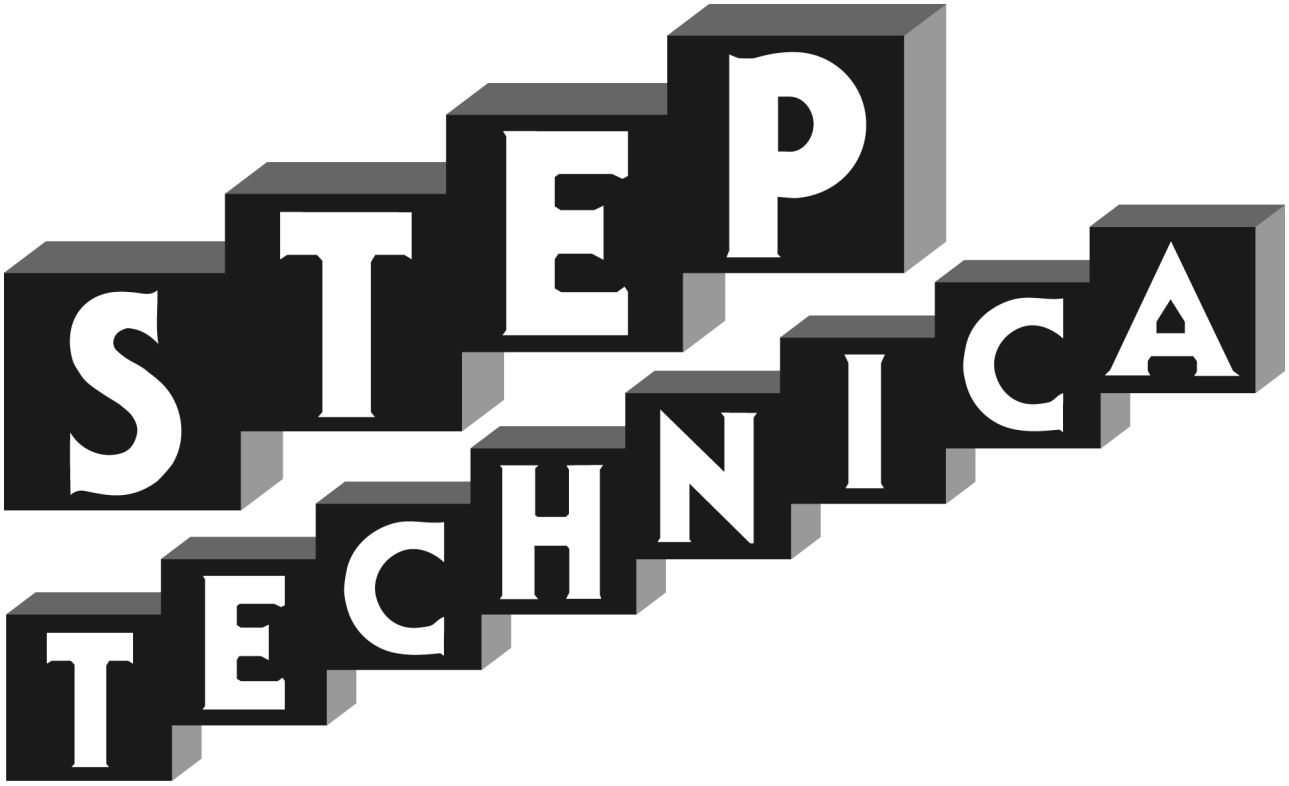
STD_HLS36USBSW_V1.1E

STEP TECHNICA CO.,LTD.



HLS（MKY36）USB Unit

HLS-36USB

Software Manual

Preface

This document describes API included with HLS-36USB unit.

Before using the product, please check the latest information on our website.

In this document, the "Hi-speed Link System" is abbreviated as "HLS".

● Target readers

Those who do the programming with HLS-36USB to build an HLS.

● Prerequisites

This manual assumes that you are familiar with :

・Network technology.

・Semiconductor products (especially microcontrollers and memory).

・Windows Application Programming

● Related manuals

・Hi-speed Link System Introduction Guide

・Hi-speed Link System Technical Guide

・Hi-speed Link System Center IC MKY36 User’s Manual

・HLS-36USB Product Manual

【Note】

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).

Revision history

Version	Date	Content	
		Page	Description
1.0E	AUG 2018		Issued the first edition
1.1E	APR 2020		Added Windows 10 as supported OS

Table of Contents

Tables

Chapter 1 Outline

StepTechnica Co., Ltd., provides an API to access HLS-36USB for the user application.
This document is intended for use with firmware version "V_1.0" and API version "1.0.0" of HLS-36USB.
Please download the API from StepTechnica website's 'Downloads' page.
URL: https://www.steptechnica.com/en/download/index.html

Chapter 2 System requirements

The API works correctly in the following operating systems.
· Windows 10 (64bit / 32bit)
· Windows 8.1 (64bit / 32bit)
· Windows 8 (64bit / 32bit)
· Windows 7 (64bit / 32 bit)
This API can be called from Microsoft Visual Studio and VB6 etc.

Chapter 3 Copyright and disclaimer

1. StepTechnica Co., Ltd. assumes no responsibility for any results caused by using the attached driver disk or all softwares 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 any compatibility in the future.
4. We can not support for inquiries regarding OS or a development environment.
5. If you have found an error, please contact our system development department.

Chapter 4 File structure

The files in “DLL” folder are described in the following.

```
[ DLL]
    |
    + ---- [hls36usb.dll] : DLL body. Before your use, copy it to the system folder of Windows or the directory in
    |                       which user program using this DLL is stored.
    + ---- [hls36usb.lib] : Import library
    + ---- [hls36usb.h]   : DLL header file. Please include after than Windows.h.
```

Chapter 5 Limitations

This chapter describes limitations when creating an application using this API.

5.1 Multi-thread

This API Function cannot be used from multiple threads at the same time.
Consider not to generate a collective call if the application has multithreaded structure.

5.2 Timeout in USB communication

In this API, the maximum waiting time (timeout) of data transmission and reception between HLS-36USB is 1 second. Even if the timeout period is over, sending and receiving may not end in some of the system environments.
Return parameters of the API function returns an error when timeout has been occurred.
If a timeout occurs, the HLS network and periodic update function stops in the internal API.
The HLS network and periodic transmission function was able to stop normally, the following is set to the error code.
HLS_ERR_USB_TIMEOUT_SUCCESS_STOP_HLS (9)
If it fails to stop HLS network and periodic transmission function, the following is set to the error code.
HLS_ERR_USB_TIMEOUT_FAILED_STOP_HLS (10)

After the timeout occurs, close the handle which is used in HlsCloseHandle function, please reobtain a handle at HlsOpenHandle function. Until a handle is reobtained, the return parameter of the API function other than HlsGetVersion,HlsCountDevice, HlsSearchBoard,HlsGetLastError, HlsOpenHandle and HlsCloseHandle returns an error.
At that time, HlsGetLastError function returns HLS_ERR_REACQUISITION_OF_HANDLE (11).

5.3 Power saving mode

This product is not applied to the power saving mode of PC (personal computer).

Chapter 6 API specifications

This chapter describes API specifications.
This API is prepared for easy operation of HLS-36USB for user application.
In addition to the normal function to read and write to MKY36, this API has a function to sample all control word of MKY36, all DI and all DRC at specified cycle. This function is called "periodic update".

The API function list is shown in Table 6-1.

Table 6-1 API functions

API Function	Description
HlsGetVersion	Obtains the version number of API
HlsGetLastError	Obtains the termination status of API function
HlsOpenHandle	Opens a handle of HLS-36USB
HlsCloseHandle	Closes a handle obtained by HlsOpenHandle
HlsCountDevice	Obtains the number of HLS-36USB connected to PC
HlsBoardID	Obtain the board ID
HlsSearchBoard	Obtains the number of HLS-36USB connected to PC and obtain the board ID
HlsResetMKY36	Orders a reset to MKY36
HlsStartAutoTrans	Starts Periodic Update
HlsStopAutoTrans	Stops Periodic Update
HlsReadWord	Reads 2 bytes of data from the specified address
HlsWriteWord	Writes 2 bytes of data to the specified address
HlsReadCTL	Obtains the latest data of all control words by periodic update
HlsReadDI	Obtains the latest data of all Di by Periodic Update
HlsReadDRC	Obtains the latest data of all DRC by Periodic Update
HlsReadData	Reads data of the specified word length from the specified address
HlsWriteData	Writes data of the specified word length to the specified address
HlsGetFirmwareVersion	Obtains the firmware version number of HLS-36USB

6.1 HlsGetVersion

Format

UINT HlsGetVersion(void);

Function

Obtains the version number of API.

Parameter

None

Return value

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

Error code

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

HLS_SUCCESS Terminated normally

Note

The configuration of API version number is as shown in Table 6-2.
The reasons for updating the version number are as follows.

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

Table 6-2 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

6.2 HlsGetLastError

Format

UINT HlsGetLastError(void);

Function

Obtains the termination state of the API function called last time.

Parameter

None

Return value

Returns the error code defined in hls36usb.h.

Note

The error codes defined in hls36usb.h. are shown in Table 6-3.

Table 6-3 Error code list

Error Code	Value	Content
HLS_SUCCESS	0	Terminated normally
HLS_ERR_DEVICENOTEXIST	1	Device does not exist.
HLS_ERR_ALREADYOPENED	2	Handle has been 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_ERR_AUTO_TRANS_ALREADY_START	7	Periodic update has already started.
HLS_ERR_AUTO_TRANS_STOP	8	Periodic update has not started.
HLS_ERR_USB_TIMEOUT_SUCCESS_STOP_HLS	9	Timeout has occurred during USB communication, and HLS communication was successfully stopped.
HLS_ERR_USB_TIMEOUT_FAILED_STOP_HLS	10	Timeout has occurred during USB communication, and HLS communication failed to be stopped.
HLS_ERR_REACQUISITION_OF_HANDLE	11	Handle has not been reobtained.
HLS_ERR_NOT_SUPPORT_FIRM_VERSION	12	Unsupported firmware version
HLS_ERR_INVALID_SEQUENCE_NUMBER	13	Invalid sequence number
HLS_NOTCALLYET	99	HLSAPI has never been called.

6.3 HlsCountDevice

Format

INT HlsCountDevice(void);

Function

Obtains the number of HLS-36USB connected to PC.

Parameter

None.

Return value

Returns the number of HLS-36USB connected to PC.

-1 : 5 or more
0 : Not connected
1 to 4 : 1 to 4

Error code

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

HLS_SUCCESS	Terminated normally

Note

No more than five devices can be connected to a PC.

6.4 HlsBoardID

Format

INT HlsBoardID(HANDLE HLSHandle);

Function

Obtains board ID of HLS-36USB.

Parameter

HANDLE HLSHandle　　Handle value of HLS-36USB

Return value

Succeeded: Board ID (0 to 3) is returned.
Failed　　: -1 is returned.

Error code

The error codes and error factors returned by the HlsGetLastError after executing this function are as follows.

HLS_SUCCESS	Terminated normally
HLS_ERR_INVALIDPARAM	Invalid HLSHandle is specified.
HLS_ERR_USB_TIMEOUT_SUCCESS_STOP_HLS	Timeout has occurred during USB communication, and HLS communication was successfully stopped.
HLS_ERR_USB_TIMEOUT_FAILED_STOP_HLS	Timeout has occurred during USB communication, and HLS communication failed to be stopped.
HLS_ERR_REACQUISITION_OF_HANDLE	Handle has not been reobtained.
HLS_ERR_INVALID_SEQUENCE_NUMBER	Invalid sequence number
HLS_ERR_FAILED	The process failed due to unknown reason.

6.5 HlsSearchBoard

Format

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

Function

Obtains the number of HLS-36USB connected to PC and obtain the Board ID list.

Parameter

*board_num	Specify the address to the byte type variable in which the number of boards is set. The meanings of the set values are as follows. -1 : 5 or more 0 : Not connected 1 to 4 :Number of boards identified
*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 has been specified, only the number of boards is returned. The meanings of the set values are as follows. 0x00 to 0x03 : Board ID 0x80 : Handle value has already been obtained by HlsOpenHandle. 0xFF : No board has been identified.

Return value

Succeeded:TRUE(1) is returned.
Failed : FALSE(0) is returned.

Error code

The error codes and error factors returned by the HlsGetLastError after executing this function are as follows.

HLS_SUCCESS	Terminated normally
HLS_ERR_INVALIDPARAM	* board_num is NULL
HLS_ERR_USB_TIMEOUT_SUCCESS_STOP_HLS	Timeout has occurred during USB communication, and HLS communication was successfully stopped.
HLS_ERR_USB_TIMEOUT_FAILED_STOP_HLS	Timeout has occurred during USB communication, and HLS communication failed to be stopped.
HLS_ERR_INVALID_SEQUENCE_NUMBER	Invalid sequence number
HLS_ERR_FAILED	The process failed due to unknown reason.

Addendum

The board ID is set by Option switch. If two or more HLS-36USB devices are connected, it can be distinguished by board IDs. This API function can identify up to four HLS-36USB devices. 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 an example, three HLS-36USB devices are connected to the PC, and each board IDs are set in sequence ; 1st board ID = 0, 2nd board ID = 1, 3rd board board ID = 2.

If the devices have been identified by the PC in sequence with first, third, and second, and run HlsSearchBoard, 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

6.6 HlsStartAutoTrans

Format

BOOL HlsStartAutoTrans(HANDLE HLSHandle, WORD MfCnt);

Function

Starts periodic update of HLS-36USB (all control word, all Di, all DRC).
The update cycle can be specified in units of 125 μs.
The updated data is retained inside the API. Retained data can be obtained with HlsReadCTL, HlsReadDI, HlsReadDRC.

Parameter

HANDLE HLSHandle	Handle value of HLS-36USB
WORD MfCnt	Set update cycle. The update cycle can be specified in units of 125 μs from 1 ms to 100 ms. Regarding the update cycle interval, refer to Table 6-4 to set the period. The setting values other than in Table 6-4 will be an error.

Table 6-4 Update cycle setting list

Setting value	Update cycle (μ sec)
8	1,000(1msec)
9	1,125
10	1,250
:	:
797	99,625
798	99,750
799	99,875
800	100,000(100msec)

Return value

Succeeded:TRUE(1) is returned.
Failed : FALSE(0) is returned.

Error code

The error codes and error factors returned by the HlsGetLastError after executing this function are as follows.

HLS_SUCCESS	Terminated normally
HLS_ERR_INVALIDPARAM	Invalid HLSHandle is specified. MfCnt is out of range.
HLS_ERR_AUTO_TRANS_ALREADY_START	Periodic update has already started.
HLS_ERR_USB_TIMEOUT_SUCCESS_STOP_HLS	Timeout has occurred during USB communication, and HLS communication was successfully stopped.
HLS_ERR_USB_TIMEOUT_FAILED_STOP_HLS	Timeout has occurred during USB communication, and HLS communication failed to be stopped.
HLS_ERR_REACQUISITION_OF_HANDLE	Handle has not been reacquired.
HLS_ERR_INVALID_SEQUENCE_NUMBER	Invalid sequence number
HLS_ERR_FAILED	The process failed due to unknown reason.

Note

Please be aware that periodic updating may not be executed due to the specifications of the PC, or other applications running on the same PC.
When using HlsReadCTL, HlsReadDI, HlsReadDRC, please use this API to enable periodic update.

6.7 HlsStopAutoTrans

Format

BOOL HlsStopAutoTrans(HANDLE HLSHandle);

Function

Stops periodic update of HLS-36USB (all control word, all Di, all DRC).

Parameter

HANDLE HLSHandle　　Handle value of HLS-36USB

Return value

Succeeded:TRUE(1) is returned.
Failed　　: FALSE(0) is returned.

Error code

The error codes and error factors returned by the HlsGetLastError after executing this function are as follows.

HLS_SUCCESS	Terminated normally
HLS_ERR_INVALIDPARAM	Invalid HLSHandle is specified.
HLS_ERR_AUTO_TRANS_STOP	Periodic update has not started.
HLS_ERR_USB_TIMEOUT_SUCCESS_STOP_HLS	Timeout has occurred during USB communication, and HLS communication was successfully stopped.
HLS_ERR_USB_TIMEOUT_FAILED_STOP_HLS	Timeout has occurred during USB communication, and HLS communication failed to be stopped.
HLS_ERR_REACQUISITION_OF_HANDLE	Handle has not been reacquired.
HLS_ERR_INVALID_SEQUENCE_NUMBER	Invalid sequence number
HLS_ERR_FAILED	The process failed due to unknown reason.

6.8 HlsOpenHandle

Format

HANDLE HlsOpenHandle(int index_no);

Function

Opens handles to the HLS-36USB.

Parameter

int index_no Index number
You can specify an index number from 0 to 3.
If just one HLS-36USB is connected to PC, set 0.
For more information, see "Addendum".

Return value

Succeeded:1 or greater value is returned.
Failed : -1(INVALID_HANDLE_VALUE) is returned.

Error code

The error codes and error factors returned by the HlsGetLastError after executing this function are as follows.

HLS_SUCCESS	Terminated normally
HLS_ERR_ALREADYOPENED	Handle has been already opened.
HLS_ERR_DEVICENOTEXIST	Device does not exist.
HLS_ERR_NOT_SUPPORT_FIRM_VERSION	Unsupported firmware version
HLS_ERR_REACQUISITION_OF_HANDLE	Invalid sequence number
HLS_ERR_FAILED	The process failed due to unknown reason.

Addendum

It's not necessary to run HlsSearchBoard when just one HLS-36USB is connected to PC.
If two or more HLS-36USB are connected to PC, you must run "HlsSearchBoard" in advance to check which HLS-36USB to manipulate. As an example, three HLS-36USB devices are connected to the PC, and each board IDs are set in sequence ; 1st board ID = 0, 2nd board ID = 1, 3rd board board ID = 2.
To obtain the handle value of Board ID=2, operate as follows.

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

Assuming that the results of executing in the above was the following.

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 index number 1 is the board ID=2.
That means 1 is the index number, the parameter of HlsOpenHandle.
Close the handle with HlsCloseHandle at finishing the program.

6.9 HlsCloseHandle

Format

BOOL HlsCloseHandle(HANDLE HLSHandle);

Function

Closes the handle obtained by HlsOpenHandle.
Stops periodic update as well if it's running.

Parameter

HANDLE HLSHandle	Handle value of HLS-36USB

Return value

Succeeded: TRUE(1) is returned.
Failed : FALSE(0) is returned.

Error code

The error codes and error factors returned by the HlsGetLastError after executing this function are as follows.

HLS_SUCCESS	Terminated normally
HLS_ERR_INVALIDPARAM	Invalid HLSHandle is specified.
HLS_ERR_INVALID_SEQUENCE_NUMBER	Invalid sequence number
HLS_ERR_FAILED	The process failed due to unknown reason.

6.10 HlsResetMKY36

Format

BOOL HlsResetMKY36(HANDLE HLSHandle);

Function

Resets MKY36

Parameter

HANDLE HLSHandle	Handle value of HLS-36USB

Return value

Succeeded: TRUE(1) is returned.
Failed : FALSE(0) is returned.

Error code

The error codes and error factors returned by the HlsGetLastError after executing this function are as follows.

HLS_SUCCESS	Terminated normally
HLS_ERR_INVALIDPARAM	Invalid HLSHandle is specified
HLS_ERR_USB_TIMEOUT_SUCCESS_STOP_HLS	Timeout has occurred during USB communication, and HLS communication was successfully stopped.
HLS_ERR_USB_TIMEOUT_FAILED_STOP_HLS	Timeout has occurred during USB communication, and HLS communication failed to be stopped.
HLS_ERR_REACQUISITION_OF_HANDLE	Handle has not been reobtained.
HLS_ERR_INVALID_SEQUENCE_NUMBER	Invalid sequence number
HLS_ERR_FAILED	The process failed due to unknown reason.

6.11 HlsReadWord

Format

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

Function

Reads 2 bytes of data from the specified address

Parameter

HANDLE HLSHandle	The handle value of HLS-36USB
const ULONG Adr	Address value Input conditions are the following Multiples of 2 Input range : 0x0000 - 0xF02
WORD *Dat	The storage address of read data

Return value

Succeeded:TRUE(1) is returned.
Failed : FALSE(0) is returned.

Error code

The error codes and error factors returned by the HlsGetLastError after executing this function are as follows.

HLS_SUCCESS	Terminated normally
HLS_ERR_INVALIDPARAM	Invalid HLSHandle is specified. Adr is out of range. Adr value is not multiple of 2. NULL has been specified to Dat.
HLS_ERR_USB_TIMEOUT_SUCCESS_STOP_HLS	Timeout has occurred during USB communication, and HLS communication was successfully stopped.
HLS_ERR_USB_TIMEOUT_FAILED_STOP_HLS	Timeout has occurred during USB communication, and HLS communication failed to be stopped.
HLS_ERR_REACQUISITION_OF_HANDLE	Handle has not been reobtained.
HLS_ERR_INVALID_SEQUENCE_NUMBER	Invalid sequence number
HLS_ERR_FAILED	The process failed due to unknown reason.

6.12 HlsWriteWord

Format

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

Function

Writes 2 bytes of data from the specified address

Parameter

HANDLE HLSHandle	The handle value of HLS-36USB
const ULONG	Adr address value Input conditions are the following. Multiples of 2 Input range : 0x0000 - 0xF02
cconst WORD Dat	Write data

Return value

Succeeded:TRUE(1) is returned.
Failed : FALSE(0) is returned.

Error code

The error codes and error factors returned by the HlsGetLastError after executing this function are as follows.

HLS_SUCCESS	Terminated normally
HLS_ERR_INVALIDPARAM	Invalid HLSHandle is specified. Adr is out of range. Adr value is not multiple of 2.
HLS_ERR_USB_TIMEOUT_SUCCESS_STOP_HLS	Timeout has occurred during USB communication, and HLS communication was successfully stopped.
HLS_ERR_USB_TIMEOUT_FAILED_STOP_HLS	Timeout has occurred during USB communication, and HLS communication failed to be stopped.
HLS_ERR_REACQUISITION_OF_HANDLE	Handle has not been reobtained.
HLS_ERR_INVALID_SEQUENCE_NUMBER	Invalid sequence number
HLS_ERR_FAILED	The process failed due to unknown reason.

6.13 HlsReadCTL

Format

BOOL HlsReadCTL(HANDLE HLSHandle, void*Data);

Function

Obtains the latest data of all control words by periodic update.
Error is returned when HlsReadCTL has been called while periodic update was stopping.

Parameter

HANDLE HLSHandle	The handle value of HLS-36USB
void *Data	The storage address of 128 bytes data

Return value

Succeeded: TRUE(1) is returned.
Failed : FALSE(0) is returned.

Error code

The error codes and error factors returned by the HlsGetLastError after executing this function are as follows.

HLS_SUCCESS	Terminated normally
HLS_ERR_INVALIDPARAM	Invalid HLSHandle is specified. NULL has been specified to *Data.
HLS_ERR_AUTO_TRANS_STOP	Periodic update is stopping
HLS_ERR_REACQUISITION_OF_HANDLE	Handle has not been reobtained.
HLS_ERR_FAILED	The process failed due to unknown reason.

Note

HlsReadCTL is an API that obtains data by periodic update, which is not accessed to MKY36 in direct.
When obtaining control word from MKY36 directly, use "HlsReadWord", "HlsReadData".

6.14 HlsReadDI

Format

BOOL HlsReadDI(HANDLE HLSHandle, void *Data);

Function

Obtains the latest data of all Di by periodic update.
Error is returned when HlsReadDI has been called while periodic update was stopping.

Parameter

HANDLE HLSHandle	The handle value of HLS-36USB
void *Data	The storage address of 128 bytes data

Return value

Succeeded:TRUE(1) is returned.
Failed : FALSE(0) is returned.

Error code

The error codes and error factors returned by the HlsGetLastError after executing this function are as follows.

HLS_SUCCESS	Terminated normally
HLS_ERR_INVALIDPARAM	Invalid HLSHandle is specified. NULL has been specified to *Data.
HLS_ERR_AUTO_TRANS_STOP	Periodic update is stopping.
HLS_ERR_REACQUISITION_OF_HANDLE	Handle has not been reobtained.
HLS_ERR_FAILED	The process failed due to unknown reason.

Note

HlsReadDI is an API that obtains data by periodic update, which is not accessed to MKY36 in direct.
When obtaining Di from MKY36 directly, use "HlsReadWord", "HlsReadData".

6.15 HlsReadDRC

Format

BOOL HlsReadDRC(HANDLE HLSHandle, void *Data);

Function

Obtains the latest data of all DRC by periodic update.
Error is returned when HlsReadDRC has been called while periodic update was stopping.

Parameter

HANDLE HLSHandle	The handle value of HLS-36USB
void *Data	The storage address of 128 bytes data

Return value

Succeeded:TRUE(1) is returned.
Failed : FALSE(0) is returned.

Error code

The error codes and error factors returned by the HlsGetLastError after executing this function are as follows.

HLS_SUCCESS	Terminated normally
HLS_ERR_INVALIDPARAM	Invalid HLSHandle is specified. NULL has been specified to *Data.
HLS_ERR_AUTO_TRANS_STOP	Periodic update is stopping.
HLS_ERR_REACQUISITION_OF_HANDLE	Handle has not been reobtained.
HLS_ERR_FAILED	The process failed due to unknown reason.

Note

HlsReadDRC is an API that obtains data by periodic update, which is not accessed to MKY36 in direct.
When obtaining control word from MKY36 directly, use "HlsReadWord", "HlsReadData".

6.16 HlsReadData

Format

BOOL HlsReadData(HANDLE HLSHandle, WORD Adr, WORD WordLen, void *Data);

Function

Reads data of the specified word length from the specified address.

Parameter

HANDLE HLSHandle	The handle value of HLS-36USB
WORD Adr	Address value Input conditions are the following Multiples of 2 Input range : 0x0000 - 0x07FE
WORD WordLen	Word length Input conditions are the following Input range : 0x0001 - 0x0400
void *Data	The storage address of read data

Return value

Succeeded:TRUE(1) is returned.
Failed : FALSE(0) is returned.

Error code

The error codes and error factors returned by the HlsGetLastError after executing this function are as follows.

HLS_SUCCESS	Terminated normally
HLS_ERR_INVALIDPARAM	Invalid HLSHandle is specified. Specified Adr is out of range. Adr value is not multiple of 2. Specified WordLen is out of range. Specified read range exceeded 0x800. NULL has been specified to *Data.
HLS_ERR_USB_TIMEOUT_SUCCESS_STOP_HLS	Timeout has occurred during USB communication, and HLS communication was successfully stopped.
HLS_ERR_USB_TIMEOUT_FAILED_STOP_HLS	Timeout has occurred during USB communication, and HLS communication failed to be stopped.
HLS_ERR_REACQUISITION_OF_HANDLE	Handle has not been reobtained.
HLS_ERR_INVALID_SEQUENCE_NUMBER	Invalid sequence number
HLS_ERR_FAILED	The process failed due to unknown reason.

6.17 HlsWriteData

Format

BOOL HlsWriteData(HANDLE HLSHandle, WORD Adr, WORD WordLen, void *Data);

Function

Writes data of the specified word length to the specified address

Parameter

HANDLE HLSHandle	The handle value of HLS-36USB
WORD Adr	Address value Input conditions are the following Multiples of 2 Input range : 0x0000 - 0x07FE
WORD WordLen	Word length Input conditons are the following Input range : 0x0001 - 0x0400
void *Data	The storage address of write data

Return value

Succeeded: TRUE(1) is returned.
Failed : FALSE(0) is returned.

Error code

The error codes and error factors returned by the HlsGetLastError after executing this function are as follows.

HLS_SUCCESS	Terminated normally
HLS_ERR_INVALIDPARAM	Invalid HLSHandle is specified. Specified Adr is out of range. Adr value is not multiple of 2. Specified WordLen is out of range. Specified write range exceeded 0x800. NULL has been specified to *Data.
HLS_ERR_USB_TIMEOUT_SUCCESS_STOP_HLS	Timeout has occurred during USB communication, and HLS communication was successfully stopped.
HLS_ERR_USB_TIMEOUT_FAILED_STOP_HLS	Timeout has occurred during USB communication, and HLS communication failed to be stopped.
HLS_ERR_REACQUISITION_OF_HANDLE	Handle has not been reobtained.
HLS_ERR_INVALID_SEQUENCE_NUMBER	Invalid sequence number
HLS_ERR_FAILED	The process failed due to unknown reason.

6.18 HlsGetFirmwareVersion

Format

INT HlsGetFirmwareVersion(HANDLE HLSHandle);

Function

Obtains the firmware version number of HLS-36USB.

Parameter

HANDLE HLSHandle The handle value of HLS-36USB

Return value

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

Error code

The error codes and error factors returned by the HlsGetLastError after executing this function are as follows.

HLS_SUCCESS	Terminated normally
HLS_ERR_INVALIDPARAM	Invalid HLSHandle is specified.
HLS_ERR_USB_TIMEOUT_SUCCESS_STOP_HLS	Timeout has occurred during USB communication, and HLS communication was successfully stopped.
HLS_ERR_USB_TIMEOUT_FAILED_STOP_HLS	Timeout has occurred during USB communication, and HLS communication failed to be stopped.
HLS_ERR_REACQUISITION_OF_HANDLE	Handle has not been reobtained.
HLS_ERR_INVALID_SEQUENCE_NUMBER	Invalid sequence number
HLS_ERR_FAILED	The process failed due to unknown reason.

Note

The configuration of API version number is as shown in Table 6-5.
The reasons for updating the version number are as follows.

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

Table 6-5 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

Chapter 7 Appendix

7.1 Sample program

The sample of initializing and finishing program to control HLS-36USB is the following.
Please refer to "2.8 Register Reference" described in MKY36 User's Manual ("Chapter 2: MKY36 Software").

```
int main(int argc, char argv[])
{
    unsigned char buf[0x580];
    unsigned char board_count;
    unsigned char board_id_list[4];
    /** Check an API version number*/
    int version=HlsGetVersion();
    if (version < 0x100 || version > 0x199) {
       printf(" This version of hls36usb.dll is incompatible.\n");
       exit(1);
    }
    /** Search HLS-36USB
     * Up to four HLS-36USB devices can be identified. When five or more devices are connected to PC,
         it returns an error.
     * The number of HLS-36USB devices connected to PC and its Board ID list are returned.
     * It's not necessary to execute HlsSearchBoard when just one HLS-36USB device is connected to PC.
     */
    if (HlsSearchBoard(&board_count, &board_id_list[0])) {
        exit(1);
     }

    If (board_cont == 0) {
       printf("No HLS-36USB is connected to PC. \n");
       exit(1);
     } else if (board_cont == 0xFF) {
         printf("Number of  HLS-36USB devices connected to PC exceeded the limit. \n");
         exit(1);
    }

    /** A handle corresponded with HLS-36USB to control is generated.
     * If only one HLS-36USB is connected to PC, open handle with 0 parameter.
     */
    HANDLE dev_handle;
    dev_handle=HlsOpenHandle(0);
    if (dev_handle == INVALID_HANDLE_VALUE) {
        printf(" Failed to obtain a handle value to HLS-36USB \n");
        exit(1);
    }

    memset(buf, 0, sizeof(buf));
```

```
    /** Initializing HLS-36USB */
    // Clear control, Do, Di, C1 to C7, DRC area (0x000 to 0x57F)
    HlsWriteData(dev_handle, 0, 0x2C0, buf);
    // Setting BCR / HUB nesting (LF), and baud rate (BPS)
    // HUB nesting (LF): 0, Communication method (FH): Half-duplex, Baud rate (BPS) : 6Mbps
    HlsWriteWord (dev_handle, 0x58E, 0x0002);

    /** Start HLS communication after initializing has been finished.
     * For instance, FS value which is subject to be scanned continuously.
     * Setting FS value 63
     */
    HlsWriteWord (dev_handle, 0x580, 0x003F);
    /**Start periodic update. It's not necessary to execute when HlsReadCTL, HlsReadDI and HlsReadDRC
          are not used.
    * Data sending at 1000µs (1msec) cycle
    */
    HlsStartAutoTrans(dev_handle, 8);

    /** -- Describe user process here -- **/

    /** Stop periodic update. (It's not necessary to execute when HlsStartAutoTrans is not used.)*/
    HlsStopAutoTrans(dev_handle);

    /** Stop HLS communication **/
    // Set 0x0000 to SCR to stop HLS communication
    HlsWriteWord (dev_handle, 0x580, 0x0000);

     /** Close the generated handle. */
    HlsCloseHandle(dev_handle);

    return 0;
}
```

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 this 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. No part of this document may be copied or reproduced in any form or by any means without prior written permission from StepTechnica Co., Ltd.

■Developed and manufactured by

StepTechnica Co., Ltd.

757-3, Shimofujisawa, Iruma, Saitama
https://www.steptechnica.com/en/index.html
info@steptechnica.com

HLS （MKY36）USB Unit

HLS-36USB

Software Manual

Document No. STD_HLS36USBSW_V1.1E
Issued: April 2020