupt by DR0 function	0x01

3.6 Sample program

3.6.1 Access to MKY36

Sample program of MKY36 initialization, HLS communication setting, Do, Di operation using this API.

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

        /* Generate a handle for HLSB-36 PCIEXP */
        HLSBHandle = HlsOpenHandle(0);
        /* Check the generated handle */
        if (HLSBHandle == INVALID_HANDLE_VALUE) {
                exit(1);   /* FALSE : end of program*/
        }

        /* Initialization of MKY36 */
        // (1) Write 0x000 to 0x57F in the MKY36 memory map with 0x00
        for (i=0;i<0x580;i+=2) {
                HlsWriteWord(HLSBHandle, i, 0);
        }

        // (2) Set scan operating condition of HLS to BCR.
        //     Set FH=0 (half duplex) and BPS 1,0=2 (6Mbps).
        HlsWriteWord(HLSBHandle, 0x58E, 0x0002);

        // (3) If necessary, set the output initial value to the Do area (0x80 to 0xFF).
        //     In this sample program, initial value is not specified.

        /* Write FS (Final Satellite) to SCR */
        HlsWriteWord(HLSBHandle, 0x580, 0x003F);

        /* Obtain Di information of SA1 */
        HlsReadWord(HLSBHandle, 0x0102, &sa1_di);

        /* Obtain Di information of SA63 */
        HlsReadWord(HLSBHandle, 0x017E, &sa63_di);

        /* Change Do information of SA1 */
        HlsWriteWord(HLSBHandle, 0x0082, 0xFF00);

        /* Change Do information of SA63 */
        HlsWriteWord(HLSBHandle, 0x00FE, 0x00FF);

        /* Close the generated handle */
        HlsCloseHandle(HLSBHandle);

        return 0;
}
```

3.6.2 Interrupt processing sample

Sample program of MKY36 interrupt using this API.

```
int main(int argc, char *argv[])
{
        HANDLE HLSBHandle;
        BYTE int0_current_numOfOccurr;          // Current INT0 interrupt occurrence count
        BYTE int0_lastTime_numOfOccurr;         // Last time INT0 interrupt occurrence count
        BYTE int0_factor;                       // INT0 Interrupt generation factor

        /* Generate a handle for HLSB-36 PCIEXP */
        HLSBHandle = HlsOpenHandle(0);

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

        /* Write FS (Final Satellite) to SCR */
        HlsWriteWord(HLSBHandle, 0x580, 0x003F);

        /* Clear "Interrupt generation factor register" */
        HlsClearInterrupt0StatusInfo(HLSBHandle, 0xFF);

        /*Clear "Interrupt count register" */
        HlsClearInterrupt0Count(HLSBHandle);

        /* Clear variable*/
        int0_lastTime_numOfOccurr = 0;

        /* Set the factor of interrupt generation and generate INT0 interrupt at CHECK-1.*/
        HlsWriteWord(HLSBHandle, 0x586, 0x2000);

        while (1) {
                /* Obtain the interrupt occurrence count register information. */
                HlsGetInterrupt0Count(HLSBHandle, &int0_current_numOfOccurr);

                /* Compared with the previous number of interrupt occurrences,
                if different, an interrupt has occurred. */

                if (int0_lastTime_numOfOccurr != int0_current_numOfOccurr) {
                        /* Copy current value to previous value */
                        int0_lastTime_numOfOccurr = int0_current_numOfOccurr;
                        /* Obtain Interrupt generation factor register information. */
                        HlsGetInterrupt0StatusInfo(HLSBHandle, &int0_factor);

                        /* Check the cause of interrupt generation is CHECK-1. */
                        if ((int0_factor & 0x20) == 0x20) {
                                /* --- Processing of CHECK-1 --- */

                                /* Clear CHECK-1 of Interrupt generation factor register. */
                                HlsClearInterrupt0StatusInfo(HLSBHandle, 0x20);
                        }
                }
        }
        /*Close the generated handle */
        HlsCloseHandle(HLSBHandle);
        return 0;
}
```

■ Developed and manufactured by
Step Technica Co., Ltd
757-3, Shimofujisawa, Iruma, Saitama
TEL: +81-4-2964-8804
http://www.steptechnica.com/
info@steptechnica.com

HLS(MKY36) PCI Express Board
HLSB-36PCIEXP
User's Manual

Document No.：STD-HLSB36PCIEXP_V2.0E
Date of publication：March 2017