P10992-100

CUnet Family

CU-49USB

REFERENCE MANUAL

https://www.steptechnica.com/

CUnet

CUnet (MKY49) USB Unit

CU-49USB Reference Manual

The CU-49USB, equipped with the CUnet IC MKY49, is designed to control the CUnet through USB communication.

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

©2026- StepTechnica Co., Ltd.

Preface

This manual describes the software for the CU-49USB equipped with the CUnet IC MKY49. Before reading this manual, be sure to read [**CUnet Introduction Guide**].

■ Target Readers

This manual is intended for:

- Those who build a CUnet network for the first time
- Those who use StepTechnica ICs for the first time to build a CUnet network
- Those who use the CU-49USB to evaluate CUnet
- Those who have experience of using CUnet with the CU-43USB

■ Prerequisites

This manual assumes that you are familiar with:

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

■ Related Manuals

- CU-49USB Data Sheet
- CUnet Introduction Guide
- CUnet Technical Guide
- MKY49 Data Sheet
- MKY49 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 CUnet 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=0x3F00) 3-2
3.2.2 ReadID (Adrs=0x3F02) 3-2
3.3 AutoTrans function 3-3
3.4 API references 3-4
3.4.1 CubGetVersion 3-5
3.4.2 CubGetLastError 3-6
3.4.3 CubCountDevice 3-7
3.4.4 CubSearchBoard 3-8
3.4.5 CubOpenHandle 3-10
3.4.6 CubCloseHandle 3-11
3.4.7 CubBoardID 3-12
3.4.8 CubResetMKY 3-13
3.4.9 CubReadWord 3-14
3.4.10 CubWriteWord 3-15
3.4.11 CubReadData 3-16
3.4.12 CubWriteData 3-17
3.4.13 CubReadProtect 3-18
3.4.14 CubWriteProtect 3-19
3.4.15 CubGetFirmwareVersion 3-20
3.4.16 CubStartAutoTrans 3-21
3.4.17 CubStopAutoTrans 3-22
3.4.18 CubReadGM 3-23
3.4.19 CubReadMFR 3-24
3.5 Limitations 3-25

4 MKY43-compatible mode

4.1 MKY43 registers 4-2

1 General description

The CUnet can be operated by user applications on a PC connected to the CU-49USB with a USB cable, using the driver and library explained in this manual.

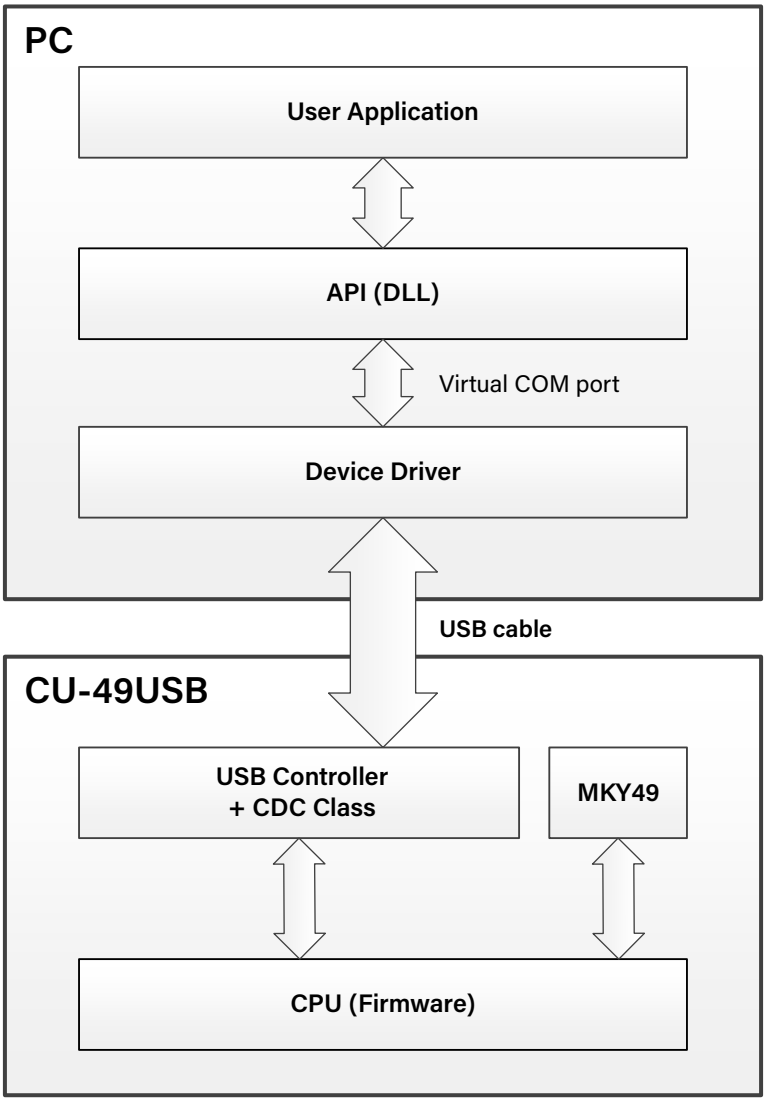


Fig. 1.1 Software components

1.1 Operating environment

- PC with a free USB port for connecting with CU-49USB
  (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 CU-49USB. 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 CU-49USB.



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 CU-49USB is used to connect a PC to CUnet network.
Users can use the MKY49 functions easily for evaluation purposes when building a CUnet system and for development of Sub-Stations, by accessing Windows driver and library supplied separately by StepTechnica.
The MKY43-compatible mode is available so that the environment used with the CU-43USB can be transferred to this product for continuous use.



Certain functions are no longer supported by CU-49USB. For details, refer to "4 MKY43-compatible mode".

2 Driver

Windows 64-bit driver is available for CU-49USB.
Please download it from our website.

URL: https://www.steptechnica.com/en/products/03_board/cu-49usb/cu-49usb.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:

CU-49USB_Driver_Vx.x.zip	Provided file
├ cu49usb.cat	Catalog file
└ cu49usb.inf	Setup information file

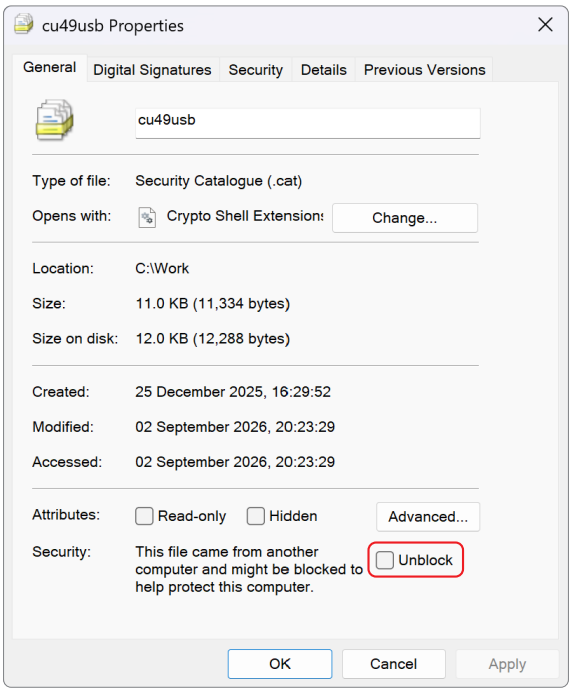
2.2 Installing driver

Follow the instructions below to install the driver which allows your PC to recognize the CU-49USB.

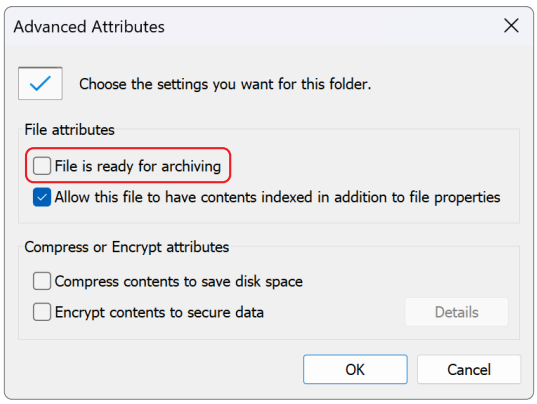
1. Unzip the downloaded zip archive.

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

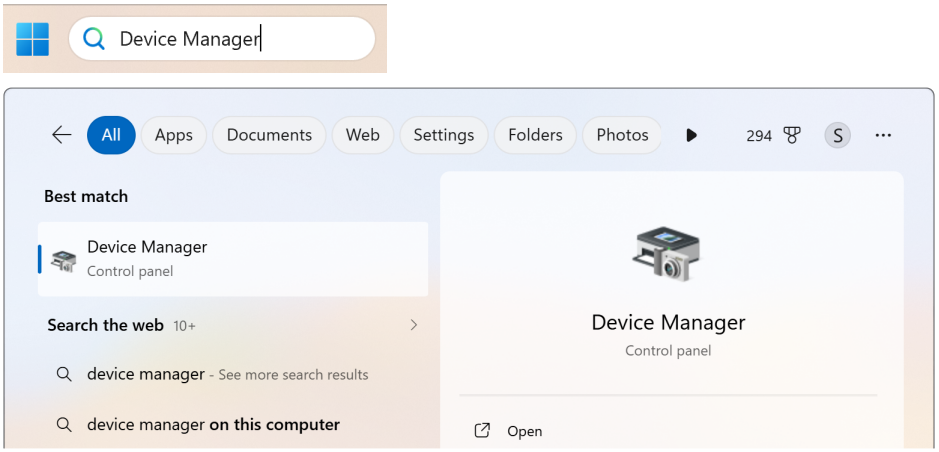
   (1) If a security warning is indicated, check the checkbox to "unblock".



   (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].

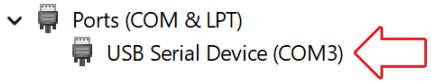


3. Enter “Device Manager” in the Search field on the taskbar of Windows 11 and click on “Device Manager” among the search results.

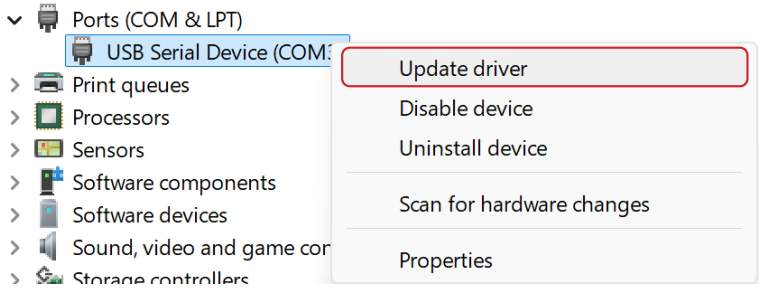


4. Connect the CU-49USB to the PC.

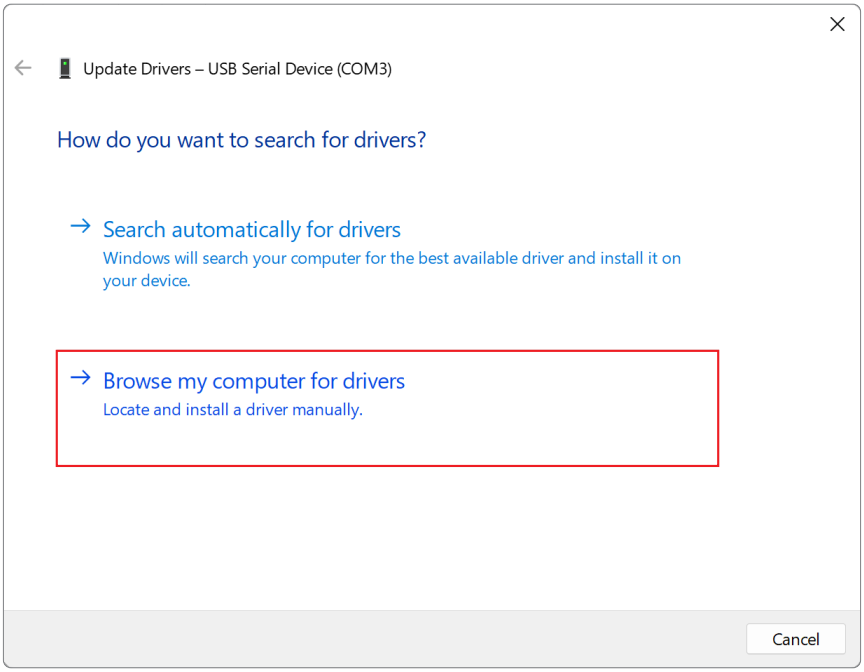
5. Find “USB Serial Device (COMn)” under “Ports (COM and LPT)”.
(“n” following “COM” = Port number)



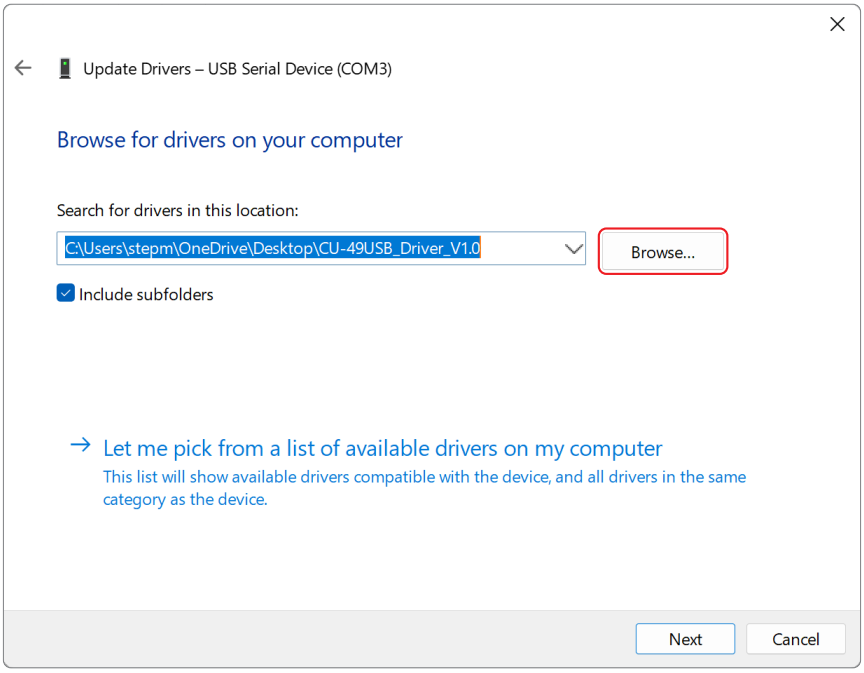
6. Choose “USB Serial Device (COMn)” and right-click on the mouse to select “Update driver”.



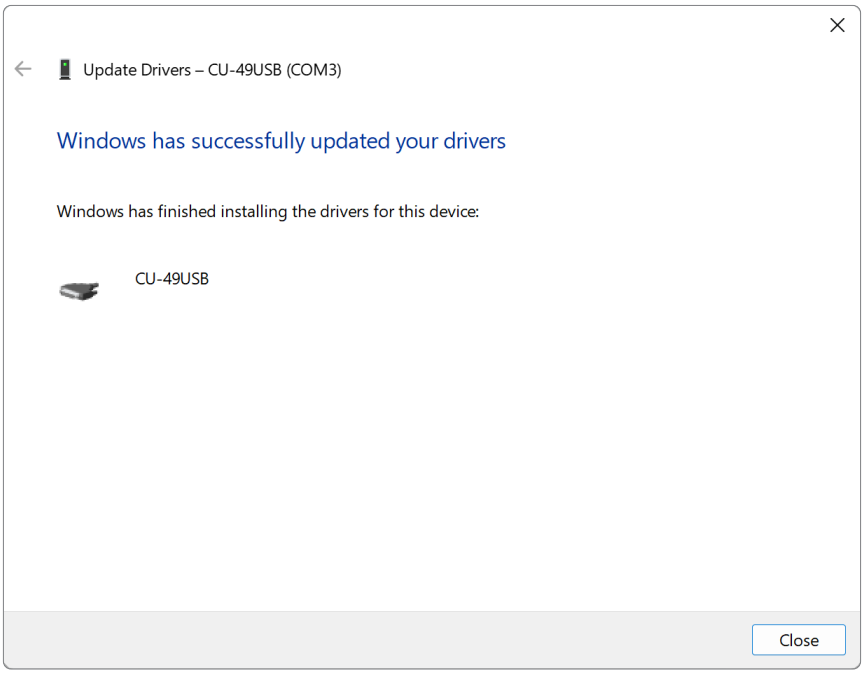
7. Click “Browse my computer for drivers”.



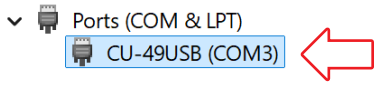
8. Click [Browse] button and select the folder containing the unzipped driver. Press [Next] button.



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



10. Start “Device Manager” and confirm that “CU-49USB (COMn) is listed.



3 Library (DLL)

A library used for controlling the CU-49USB is available.
Please download it from our website.

URL: https://www.steptechnica.com/en/products/03_board/cu-49usb/cu-49usb.html

3.1 File contents

The downloaded zip archive contains the following data:

CU-49USB_DLL_Vx.x.zip	Provided file
├ cub49usb.dll	Dynamic library
├ cub49usb.lib	Static library
├ cub49usb.cs	Template file for C#
└ cub49usb.h	Header file of library functions

3.2 Address area

The MKY49 has an address area covering a range from 0x0000 through 0x23FF.
In addition to this entire area that can be accessed, two special registers are available.

Table 3.1 Registers dedicated to CU-49USB

Address	Area name	Bit count	R/W	Target function
0x3F00	MKYReset	16	W	Resetting CUnet device (MKY49)
0x3F02	ReadID	16	R	Reading the unit ID from the setting switch positions

3.2.1 MKYReset (Adrs=0x3F00)

The MKYReset resets the CUnet IC (MKY49) built in the CU-49USB.
When any value is written, the reset pin of the IC goes to Low state to reset the device.

MKYReset (Adrs=0x3F00)

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=0x3F02)

The ReadID returns the ID bit states assigned to the No. 1 and 2 positions of the setting switch on the CU-49USB 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=0x3F02)

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 CU-49USB, just like that of CU-43USB, has AutoTrans function.
Data in the entire Global Memory area and the MFR (Member Flag Register) are read and stored internally in preset intervals. Dedicated API functions are used to read out these values.

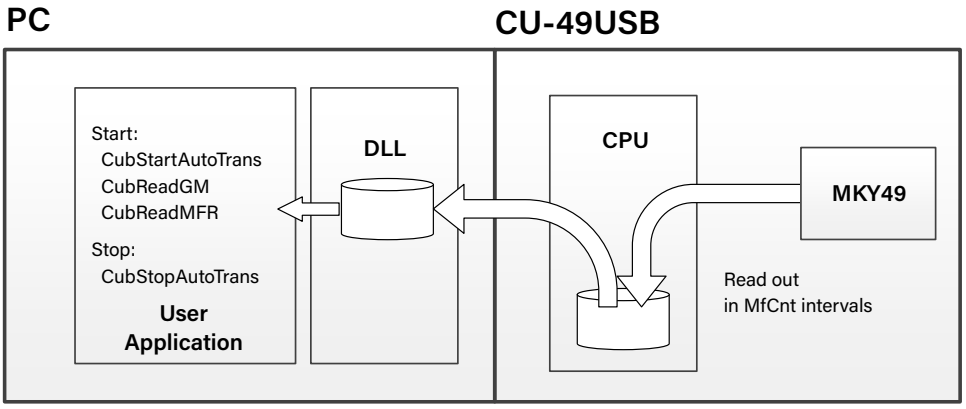


Fig. 3.1 AutoTrans function

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
CubGetVersion	Obtains the version number of API
CubGetLastError	Obtains termination status of the last called API function
CubCountDevice	Obtains the number of CU-49USB units connected to the PC
CubSearchBoard	Obtains the number of CU-49USB units connected to the PC and the unit ID of each unit
CubOpenHandle	Opens a handle of CU-49USB
CubCloseHandle	Closes a handle obtained by CubOpenHandle
CubBoardID	Obtains the unit ID
CubResetMKY	Commands to reset the MKY49
CubReadWord	Reads 2 bytes of data from a specified address
CubWriteWord	Writes 2 bytes of data to a specified address
CubReadData	Reads data of specified word length from a specified address
CubWriteData	Writes data of specified word length to a specified address
CubReadProtect	Reads data from Global Memory
CubWriteProtect	Writes data to Global Memory
CubGetFirmwareVersion	Obtains the firmware version number of CU-49USB
CubStartAutoTrans	Starts regular updates
CubStopAutoTrans	Stops regular updates
CubReadGM	Obtains the latest data in the entire Global Memory area by regular updates
CubReadMFR	Obtains the latest data of MFR by regular updates

3.4.1 CubGetVersion

Format

WORD CubGetVersion(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 CubGetLastError() after execution of this API)

CUB_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 CubGetLastError

Format

WORD CubGetLastError(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 cub49usb.h	Contents
0	CUB_SUCCESS	Successfully completed
1	CUB_ERR_DEVICENOTEXIST	Device does not exist.
2	CUB_ERR_ALREADYOPENED	Handle has already opened.
3	CUB_ERR_CLOSED	CubOpenHandle has never been called.
4	CUB_ERR_INVALIDPARAM	Called with invalid parameters
5	CUB_ERR_NORESOURCE	No resource to execute the process
6	CUB_ERR_FAILED	Process failed due to unknown reason
7	CUB_ERR_AUTO_TRANS_ALREADY_START	Regular update has already started.
8	CUB_ERR_AUTO_TRANS_STOP	Regular update has not started.
9	CUB_ERR_USB_TIMEOUT_SUCCESS_STOP_CUNET	Timeout occurred during USB communication, and CUnet communication successfully stopped.
10	CUB_ERR_USB_TIMEOUT_FAILED_STOP_CUNET	Timeout occurred during USB communication, and CUnet communication failed to stop.
11	CUB_ERR_REACQUISITION_OF_HANDLE	Handle has not been reacquired.
12	CUB_ERR_NOT_SUPPORT_FIRM_VERSION	Unsupported firmware version
13	CUB_ERR_INVALID_SEQUENCE_NUMBER	Invalid sequence number
99	CUB_NOTCALLYET	CUBAPI has never been called.

3.4.3 CubCountDevice

Format

short CubCountDevice(void);

Function

Obtains the number of CU-49USB units connected to the PC

Parameter

None

Return value

Number of CU-49USB units	-1 : 5 or more units
	0 : No unit
	1 through 4 : 1 through 4 units

Error code (obtained by CubGetLastError())

CUB_SUCCESS	Successfully completed

Notes

Five or more units cannot be connected to the PC.

3.4.4 CubSearchBoard

Format

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

Function

Obtains the number of CU-49USB 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 CU-49USB 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 CubOpenHandle. 0xFF : Does not exist, or cannot recognize

Error code (obtained by CubGetLastError())

CUB_SUCCESS	Successfully completed
CUB_ERR_INVALIDPARAM	NULL has been specified as *unit_num.
CUB_ERR_USB_TIMEOUT_SUCCESS_STOP_CUNET	Timeout occurred during USB communication, and CUnet communication successfully stopped.
CUB_ERR_USB_TIMEOUT_FAILED_STOP_CUNET	Timeout occurred during USB communication, and CUnet communication failed to stop.
CUB_ERR_INVALID_SEQUENCE_NUMBER	Invalid sequence number
CUB_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 CU-49USB units can be identified using this API.
Specify the following byte data array as a parameter.

```
__int8 unit_num;
__int8 unit_id_list[4];
CubSearchBoard(&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 CU-49USB units connected to the PC, with unit IDs set as follows:

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

When CubSearchBoard 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 CubOpenHandle

Format

HANDLE CubOpenHandle(short index_no);

Function

Opens a handle of CU-49USB

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 CubGetLastError())

CUB_SUCCESS	Successfully completed
CUB_ERR_ALREADYOPENED	Handle has already opened.
CUB_ERR_DEVICENOTEXIST	Device does not exist.
CUB_ERR_NOT_SUPPORT_FIRM_VERSION	Unsupported firmware version
CUB_ERR_REACQUISITION_OF_HANDLE	Handle has not been reacquired.
CUB_ERR_FAILED	Process failed due to unknown reason

Notes

It is not necessary to execute CubSearchBoard when only one CU-49USB is connected.
If multiple units are connected, execute CubSearchBoard in advance to determine the index_no of the target unit specified in CubOpenHandle.
An index_no is not a physical unit ID, but that returned in the response array for *unit_id_list by CubSearchBoard function.

[Example] Result of calling CubSearchBoard 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 CU-49USB unit with the unit ID = 2.

3.4.6 CubCloseHandle

Format

BOOL CubCloseHandle(HANDLE CUHandle);

Function

Closes a handle obtained by CubOpenHandle

Parameter

CUHandle	Handle value of CU-49USB

Return value

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

Error code (obtained by CubGetLastError())

CUB_SUCCESS	Successfully completed
CUB_ERR_INVALIDPARAM	Specified CUHandle is invalid.
CUB_ERR_INVALID_SEQUENCE_NUMBER	Invalid sequence number
CUB_ERR_FAILED	Process failed due to unknown reason

Notes

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

3.4.7 CubBoardID

Format

short CubBoardID(HANDLE CUHandle);

Function

Obtains the unit ID of the CU-49USB

Parameter

CUHandle	Handle value of CU-49USB

Return value

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

Error code (obtained by CubGetLastError())

CUB_SUCCESS	Successfully completed
CUB_ERR_INVALIDPARAM	Specified CUHandle is invalid.
CUB_ERR_USB_TIMEOUT_SUCCESS_STOP_CUNET	Timeout occurred during USB communication, and CUnet communication successfully stopped.
CUB_ERR_USB_TIMEOUT_FAILED_STOP_CUNET	Timeout occurred during USB communication, and CUnet communication failed to stop.
CUB_ERR_REACQUISITION_OF_HANDLE	Handle has not been reacquired.
CUB_ERR_INVALID_SEQUENCE_NUMBER	Invalid sequence number
CUB_ERR_FAILED	Process failed due to unknown reason

3.4.8 CubResetMKY

Format

BOOL CubResetMKY(HANDLE CUHandle);

Function

Commands to reset the CUnet IC (MKY49) mounted on the unit

Parameter

CUHandle	Handle value of CU-49USB

Return value

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

Error code (obtained by CubGetLastError())

CUB_SUCCESS	Successfully completed
CUB_ERR_INVALIDPARAM	Specified CUHandle is invalid.
CUB_ERR_USB_TIMEOUT_SUCCESS_STOP_CUNET	Timeout occurred during USB communication, and CUnet communication successfully stopped.
CUB_ERR_USB_TIMEOUT_FAILED_STOP_CUNET	Timeout occurred during USB communication, and CUnet communication failed to stop.
CUB_ERR_REACQUISITION_OF_HANDLE	Handle has not been reacquired.
CUB_ERR_INVALID_SEQUENCE_NUMBER	Invalid sequence number
CUB_ERR_FAILED	Process failed due to unknown reason

3.4.9 CubReadWord

Format

BOOL CubReadWord(HANDLE CUHandle, WORD Adrs, WORD *Data);

Function

Reads 2 bytes of data from a specified address

Parameter

CUHandle	Handle value of CU-49USB
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 CubGetLastError())

CUB_SUCCESS	Successfully completed
CUB_ERR_INVALIDPARAM	Specified CUHandle is invalid. Adrs not a multiple of 2 Adrs out of range NULL has been specified for *Data.
CUB_ERR_USB_TIMEOUT_SUCCESS_STOP_CUNET	Timeout occurred during USB communication, and CUnet communication successfully stopped.
CUB_ERR_USB_TIMEOUT_FAILED_STOP_CUNET	Timeout occurred during USB communication, and CUnet communication failed to stop.
CUB_ERR_REACQUISITION_OF_HANDLE	Handle has not been reacquired.
CUB_ERR_INVALID_SEQUENCE_NUMBER	Invalid sequence number
CUB_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 0x3F02.
(The library for the CU-43USB had a range from 0x0000 through 0x0F02.)

3.4.10 CubWriteWord

Format

BOOL CubWriteWord(HANDLE CUHandle, WORD Adrs, WORD Data);

Function

Writes 2 bytes of data to a specified address

Parameter

CUHandle	Handle value of CU-49USB
Adrs	Target address to write
Data	Write data

Return value

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

Error code (obtained by CubGetLastError())

CUB_SUCCESS	Successfully completed
CUB_ERR_INVALIDPARAM	Specified CUHandle is invalid. Adrs not a multiple of 2 Adrs out of range
CUB_ERR_USB_TIMEOUT_SUCCESS_STOP_CUNET	Timeout occurred during USB communication, and CUnet communication successfully stopped.
CUB_ERR_USB_TIMEOUT_FAILED_STOP_CUNET	Timeout occurred during USB communication, and CUnet communication failed to stop.
CUB_ERR_REACQUISITION_OF_HANDLE	Handle has not been reacquired.
CUB_ERR_INVALID_SEQUENCE_NUMBER	Invalid sequence number
CUB_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 0x3F02.
(The library for the CU-43USB had a range from 0x0000 through 0x0F02.)

3.4.11 CubReadData

Format

BOOL CubReadData(HANDLE CUHandle, WORD Adrs, WORD WordLen, void *Data);

Function

Reads data of specified word length from a specified address

Parameter

CUHandle	Handle value of CU-49USB
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 CubGetLastError())

CUB_SUCCESS	Successfully completed
CUB_ERR_INVALIDPARAM	Specified CUHandle is invalid. Adrs not a multiple of 2 Adrs out of range WordLen out of range NULL has been specified for *Data.
CUB_ERR_USB_TIMEOUT_SUCCESS_STOP_CUNET	Timeout occurred during USB communication, and CUnet communication successfully stopped.
CUB_ERR_USB_TIMEOUT_FAILED_STOP_CUNET	Timeout occurred during USB communication, and CUnet communication failed to stop.
CUB_ERR_REACQUISITION_OF_HANDLE	Handle has not been reacquired.
CUB_ERR_INVALID_SEQUENCE_NUMBER	Invalid sequence number
CUB_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 0x3F02.
(The library for the CU-43USB had a range from 0x0000 through 0x0F02.)
The maximum size of WordLen is 0x0180 (0x0300 bytes).
(The library for the CU-43USB had the maximum size of 0x0400 (0x0800 bytes).)

3.4.12 CubWriteData

Format

BOOL CubWriteData(HANDLE CUHandle, WORD Adrs, WORD WordLen, void *Data);

Function

Writes data of specified word length to a specified address

Parameter

CUHandle	Handle value of CU-49USB
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 CubGetLastError())

CUB_SUCCESS	Successfully completed
CUB_ERR_INVALIDPARAM	Specified CUHandle is invalid. Adrs not a multiple of 2 Adrs out of range WordLen out of range NULL has been specified for *Data.
CUB_ERR_USB_TIMEOUT_SUCCESS_STOP_CUNET	Timeout occurred during USB communication, and CUnet communication successfully stopped.
CUB_ERR_USB_TIMEOUT_FAILED_STOP_CUNET	Timeout occurred during USB communication, and CUnet communication failed to stop.
CUB_ERR_REACQUISITION_OF_HANDLE	Handle has not been reacquired.
CUB_ERR_INVALID_SEQUENCE_NUMBER	Invalid sequence number
CUB_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 0x3F02.
(The library for the CU-43USB had a range from 0x0000 through 0x0F02.)
The maximum size of WordLen is 0x0180 (0x0300 bytes).
(The library for the CU-43USB had the maximum size of 0x0400 (0x0800 bytes).)

3.4.13 CubReadProtect

Format

BOOL CubReadProtect(HANDLE CUHandle, WORD BlockNo, void *Data);

Function

Reads data from Global Memory by using hazard protection function

Parameter

CUHandle	Handle value of CU-49USB
BlockNo	Target memory block number to read
*Data	Address to store read data (8 bytes)

Return value

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

Error code (obtained by CubGetLastError())

CUB_SUCCESS	Successfully completed
CUB_ERR_INVALIDPARAM	Specified CUHandle is invalid. BlockNo out of range NULL has been specified for *Data.
CUB_ERR_USB_TIMEOUT_SUCCESS_STOP_CUNET	Timeout occurred during USB communication, and CUnet communication successfully stopped.
CUB_ERR_USB_TIMEOUT_FAILED_STOP_CUNET	Timeout occurred during USB communication, and CUnet communication failed to stop.
CUB_ERR_REACQUISITION_OF_HANDLE	Handle has not been reacquired.
CUB_ERR_INVALID_SEQUENCE_NUMBER	Invalid sequence number
CUB_ERR_FAILED	Process failed due to unknown reason

Notes

BlockNo can be specified from 0 through 63.
The MKY49 is designed to prevent data hazards within the Global Memory.
CubReadData will give the same result for a single block of Global Memory.
This API is provided for compatibility with the CU-43USB.

3.4.14 CubWriteProtect

Format

BOOL CubWriteProtect(HANDLE CUHandle, WORD BlockNo, void *Data);

Function

Writes data to Global Memory by using hazard protection function

Parameter

CUHandle	Handle value of CU-49USB
BlockNo	Target memory block number to write
*Data	Address to store write data (8 bytes)

Return value

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

Error code (obtained by CubGetLastError())

CUB_SUCCESS	Successfully completed
CUB_ERR_INVALIDPARAM	Specified CUHandle is invalid. BlockNo out of range NULL has been specified for *Data.
CUB_ERR_USB_TIMEOUT_SUCCESS_STOP_CUNET	Timeout occurred during USB communication, and CUnet communication successfully stopped.
CUB_ERR_USB_TIMEOUT_FAILED_STOP_CUNET	Timeout occurred during USB communication, and CUnet communication failed to stop.
CUB_ERR_REACQUISITION_OF_HANDLE	Handle has not been reacquired.
CUB_ERR_INVALID_SEQUENCE_NUMBER	Invalid sequence number
CUB_ERR_FAILED	Process failed due to unknown reason

Notes

BlockNo can be specified from 0 through 63.
The MKY49 is designed to prevent data hazards within the Global Memory.
CubWriteData will give the same result for a single block of Global Memory.
This API is provided for compatibility with the CU-43USB.

3.4.15 CubGetFirmwareVersion

Format

short CubGetFirmwareVersion(HANDLE CUHandle);

Function

Obtains the firmware version number of CU-49USB

Parameter

CUHandle	Handle value of CU-49USB

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 CubGetLastError())

CUB_SUCCESS	Successfully completed
CUB_ERR_INVALIDPARAM	Specified CUHandle is invalid.
CUB_ERR_USB_TIMEOUT_SUCCESS_STOP_CUNET	Timeout occurred during USB communication, and CUnet communication successfully stopped.
CUB_ERR_USB_TIMEOUT_FAILED_STOP_CUNET	Timeout occurred during USB communication, and CUnet communication failed to stop.
CUB_ERR_REACQUISITION_OF_HANDLE	Handle has not been reacquired.
CUB_ERR_INVALID_SEQUENCE_NUMBER	Invalid sequence number
CUB_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.

3.4.16 CubStartAutoTrans

Format

BOOL CubStartAutoTrans(HANDLE CUHandle, WORD MfCnt);

Function

Starts regular updates of the entire Global Memory area and the MFR

Parameter

CUHandle	Handle value of CU-49USB
MfCnt	Update interval

Return value

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

Error code (obtained by CubGetLastError())

CUB_SUCCESS	Successfully completed
CUB_ERR_INVALIDPARAM	Specified CUHandle is invalid. MfCnt out of range
CUB_ERR_AUTO_TRANS_ALREADY_START	Regular update has already started.
CUB_ERR_USB_TIMEOUT_SUCCESS_STOP_CUNET	Timeout occurred during USB communication, and CUnet communication successfully stopped.
CUB_ERR_USB_TIMEOUT_FAILED_STOP_CUNET	Timeout occurred during USB communication, and CUnet communication failed to stop.
CUB_ERR_REACQUISITION_OF_HANDLE	Handle has not been reacquired.
CUB_ERR_INVALID_SEQUENCE_NUMBER	Invalid sequence number
CUB_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 CubReadGM and CubReadMFR after executing this API.

3.4.17 CubStopAutoTrans

Format

BOOL CubStopAutoTrans(HANDLE CUHandle);

Function

Stops regular updates of the entire Global Memory area and the MFR

Parameter

CUHandle	Handle value of CU-49USB

Return value

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

Error code (obtained by CubGetLastError())

CUB_SUCCESS	Successfully completed
CUB_ERR_INVALIDPARAM	Specified CUHandle is invalid.
CUB_ERR_AUTO_TRANS_STOP	Regular update has not started.
CUB_ERR_USB_TIMEOUT_SUCCESS_STOP_CUNET	Timeout occurred during USB communication, and CUnet communication successfully stopped.
CUB_ERR_USB_TIMEOUT_FAILED_STOP_CUNET	Timeout occurred during USB communication, and CUnet communication failed to stop.
CUB_ERR_REACQUISITION_OF_HANDLE	Handle has not been reacquired.
CUB_ERR_INVALID_SEQUENCE_NUMBER	Invalid sequence number
CUB_ERR_FAILED	Process failed due to unknown reason

Notes

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

3.4.18 CubReadGM

Format

BOOL CubReadGM(HANDLE CUHandle, void *Data);

Function

Obtains the latest data in the entire Global Memory areas by regular updates

Parameter

CUHandle	Handle value of CU-49USB
*Data	Address to store Global Memory data (512 bytes)

Return value

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

Error code (obtained by CubGetLastError())

CUB_SUCCESS	Successfully completed
CUB_ERR_INVALIDPARAM	Specified CUHandle is invalid. NULL has been specified for *Data.
CUB_ERR_AUTO_TRANS_STOP	Regular update is stopped.
CUB_ERR_REACQUISITION_OF_HANDLE	Handle has not been reacquired.
CUB_ERR_FAILED	Process failed due to unknown reason

Notes

The CU-49USB reads out internally Global Memory data and MFR in regular intervals when CubStartAutoTrans has been executed. This API is used to retrieve GM 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 directly to MKY49. If you want to make sure to have the latest data, use CubReadWord or CubReadData.

3.4.19 CubReadMFR

Format

BOOL CubReadMFR(HANDLE CUHandle, void *Data);

Function

Obtains the latest data of MFR by regular updates

Parameter

CUHandle	Handle value of CU-49USB
*Data	Address to store MFR (8 bytes)

Return value

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

Error code (obtained by CubGetLastError())

CUB_SUCCESS	Successfully completed
CUB_ERR_INVALIDPARAM	Specified CUHandle is invalid. NULL has been specified for *Data.
CUB_ERR_AUTO_TRANS_STOP	Regular update is stopped.
CUB_ERR_REACQUISITION_OF_HANDLE	Handle has not been reacquired.
CUB_ERR_FAILED	Process failed due to unknown reason

Notes

The CU-49USB reads out internally Global Memory data and MFR in regular intervals when CubStartAutoTrans has been executed. This API is used to retrieve MFR 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 directly to MKY49. If you want to make sure to have the latest data, use CubReadWord or CubReadData.

3.5 Limitations

The library for the CU-49USB has limitations similar to that for the CU-43USB 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 CU-49USB 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 CubGetLastError function to get one of the following two error codes:

- CUB_ERR_USB_TIMEOUT_SUCCESS_STOP_CUNET (9)
  When a timeout occurs, the API internally stops CUnet network and regular updates.
  The error code is given when it has successfully stopped the CUnet network and the regular updates.
- CUB_ERR_USB_TIMEOUT_FAILED_STOP_CUNET (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 CubCloseHandle function, and newly acquire a handle by CubOpenHandle function.
Until a handle is reacquired, any API function other than CubGetVersion, CubCountDevice, CubSearchBoard, CubGetLastError, CubOpenHandle and CubCloseHandle returns an error.
In this instance, CubGetLastError function returns CUB_ERR_REACQUISITION_OF_HANDLE (11).

4 MKY43-compatible mode

The CU-49USB has a MKY43-compatible mode to simulate the functions of CU-43USB.

The CU-49USB starts functioning as the CU-43USB when the power is turned on with the fourth position of the setting switch set to ON.
The CU-43USB driver is necessary to operate in this setting. Please refer to the manual for the CU-43USB and install the driver.

Many user applications ready for the CU-43USB can be used in the MKY43-compatible mode. However, MKY49 and MKY43, as individual ICs, are not compatible with each other in certain points.

This chapter explains these differences.

4.1 MKY43 registers

The CU-49USB is equipped with MKY49 as its control IC, while the CU-43USB is with MKY43. These two devices are not completely compatible.
Please refer to "Appendix 2: Deviations from the MKY43" in [**MKY49 Reference Manual**] for details of these differences.

Table 4.1 shows a list of MKY43 registers which can be accessed in the compatible mode of CU-49USB.
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 CU-49USB firmware compensates.
C .....Certain functionalities remain incompatible.
D .....Function is not available.

The following functions of MKY43 are changed in the MKY49:

◆ DataRenewal function
The MKY49 has no DataRenewal function.
Those registers and flags related to the function are not supported.

◆ INT1CR / IT1CR / INT1SR
The MKY49 has only one interrupt pin.
Those registers related to the second interrupt pin of the MKY43 are not supported.

◆ MGNC and MGNE of INT0CR / INT0SR / SSR
The MFR behaves differently for MKY49.
For MKY43, MGNC (bit 5) shows the state "MGR > MFR", and MGNE (bit 4) shows the state "MGR ≠ MFR".
For MKY49, bit 5 has MNG showing the state "MGR > MFR", and bit 4 has MOK showing the state "MGR ≤ MFR".
In the MKY43-compatible mode, the MNG is assigned for MGNC, but the MGNE is not supported and gives always "0".

◆ Hazard protection
The MKY49 has an SPI interface, which ensures by design that no data hazard occurs within a memory block.
The hazard protection function is simulated to maintain software compatibility. However, since it has no write-protection function, data is written even in GMM mode or to other-than-self-owned areas in MKY49 .

Table 4.1 MKY43 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	GM	64*64	Shared memory	✓				
0x0200	MSB	64*32	Mail send buffer	✓				
0x0300	RFR	64	Link detection	✓				
0x0308	LFR	64	Link detection	✓				
0x0310	MFR	64	Member detection	✓				
0x0318	DRFR	64	Data transition detection				✓	
0x0320	LGR	64	Link detection	✓				
0x0328	MGR	64	Member detection	✓				
0x0330	DRCR	64	Data transition detection				✓	
0x0338	RHCR0	16	Read hazard protection		✓			
0x033A	RHCR1	16	Read hazard protection		✓			
0x033C	WHCR0	16	Write hazard protection		✓			
0x033E	WHCR1	16	Write hazard protection		✓			
0x0340	MSLR	16	Mail sending	✓				
0x0342	MSRR	16	Mail sending	✓				
0x0344	MESR	16	Mail sending	✓				
0x0346	MSCR	16	Mail sending	✓				
0x0348	MR0CR	16	Mail receiving		✓			
0x034A	MR1CR	16	Mail receiving		✓			
0x034C	CCTR	16	System's supplementary		✓			
0x034E	UTCR	16	System's supplementary				✓	
0x0350	QCR	16	System's supplementary	✓				Type=0x05 added
0x0352	NFSR	16	System	✓				
0x0354	FSR	16	System	✓				
0x0356	BCR	16	System		✓			
0x0358	INT0CR	16	Interrupt handling			✓		MGNC≤MNG MGNE/DR not supported
0x035A	INT1CR	16	Interrupt handling				✓	
0x035C	IT0CR	16	Interrupt handling			✓		DR not supported
0x035E	IT1CR	16	Interrupt handling				✓	
0x0360	INT0SR	16	Interrupt handling			✓		MGNC≤MNG MGNE/DR not supported
0x0362	INT1SR	16	Interrupt handling				✓	
0x0364	SSR	16	System		✓			
0x0366	SCR	16	System			✓		DR not supported
0x0368	CCR	64	System	✓				
0x0370	RHPB0	64	Read hazard protection		✓			
0x0378	RHPB1	64	Read hazard protection		✓			
0x0380	WHPB0	64	Write hazard protection		✓			
0x0388	WHPB1	64	Write hazard protection		✓			
0x0400	MRB0	64*32	Mail receive buffer 0	✓				
0x0500	MRB1	64*32	Mail receive buffer 1	✓				

Revision History

Version No.	Date	Page	Contents
100	September 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.

CUnet (MKY49) USB Unit

CU-49USB Reference Manual

Document No.: P10992-100
Issued: September 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.