P10996-100

HLS Family

HLS-39USB

REFERENCE MANUAL

https://www.steptechnica.com/

HLS Hi-speed Link System

HLS (MKY39-1) USB Unit

HLS-39USB Reference Manual

The HLS-39USB, featuring the MKY39-1, the center IC of Hi-speed Link System (HLS), is designed to control the HLS through USB communication.

This manual provides details on the driver and library which can be used on Windows environments. The HLS can be operated from a PC by using them.

©2026- StepTechnica Co., Ltd.

Preface

This manual describes the software for the HLS-39USB equipped with the HLS IC MKY39-1. Before reading this manual, be sure to read [**HLS Introduction Guide**].

■ Target Readers

This manual is intended for:

- Those who build an HLS network for the first time
- Those who use StepTechnica ICs for the first time to build an HLS network
- Those who use the HLS-39USB to evaluate HLS
- Those who have experience of using HLS with the HLS-36USB

■ Prerequisites

This manual assumes that you are familiar with:

- Network technology
- Semiconductor products (especially microcontrollers and memory devices)

■ Related Manuals

- HLS-39USB Data Sheet
- HLS Introduction Guide
- HLS Technical Guide
- MKY39-1 / MKY39-2 Data Sheet
- MKY39-1 / MKY39-2 Reference 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

1 General description

1.1 Operating environment 1-2
1.2 Copyright and disclaimer 1-2
1.3 Function description 1-2

2 Driver

2.1 File contents 2-1
2.2 Installing driver 2-2

3 Library (DLL)

3.1 File contents 3-1
3.2 Address area 3-2
3.2.1 MKYReset (Adrs=0x0F00) 3-2
3.2.2 ReadID (Adrs=0x0F02) 3-2
3.3 AutoTrans function 3-3
3.4 API references 3-4
3.4.1 HlsGetVersion 3-5
3.4.2 HlsGetLastError 3-6
3.4.3 HlsCountDevice 3-7
3.4.4 HlsSearchBoard 3-8
3.4.5 HlsOpenHandle 3-10
3.4.6 HlsCloseHandle 3-11
3.4.7 HlsBoardID 3-12
3.4.8 HlsResetMKY 3-13
3.4.9 HlsReadWord 3-14
3.4.10 HlsWriteWord 3-15
3.4.11 HlsReadData 3-16
3.4.12 HlsWriteData 3-17
3.4.13 HlsGetFirmwareVersion 3-18
3.4.14 HlsStartAutoTrans 3-19
3.4.15 HlsStopAutoTrans 3-20
3.4.16 HlsReadTransData 3-21
3.5 Limitations 3-22

4 MKY36-compatible mode

4.1 MKY36 registers 4-1

1 General description

The HLS (Hi-speed Link System) can be operated by user applications on a PC connected to the HLS-39USB with a USB cable, using the driver and library explained in this manual.

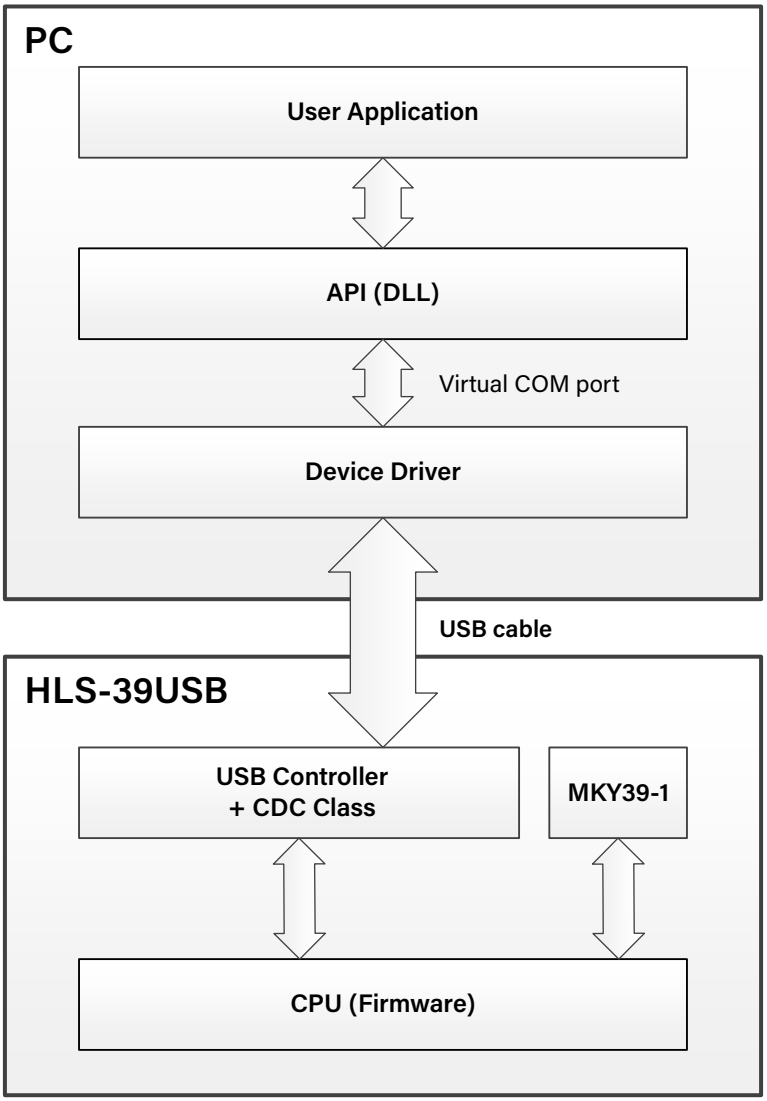


Fig. 1.1 Software components

1.1 Operating environment

- PC with a free USB port for connecting with HLS-39USB
  (The product package does not include a cable. Please prepare one.)
- Operating system: Windows 11

PC's energy-saving scheme is not applicable to this product.

1.2 Copyright and disclaimer

All copyrights for documents, programs, and program sources are owned by StepTechnica Co., Ltd. StepTechnica Co., Ltd. licenses the right to reproduce and use these copyrighted materials only to individuals, corporations, or other organizations that have accepted the following points of caution and use the HLS-39USB. Without the consent of StepTechnica Co., Ltd., it is not permitted to modify, redistribute, or reproduce and use part or all of these copyrighted materials for purposes other than the HLS-39USB.



1. We assume no responsibility whatsoever for any results arising from the use of any software obtained from our website.
2. Please use the library API correctly according to the instructions.
3. All specifications and contents are subject to change without prior notice. We do not provide any guarantee regarding future compatibility.
4. We do not offer any support for inquiries regarding OS or development environments.
5. If you discover any malfunctions or other issues, please contact us through the 'Contact Form' on our website.

1.3 Function description

The HLS-39USB is used to connect a PC to HLS network.
Users can use the MKY39-1 functions easily for evaluation purposes when building an HLS system and for development of satellites, by accessing Windows driver and library supplied separately by StepTechnica.
The MKY36-compatible mode is available so that the environment used with the HLS-36USB can be transferred to this product for continuous use.



Certain functions are no longer supported by HLS-39USB. For details, refer to "4 MKY36-compatible mode".

2 Driver

Windows 64-bit driver is available for HLS-39USB.
Please download it from our website.

URL: https://www.steptechnica.com/en/products/03_board/hls-39usb/hls-39usb.html

The driver performs virtual serial communication using the standard Windows USB serial communication driver (Usbser.sys).

2.1 File contents

The downloaded zip archive contains the following data:

HLS-39USB_Driver_Vx.x.zip	Provided file
├ hls39usb.cat	Catalog file
└ hls39usb.inf	Setup information file

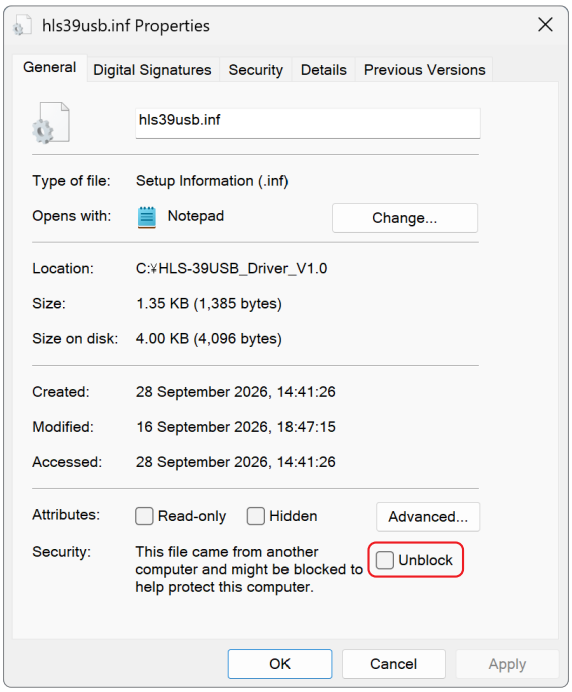
2.2 Installing driver

Follow the instructions below to install the driver which allows your PC to recognize the HLS-39USB.

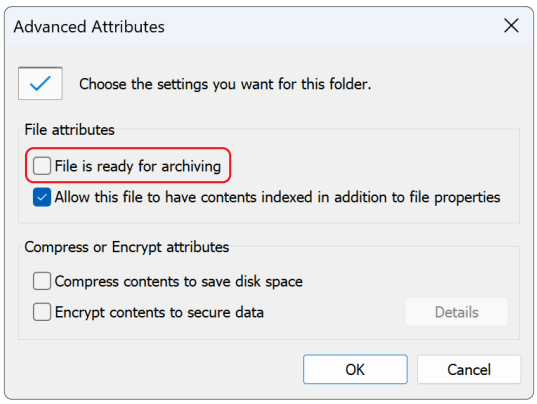
1. Unzip the downloaded zip archive.

2. Properties of the unzipped files may be modified due to the OS's restrictions. Confirm the following two property features.

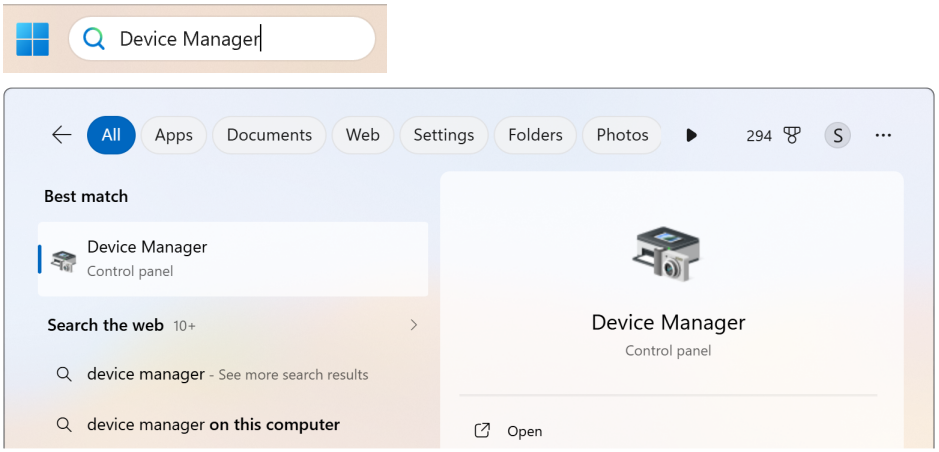
   (1) If a security warning is indicated, check the checkbox to "unblock".



   (2) Press [Advanced] button and confirm status of the checkbox "File is ready for archiving". Check the checkbox if it is not checked, and press [OK].

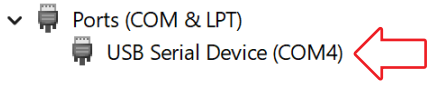


3. Enter “Device Manager” in the Search field on the taskbar of Windows 11 and click on “Device Manager” among the search results.

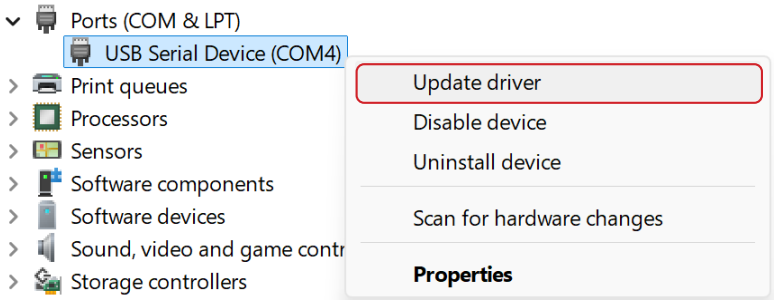


4. Connect the HLS-39USB to the PC.

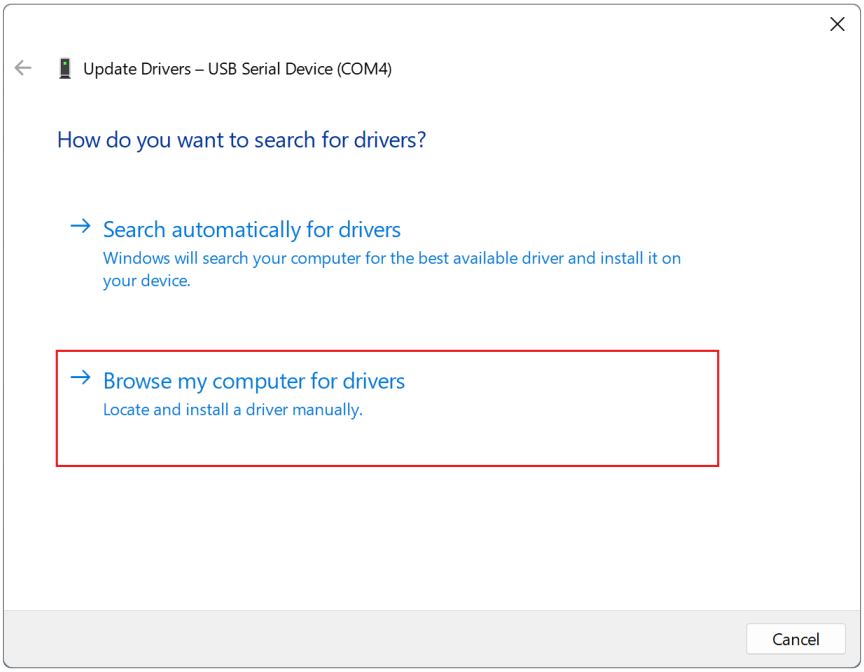
5. Find “USB Serial Device (COMn)” under “Ports (COM and LPT)”.
(“n” following “COM” = Port number)



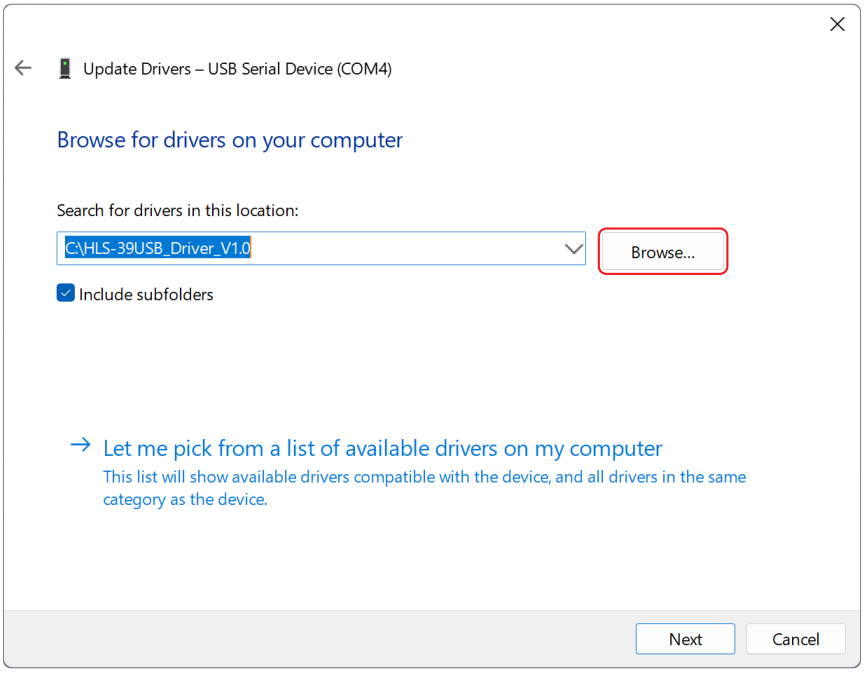
6. Right-click “USB Serial Device (COMn)” and select “Update driver”.



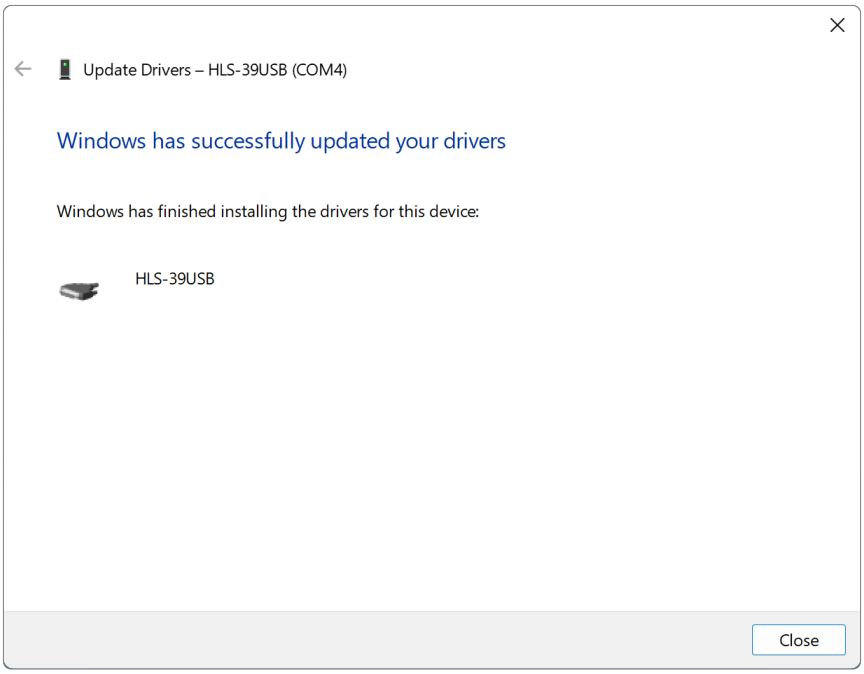
7. Click “Browse my computer for drivers”.



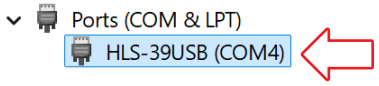
8. Click [Browse] button and select the folder containing the unzipped driver. Press [Next] button.



9. After the driver is installed, the dialog box will show the result.



10. Start “Device Manager” and confirm that “HLS-39USB (COMn)” is listed.



3 Library (DLL)

A library used for controlling the HLS-39USB is available.
Please download it from our website.

URL: https://www.steptechnica.com/en/products/03_board/hls-39usb/hls-39usb.html

3.1 File contents

The downloaded zip archive contains the following data:

HLS-39USB_DLL_Vx.x.zip	Provided file
├ hls39usb.dll	Dynamic library
├ hls39usb.lib	Static library
├ hls39usb.cs	Template file for C#
└ hls39usb.h	Header file of library functions

3.2 Address area

The MKY39-1 has an address area covering a range from 0x0000 through 0x025B.
In addition to this entire area that can be accessed, two special registers are available.

Table 3.1 Registers dedicated to HLS-39USB

Address	Area name	Bit count	R/W	Target function
0x0F00	MKYReset	16	W	Resetting HLS device (MKY39-1)
0x0F02	ReadID	16	R	Reading the unit ID from the setting switch positions

3.2.1 MKYReset (Adrs=0x0F00)

The MKYReset resets the HLS IC (MKY39-1) built in the HLS-39USB.
When any value is written, the reset pin of the IC goes Low to reset the device.

MKYReset (Adrs=0x0F00)

Bit :	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Default :	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W :	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1

After the procedure, read out the CCR (Chip Code Register) to confirm that the device is operating, before resuming operation.

3.2.2 ReadID (Adrs=0x0F02)

The ReadID returns the ID bit states assigned to the No. 1 and 2 positions of the setting switch on the HLS-39USB unit.
The value represents only the status at the startup. The same startup value is returned even if the switch is manipulated later.

ReadID (Adrs=0x0F02)

Bit :	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	–	–	–	–	–	–	–	–	–	–	–	–	–	–	ID1	ID0
Default :	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W :	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

3.3 AutoTrans function

The library of HLS-39USB, just like that of HLS-36USB, has AutoTrans function.
Data in the area from DIN to SSR of the HLS-39USB are read and stored internally in preset intervals. A dedicated API function is used to read out these values.

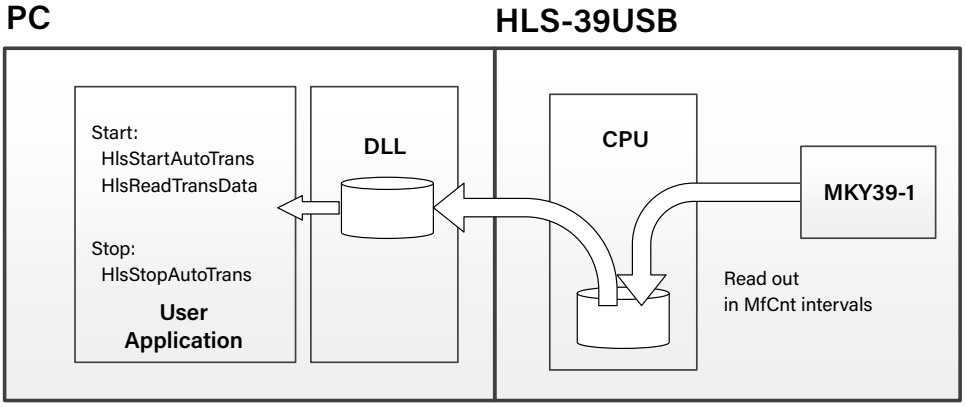


Fig. 3.1 AutoTrans function

The readout interval (MfCnt) can be specified from 1 msec. to 100 msec. in 0.125 msec. increments.

Caution The interval may not be constant depending on the PC's specifications and other applications running at the same time.

3.4 API references

Table 3.2 API functions

API function	Description
HlsGetVersion	Obtains the version number of API
HlsGetLastError	Obtains termination status of the last called API function
HlsCountDevice	Obtains the number of HLS-39USB units connected to the PC
HlsSearchBoard	Obtains the number of HLS-39USB units connected to the PC and the unit ID of each unit
HlsOpenHandle	Opens a handle of HLS-39USB
HlsCloseHandle	Closes a handle obtained by HlsOpenHandle
HlsBoardID	Obtains the unit ID
HlsResetMKY	Commands to reset the MKY39-1
HlsReadWord	Reads 2 bytes of data from a specified address
HlsWriteWord	Writes 2 bytes of data to a specified address
HlsReadData	Reads data of specified word length from a specified address
HlsWriteData	Writes data of specified word length to a specified address
HlsGetFirmwareVersion	Obtains the firmware version number of HLS-39USB
HlsStartAutoTrans	Starts regular updates
HlsStopAutoTrans	Stops regular updates
HlsReadTransData	Obtains the latest data by regular updates

3.4.1 HlsGetVersion

Format

WORD HlsGetVersion(void);

Function

Obtains the version number of library API

Parameter

None

Return value

Version number	Bit 15–8 : Major number Bit 7–4 : Minor number Bit 3–0 : Build number
[Example]	0x0102 → Major number = 1 Minor number = 0 Build number = 2

Error code (obtained by HlsGetLastError() after execution of this API)

HLS_SUCCESS	Successfully completed

Notes

Each part comprising a version number is updated by the following factors:

- Major number: When backward compatibility is no longer maintained by revisions such as specification changes in API.
- Minor number: When backward compatibility is maintained by revisions such as an addition of API functions.
- Build number: When the API has been modified without affecting its specifications, such as correcting malfunctions.

3.4.2 HlsGetLastError

Format

WORD HlsGetLastError(void);

Function

Obtains termination status of the last called API function

Parameter

None

Return value

Returns one of the defined error codes

Notes

The error code is overwritten with the result of a next API call.
Previous error codes are not retained.

Table 3.3 Error codes

Value	Text definition in hls39usb.h	Contents
0	HLS_SUCCESS	Successfully completed
1	HLS_ERR_DEVICENOTEXIST	Device does not exist.
2	HLS_ERR_ALREADYOPENED	Handle has already opened.
3	HLS_ERR_CLOSED	HlsOpenHandle has never been called.
4	HLS_ERR_INVALIDPARAM	Called with invalid parameters
5	HLS_ERR_NORESOURCE	No resource to execute the process
6	HLS_ERR_FAILED	Process failed due to unknown reason
7	HLS_ERR_AUTO_TRANS_ALREADY_START	Regular update has already started.
8	HLS_ERR_AUTO_TRANS_STOP	Regular update has not started.
9	HLS_ERR_USB_TIMEOUT_SUCCESS_STOP_HLS	Timeout occurred during USB communication, and HLS communication successfully stopped.
10	HLS_ERR_USB_TIMEOUT_FAILED_STOP_HLS	Timeout occurred during USB communication, and HLS communication failed to stop.
11	HLS_ERR_REACQUISITION_OF_HANDLE	Handle has not been reacquired.
12	HLS_ERR_NOT_SUPPORT_FIRM_VERSION	Unsupported firmware version
13	HLS_ERR_INVALID_SEQUENCE_NUMBER	Invalid sequence number
99	HLS_NOTCALLYET	HLSAPI has never been called.

3.4.3 HlsCountDevice

Format

short HlsCountDevice(void);

Function

Obtains the number of HLS-39USB units connected to the PC

Parameter

None

Return value

Number of HLS-39USB units	-1 : 5 or more units
	0 : No unit
	1 through 4 : 1 through 4 units

Error code (obtained by HlsGetLastError())

HLS_SUCCESS	Successfully completed

Notes

Five or more units cannot be connected to the PC.

3.4.4 HlsSearchBoard

Format

BOOL HlsSearchBoard(BYTE *unit_num, BYTE *unit_id_list);

Function

Obtains the number of HLS-39USB units connected to the PC and a list of unit IDs

Parameter

*unit_num	Address to a byte type variable in which the number of HLS-39USB units is set
*unit_id_list	Pointer to an array of four byte-data to receive unit IDs

Return value

Returns TRUE (1) in success, FALSE (0) in failure.
The following values are set in the address and the pointer handed in the parameters.

*unit_num	The number of units is set. -1 : 5 or more units 0 : No unit 1 through 4 : 1 through 4 units
*unit_id_list	The unit IDs up to the maximum of four detected units are stored. 0x00 through 0x03 : Unit ID 0x80 : Handle value has already been obtained by HlsOpenHandle. 0xFF : Does not exist, or cannot recognize

Error code (obtained by HlsGetLastError())

HLS_SUCCESS	Successfully completed
HLS_ERR_INVALIDPARAM	NULL has been specified as *unit_num.
HLS_ERR_USB_TIMEOUT_SUCCESS_STOP_HLS	Timeout occurred during USB communication, and HLS communication successfully stopped.
HLS_ERR_USB_TIMEOUT_FAILED_STOP_HLS	Timeout occurred during USB communication, and HLS communication failed to stop.
HLS_ERR_INVALID_SEQUENCE_NUMBER	Invalid sequence number
HLS_ERR_FAILED	Process failed due to unknown reason

Notes

Unit ID, used to identify each of multiple units connected to the PC, can be changed by the setting switch on the unit.
At the maximum of four HLS-39USB units can be identified using this API.
Specify the following byte data array as a parameter.

```
__int8 unit_num;
__int8 unit_id_list[4];
HlsSearchBoard(&unit_num, &unit_id_list[0]);
```

NULL may be specified as *unit_id_list.
If NULL has been specified, only the number of units is returned, with no unit ID information.

[Example] Three HLS-39USB units connected to the PC, with unit IDs set as follows:

1st unit : ID = 0
2nd unit : ID = 2
3rd unit : ID = 1

When HlsSearchBoard is called, the response is as follows:

unit_num = 3;
unit_id_list[0] = 0,
unit_id_list[1] = 2,
unit_id_list[2] = 1,
unit_id_list[3] = 0xFF

3.4.5 HlsOpenHandle

Format

HANDLE HlsOpenHandle(short index_no);

Function

Opens a handle of HLS-39USB

Parameter

index_no	Index number 0 through 3 can be specified.

Return value

Returns '1' or a greater value in success, '-1' (INVALID_HANDLE_VALUE) in failure.

Error code (obtained by HlsGetLastError())

HLS_SUCCESS	Successfully completed
HLS_ERR_ALREADYOPENED	Handle has already opened.
HLS_ERR_DEVICENOTEXIST	Device does not exist.
HLS_ERR_NOT_SUPPORT_FIRM_VERSION	Unsupported firmware version
HLS_ERR_REACQUISITION_OF_HANDLE	Handle has not been reacquired.
HLS_ERR_FAILED	Process failed due to unknown reason

Notes

It is not necessary to execute HlsSearchBoard when only one HLS-39USB is connected.
If multiple units are connected, execute HlsSearchBoard in advance to determine the index_no of the target unit specified in HlsOpenHandle.
An index_no is not a physical unit ID, but that returned in the response array for *unit_id_list by HlsSearchBoard function.

[Example] Result of calling HlsSearchBoard as in the following:

unit_num = 3;
unit_id_list[0] = 0,
unit_id_list[1] = 2,
unit_id_list[2] = 1,
unit_id_list[3] = 0xFF

Specify '1' as index_no to access the HLS-39USB unit with the unit ID = 2.

3.4.6 HlsCloseHandle

Format

BOOL HlsCloseHandle(HANDLE HLSHandle);

Function

Closes a handle obtained by HlsOpenHandle

Parameter

HLSHandle	Handle value of HLS-39USB

Return value

Returns TRUE (1) in success, FALSE (0) in failure.

Error code (obtained by HlsGetLastError())

HLS_SUCCESS	Successfully completed
HLS_ERR_INVALIDPARAM	Specified HLSHandle is invalid.
HLS_ERR_INVALID_SEQUENCE_NUMBER	Invalid sequence number
HLS_ERR_FAILED	Process failed due to unknown reason

Notes

Regular update, if it is running, is also terminated.

3.4.7 HlsBoardID

Format

short HlsBoardID(HANDLE HLSHandle);

Function

Obtains the unit ID of the HLS-39USB

Parameter

HLSHandle Handle value of HLS-39USB

Return value

Returns the unit ID (0 through 3) in success, '-1' in failure.

Error code (obtained by HlsGetLastError())

HLS_SUCCESS	Successfully completed
HLS_ERR_INVALIDPARAM	Specified HLSHandle is invalid.
HLS_ERR_USB_TIMEOUT_SUCCESS_STOP_HLS	Timeout occurred during USB communication, and HLS communication successfully stopped.
HLS_ERR_USB_TIMEOUT_FAILED_STOP_HLS	Timeout occurred during USB communication, and HLS communication failed to stop.
HLS_ERR_REACQUISITION_OF_HANDLE	Handle has not been reacquired.
HLS_ERR_INVALID_SEQUENCE_NUMBER	Invalid sequence number
HLS_ERR_FAILED	Process failed due to unknown reason

3.4.8 HlsResetMKY

Format

BOOL HlsResetMKY(HANDLE HLSHandle);

Function

Commands to reset the HLS IC (MKY39-1) mounted on the unit

Parameter

HLSHandle	Handle value of HLS-39USB

Return value

Returns TRUE (1) in success, FALSE (0) in failure.

Error code (obtained by HlsGetLastError())

HLS_SUCCESS	Successfully completed
HLS_ERR_INVALIDPARAM	Specified HLSHandle is invalid.
HLS_ERR_USB_TIMEOUT_SUCCESS_STOP_HLS	Timeout occurred during USB communication, and HLS communication successfully stopped.
HLS_ERR_USB_TIMEOUT_FAILED_STOP_HLS	Timeout occurred during USB communication, and HLS communication failed to stop.
HLS_ERR_REACQUISITION_OF_HANDLE	Handle has not been reacquired.
HLS_ERR_INVALID_SEQUENCE_NUMBER	Invalid sequence number
HLS_ERR_FAILED	Process failed due to unknown reason

3.4.9 HlsReadWord

Format

BOOL HlsReadWord(HANDLE HLSHandle, WORD Adrs, WORD *Data);

Function

Reads 2 bytes of data from a specified address

Parameter

HLSHandle	Handle value of HLS-39USB
Adrs	Target address to read
*Data	Address to store read data

Return value

Returns TRUE (1) in success, FALSE (0) in failure.

Error code (obtained by HlsGetLastError())

HLS_SUCCESS	Successfully completed
HLS_ERR_INVALIDPARAM	Specified HLSHandle is invalid. Adrs not a multiple of 2 Adrs out of range NULL has been specified for *Data.
HLS_ERR_USB_TIMEOUT_SUCCESS_STOP_HLS	Timeout occurred during USB communication, and HLS communication successfully stopped.
HLS_ERR_USB_TIMEOUT_FAILED_STOP_HLS	Timeout occurred during USB communication, and HLS communication failed to stop.
HLS_ERR_REACQUISITION_OF_HANDLE	Handle has not been reacquired.
HLS_ERR_INVALID_SEQUENCE_NUMBER	Invalid sequence number
HLS_ERR_FAILED	Process failed due to unknown reason

Notes

Accessible data is in 2-byte form. An address other than a multiple of 2 results in error.
Adrs can be specified from 0x0000 through 0x0F02.
(The library for the HLS-36USB had the same range.)

3.4.10 HlsWriteWord

Format

BOOL HlsWriteWord(HANDLE HLSHandle, WORD Adrs, WORD Data);

Function

Writes 2 bytes of data to a specified address

Parameter

HLSHandle	Handle value of HLS-39USB
Adrs	Target address to write
Data	Write data

Return value

Returns TRUE (1) in success, FALSE (0) in failure.

Error code (obtained by HlsGetLastError())

HLS_SUCCESS	Successfully completed
HLS_ERR_INVALIDPARAM	Specified HLSHandle is invalid. Adrs not a multiple of 2 Adrs out of range
HLS_ERR_USB_TIMEOUT_SUCCESS_STOP_HLS	Timeout occurred during USB communication, and HLS communication successfully stopped.
HLS_ERR_USB_TIMEOUT_FAILED_STOP_HLS	Timeout occurred during USB communication, and HLS communication failed to stop.
HLS_ERR_REACQUISITION_OF_HANDLE	Handle has not been reacquired.
HLS_ERR_INVALID_SEQUENCE_NUMBER	Invalid sequence number
HLS_ERR_FAILED	Process failed due to unknown reason

Notes

Accessible data is in 2-byte form. An address other than a multiple of 2 results in error.
Adrs can be specified from 0x0000 through 0x0F02.
(The library for the HLS-36USB had the same range.)

3.4.11 HlsReadData

Format

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

Function

Reads data of specified word length from a specified address

Parameter

HLSHandle	Handle value of HLS-39USB
Adrs	Target address to read
WordLen	Word length to read
*Data	Address to store read data

Return value

Returns TRUE (1) in success, FALSE (0) in failure.

Error code (obtained by HlsGetLastError())

HLS_SUCCESS	Successfully completed
HLS_ERR_INVALIDPARAM	Specified HLSHandle is invalid. Adrs not a multiple of 2 Adrs out of range WordLen out of range NULL has been specified for *Data.
HLS_ERR_USB_TIMEOUT_SUCCESS_STOP_HLS	Timeout occurred during USB communication, and HLS communication successfully stopped.
HLS_ERR_USB_TIMEOUT_FAILED_STOP_HLS	Timeout occurred during USB communication, and HLS communication failed to stop.
HLS_ERR_REACQUISITION_OF_HANDLE	Handle has not been reacquired.
HLS_ERR_INVALID_SEQUENCE_NUMBER	Invalid sequence number
HLS_ERR_FAILED	Process failed due to unknown reason

Notes

Accessible data is in word (16-bit) form. An address other than a multiple of 2 results in error.
Adrs can be specified from 0x0000 through 0x02FE.
The maximum size of WordLen is 0x0180 (0x0300 bytes).

3.4.12 HlsWriteData

Format

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

Function

Writes data of specified word length to a specified address

Parameter

HLSHandle	Handle value of HLS-39USB
Adrs	Target address to write
WordLen	Word length to write
*Data	Address to store write data

Return value

Returns TRUE (1) in success, FALSE (0) in failure.

Error code (obtained by HlsGetLastError())

HLS_SUCCESS	Successfully completed
HLS_ERR_INVALIDPARAM	Specified HLSHandle is invalid. Adrs not a multiple of 2 Adrs out of range WordLen out of range NULL has been specified for *Data.
HLS_ERR_USB_TIMEOUT_SUCCESS_STOP_HLS	Timeout occurred during USB communication, and HLS communication successfully stopped.
HLS_ERR_USB_TIMEOUT_FAILED_STOP_HLS	Timeout occurred during USB communication, and HLS communication failed to stop.
HLS_ERR_REACQUISITION_OF_HANDLE	Handle has not been reacquired.
HLS_ERR_INVALID_SEQUENCE_NUMBER	Invalid sequence number
HLS_ERR_FAILED	Process failed due to unknown reason

Notes

Accessible data is in word (16-bit) form. An address other than a multiple of 2 results in error.
Adrs can be specified from 0x0000 through 0x02FE.
The maximum size of WordLen is 0x0180 (0x0300 bytes).

3.4.13 HlsGetFirmwareVersion

Format

short HlsGetFirmwareVersion(HANDLE HLSHandle);

Function

Obtains the firmware version number of HLS-39USB

Parameter

HLSHandle	Handle value of HLS-39USB

Return value

Version number	Bit 15–8 : Major number Bit 7–4 : Minor number Bit 3–0 : Build number
[Example]	0x0104 → Major number = 1 Minor number = 0 Build number = 4

Error code (obtained by HlsGetLastError())

HLS_SUCCESS	Successfully completed
HLS_ERR_INVALIDPARAM	Specified HLSHandle is invalid.
HLS_ERR_USB_TIMEOUT_SUCCESS_STOP_HLS	Timeout occurred during USB communication, and HLS communication successfully stopped.
HLS_ERR_USB_TIMEOUT_FAILED_STOP_HLS	Timeout occurred during USB communication, and HLS communication failed to stop.
HLS_ERR_REACQUISITION_OF_HANDLE	Handle has not been reacquired.
HLS_ERR_INVALID_SEQUENCE_NUMBER	Invalid sequence number
HLS_ERR_FAILED	Process failed due to unknown reason

Notes

Each part comprising a version number is updated by the following factors:

- Major number: When backward compatibility is no longer maintained by revisions such as specification changes in API.
- Minor number: When backward compatibility is maintained by revisions such as an addition of API functions.
- Build number: When the API has been modified without affecting its specifications, such as correcting malfunctions.

3.4.14 HlsStartAutoTrans

Format

BOOL HlsStartAutoTrans(HANDLE HLSHandle, WORD MfCnt);

Function

Starts regular updates of the data from DIN to SSR of MKY39-1

Parameter

HLSHandle	Handle value of HLS-39USB
MfCnt	Update interval

Return value

Returns TRUE (1) in success, FALSE (0) in failure.

Error code (obtained by HlsGetLastError())

HLS_SUCCESS	Successfully completed
HLS_ERR_INVALIDPARAM	Specified HLSHandle is invalid. MfCnt out of range
HLS_ERR_AUTO_TRANS_ALREADY_START	Regular update has already started.
HLS_ERR_USB_TIMEOUT_SUCCESS_STOP_HLS	Timeout occurred during USB communication, and HLS communication successfully stopped.
HLS_ERR_USB_TIMEOUT_FAILED_STOP_HLS	Timeout occurred during USB communication, and HLS communication failed to stop.
HLS_ERR_REACQUISITION_OF_HANDLE	Handle has not been reacquired.
HLS_ERR_INVALID_SEQUENCE_NUMBER	Invalid sequence number
HLS_ERR_FAILED	Process failed due to unknown reason

Notes

MfCnt can be specified from 1 msec. to 100 msec. in 125 μsec. increments, i.e. from 8 through 800 as setting value.
Be aware that regular updates may not be performed in the specified intervals depending on the PC's specifications and other applications running at the same time.
Regular update data can be obtained by using HlsReadTransData after executing this API.

3.4.15 HlsStopAutoTrans

Format

BOOL HlsStopAutoTrans(HANDLE HLSHandle);

Function

Stops regular updates of the data from DIN to SSR of MKY39-1

Parameter

HLSHandle Handle value of HLS-39USB

Return value

Returns TRUE (1) in success, FALSE (0) in failure.

Error code (obtained by HlsGetLastError())

HLS_SUCCESS	Successfully completed
HLS_ERR_INVALIDPARAM	Specified HLSHandle is invalid.
HLS_ERR_AUTO_TRANS_STOP	Regular update has not started.
HLS_ERR_USB_TIMEOUT_SUCCESS_STOP_HLS	Timeout occurred during USB communication, and HLS communication successfully stopped.
HLS_ERR_USB_TIMEOUT_FAILED_STOP_HLS	Timeout occurred during USB communication, and HLS communication failed to stop.
HLS_ERR_REACQUISITION_OF_HANDLE	Handle has not been reacquired.
HLS_ERR_INVALID_SEQUENCE_NUMBER	Invalid sequence number
HLS_ERR_FAILED	Process failed due to unknown reason

Notes

Calling HlsCloseHandle while regular update is running will stop it automatically without calling this API.

3.4.16 HlsReadTransData

Format

BOOL HlsReadTransData(HANDLE HLSHandle, void *Data);

Function

Obtains the latest data from DIN to SSR of MKY39-1 read out by regular updates

Parameter

HLSHandle	Handle value of HLS-39USB
*Data	Address to store the data (156 bytes)

Return value

Returns TRUE (1) in success, FALSE (0) in failure.

Error code (obtained by HlsGetLastError())

HLS_SUCCESS	Successfully completed
HLS_ERR_INVALIDPARAM	Specified HLSHandle is invalid. NULL has been specified for *Data.
HLS_ERR_AUTO_TRANS_STOP	Regular update is stopped.
HLS_ERR_REACQUISITION_OF_HANDLE	Handle has not been reacquired.
HLS_ERR_FAILED	Process failed due to unknown reason

Notes

The HLS-39USB reads out internally the following data ranging from DIN to SSR in regular intervals when HlsStartAutoTrans has been executed.

DIN : 128 bytes
LSR : 8 bytes
LSR3 : 8 bytes
C1SR : 4 bytes (valid data: 2 bytes)
C2SR : 4 bytes (valid data: 2 bytes)
SSR : 4 bytes (valid data: 2 bytes)

This API is used to retrieve the last updated data.
An error will be returned if the API is called while the regular update is stopped.
Be aware that the API does not access MKY39-1 directly. If you want to make sure to have the latest data, use HlsReadWord or HlsReadData.

3.5 Limitations

The library for the HLS-39USB has limitations similar to that for the HLS-36USB as explained in the following. Please be aware of these limitations when using API.

■ Multi-thread use

API functions cannot be used from multiple threads at once.
When configuring your application for a multi-threaded setup, please ensure that simultaneous calls do not occur.

■ Timeout in USB communication

In this API, the maximum wait time (timeout) for data transmission and reception with the HLS-39USB is set to 1 second.
Depending on the system environment, data transmission/reception may not complete before the timeout period expires (timeout).

In case of a timeout, the API function returns an error.
Then you can call HlsGetLastError function to get one of the following two error codes:

- HLS_ERR_USB_TIMEOUT_SUCCESS_STOP_HLS (9)
  When a timeout occurs, the API internally stops HLS network and regular updates.
  The error code is given when it has successfully stopped the HLS network and the regular updates.
- HLS_ERR_USB_TIMEOUT_FAILED_STOP_HLS (10)
  The error code is given when it has failed to stop the network and the regular updates.

After a timeout, close the currently used handle by HlsCloseHandle function, and newly acquire a handle by HlsOpenHandle function.
Until a handle is reacquired, any API function other than HlsGetVersion, HlsCountDevice, HlsSearchBoard, HlsGetLastError, HlsOpenHandle and HlsCloseHandle returns an error.
In this instance, HlsGetLastError function returns HLS_ERR_REACQUISITION_OF_HANDLE (11).

4 MKY36-compatible mode

The HLS-39USB has the MKY36-compatible mode to simulate the functions of HLS-36USB.

The HLS-39USB starts functioning as the HLS-36USB when the power is turned on with the fourth position of the setting switch set to ON.
The HLS-36USB driver is necessary to operate in this setting. Please refer to the manual for the HLS-36USB and install the driver.

Many user applications ready for the HLS-36USB can be used in the MKY36-compatible mode. However, MKY39-1 and MKY36, as individual ICs, are not compatible with each other in certain points.

This chapter explains these differences.

4.1 MKY36 registers

The HLS-39USB is equipped with MKY39-1 as its control IC, while the HLS-36USB is with MKY36. These two devices are not completely compatible.
Please refer to “Appendix 3: Bit layout and operations of registers” in [**MKY39-1 / MKY39-2 Reference Manual**] for details of these differences.

Table 4.2 shows a list of MKY36 registers which can be accessed in the compatible mode of HLS-39USB.
There are four levels of compatibility shown in the table:

A .....Hardware compatibility is achieved by simple register address conversion.
B .....Functional compatibility is not achieved but the HLS-39USB firmware compensates.
C .....Certain functionalities remain incompatible.
D .....Function is not available.

The following functions of MKY36 are changed in the MKY39-1:

◆ DataRenewal function
The MKY39-1 has no DataRenewal function.
Those registers and flags related to the function are not supported.

◆ INT1R
The MKY39-1 has only one interrupt pin.
Those registers related to the function are not supported.

◆ Data hazard protection function
The MKY39-1 has an SPI interface, which ensures by design that no data hazard occurs during reading and writing.
The HPR is not supported due to this reason.

◆ CTL and C1 through C7
The MKY36 uses the control words to send commands and to perform CHK1/CHK2 checking functions.
The MKY39-1, on the other hand, uses the CMDCR and CMDSR to send commands and to obtain the results.
The HLS-39USB emulates the CTL area but does not support C1 through C7.

Table 4.1 CTL emulation

CTL	HLS-36USB	HLS-39USB
[15]	Not used	Not used
[14 : 12]	Number of consecutive nonresponse	0x00 : Both LSR and LSR3 are “1” 0x14 : Only LSR is “0” 0x7C : LSR3 is “0”
[11]	RX-CHK2	Corresponding bit of LSR3
[10]	RX-CHK1	Corresponding bit of LSR
[9]	DR	Not used
[8]	DREQ	Not used
[7]	Not used	Not used
[6]	Not used	Not used
[5]	ARF	Not used
[4]	ACR	Not used
[3 : 0]	Command	Not used

◆ ESF (bit 10 of SSR)
The MKY39-1 does not count the number of consecutive nonresponse which is in the CTL of MKY36. The ESF flag, consequently, is not supported.

Table 4.2 MKY36 register compatibility

A : Hardware compatibility **B** : Software compatibility **C** : Partial incompatibility in function **D** : Not supported

Address	Area name	Bit count	Target function	Compatibility level				Notes
				A	B	C	D	
0x0000	CTL	2*64	Control	✓				
0x0080	Do	2*64	Do pin output data	✓				
0x0100	Di	2*64	Di pin input data	✓				
0x0180	C1	2*64	(MKY34) Counter ch1 value				✓	
0x0200	C2	2*64	(MKY34) Counter ch2 value				✓	
0x0280	C3	2*64	(MKY34) Counter ch3 value				✓	
0x0300	C4	2*64	(MKY34) Counter ch4 value				✓	
0x0380	C5	2*64	(MKY34) Counter ch5 value				✓	
0x0400	C6	2*64	(MKY34) Counter ch6 value				✓	
0x0480	C7	2*64	(MKY34) Serial ID value				✓	
0x0500	DRC	2*64	Di Data Renewal				✓	
0x0580	SCR	16	System	✓				
0x0582	SSR	16	System			✓		
0x0584	HPR	16	Data hazard protection				✓	
0x0586	INT0R	16	Interrupt control		✓			
0x0588	INT1R	16	Interrupt control				✓	
0x058A	C1CR	16	System		✓			
0x058C	C2CR	16	System		✓			
0x058E	BCR	16	System	✓				
0x0590	CCR	4	System	✓				

Revision History

Version No.	Date	Page	Contents
100	November 2026	---	Issued the first edition

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 the document.
2. Technical information in this document, such as explanations and circuit examples, are references for the product's adequate use in accordance with your specific application.
3. When actually using this product, always fully evaluate the entire system under your sole responsibility, according to the design purpose based on considerations of peripheral environments. We assume no responsibility for any incompatibility between the product and your system.
4. 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.
5. 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.
6. This product is not designed for use in critical applications, such as life support systems. Contact us when considering such applications.
7. 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.

HLS (MKY39-1) USB Unit

HLS-39USB Reference Manual

Document No.: P10996-100
Issued: November 2026

HLS and CUnet Products are developed and manufactured by

StepTechnica Co., Ltd.
1-1-15 Tateno, Higashiyamato-shi, Tokyo 207-0021
https://www.steptechnica.com/
info@steptechnica.com

©2026- StepTechnica Co., Ltd.