STD_CUB43PCI1_V1.0E

STEP TECHNICA CO.,LTD.

CUnet (MKY43) PCI Board

CUB-43PCI1

User's Manual

Notes

1. The information in this document is subject to change without prior notice. Before using this product, please confirm that this is the latest version of document.

2. Technical information in this document, such as explanations and circuit examples, are just for references to use this product in a proper way. When actually using this product, always fully evaluate the entire system according to the design purpose based on considerations of peripheral circuits and environment. We assume no responsibility for any incompatibility between this product and your system.

3. We assume no responsibility whatsoever for any losses or damages arising from the use of the information, products, and circuits in this document, or for infringement of patents and any other rights of a third party.

4. When using this product and the information and circuits in this document, we do not guarantee the right to use any property rights, intellectual property rights, and any other rights of a third party.

5. This product is not designed for use in critical applications, such as life support systems. Contact us when considering such applications.

6. 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.

Preface

This manual describes CUB-43PCI1, PCI board with MKY43 which is a kind of CUnet family IC.
Be sure to read "CUnet Introduction Guide" before using CUB-43PCI1 and understanding this manual.

● Target readers
・Those who first build a CUnet.
・Those who first use StepTechnica's CUB-43PCI1 to build a CUnet.

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

● Related manuals
・CUnet Introduction Guide
・CUnet Technical Guide
・CUnet MKY43 User's Manual

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

Revision history

Ver	Date	Content	
		Page	Description
Ver1.0E	Feb, 2019	-	Issued the first edition

Table of Contents

Chapter 1　Product Outline

Chapter 2　Hardware

Chapter 3　Software

Figures

Tables

Chapter 1 Product Outline

This chapter describes the product outline of CUB-43PCI1.

1.1 Features

CUB-43PCI1 is a PCI expansion bus supported CUnet communication board with MKY43 chip. This product is designed to help easy operation of MKY43 functions with StepTechnica's API for Windows.

1.2 Specifications

The specifications of CUB-43PCI1 are shown in Table 1-1.

Table 1-1 Specifications

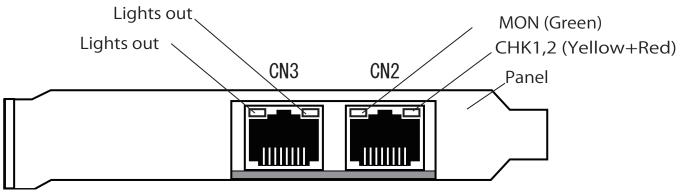
CUnet device	MKY43 × 1 pc
CUnet communication method	Half-duplex
CUnet communication rate	12M/6M/3Mbps (Set by MKY43 register)
CUnet communication connector	RJ45 type (8 pin modular) × 2pcs
Supported bus	PCI Ver2.2 supported, 32bit / 33MHz expansion bus 5V / 3.3V supported
Owned resource	16KB serial memory area (Automatically allocated by PnP)
Interrupt	1 line used (Automatically allocated by PnP)
Supported OS	Windows10 (64bit/32bit) Windows8.1 (64bit/32bit) Windows8 (64bit/32bit) Windows7 (64bit/32bit)
Power supply	DC +5.0V
Consumption current	500mA or less
Operating conditions	Temperature 0 to 50℃ Humidity 20 to 90% (with no condensation)
Storage conditions	Temperature 0 to 80℃ Humidity 0 to 90% (with no condensation)
Size	119.9mm(W) × 64.4mm(D) ※ Not including panel (Low Profile supported)
Accessory	Low-profile panel

Chapter 2 Hardware

This chapter describes hardware of CUB-43PCI1.

2.1 Connector specifications

The panel view and its details are shown in Fig.2-1.



Pin#	Function
4	TRX-
5	TRX+
8	Shield
1,2,3,6,7	Unused

MON	LED is connected to #MON pin of MKY43, lit green when link is established with other CUnet stations.
LCARE MCARE	Each LEDs are connected to #LCARE and #MCARE pin of MKY43. Lit yellow when LCARE is occurring, lit red when MCARE is occurring LEDs are lit red if LCARE and MCARE are occurring at the same time.
CN2、CN3	CUnet communication line Commercial CAT-5 or more greater straight-through cable for 100BASE-TX can be used.
Panel	Panel is connected to a PC case front panel. Connect a PC case to F.G. in accordance with the PC manual. Metal shell of CN2 / CN3, and No.8 pin can be connected to the panel by making R33 PAD and R43 PAD on the board shorted. (Refer to Fig.2-2, Fig.2-3.)

Fig. 2-1 Panel view

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

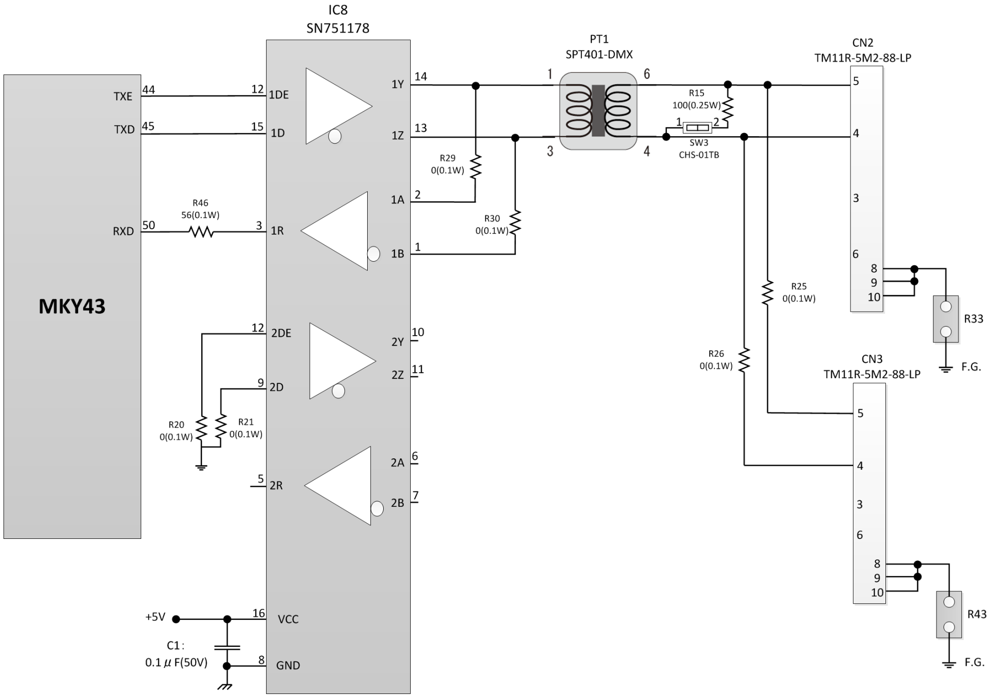


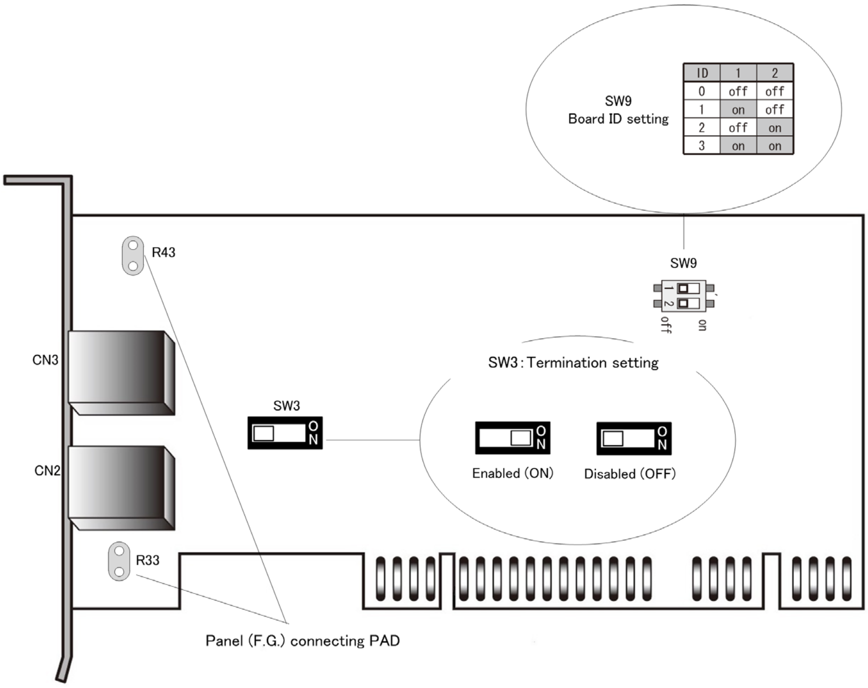
Fig. 2-2 Connector peripheral circuit

2.2 DIP switches

The settings of DIP switches of CUB-43PCI1 are shown in Fig.2-3.

If two or more CUB-43PCI1 devices are connected to one PC, set SW9 board IDs to individual number of each boards so that you can distinguish the boards using software. (Factory setting board ID : 0)

When CUB-43PCI1 is at a termination of multi-drop connection (an end of network cable), set SW3 ON to enable termination.
(Factory setting : Termination OFF)



ID	1	2
0	off	off
1	on	off
2	off	on
3	on	on

Fig. 2-3 Settings of CUB-43PCI board

2.3 Memory map

Table 2-1 describes memory map of CUB-43PCI1.
An address in memory map is the relative value from starting address of CUB-43PCI1, and actual address is the value which is added the starting address of the board.

Table 2-1 Memory map

Address	Description
000H ～ 5FFH	MKY43
600H ～ EFFH	Unused
F00H	Chip Reset Register
F02H	Board ID Register
F04H ～ FFFH	Unused

2.3.1 MKY43

For the details of memory map of MKY43, refer to "4.1.1 Memory map" in MKY43 User's Manual.

2.3.2 Unique register of CUB-43PCI1

F00H and F02H registers shown in Table 1-2 Memory map are unique registers of CUB-43PCI1. The details of these registers are described in the following.

Chip Reset Register Address : F00H

bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R/W	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	W
Function	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	CRST

[Function] By writing "1" to CRST(Chip ReSet), reset signal can be applied to #RST pin of MKY43.
A reset term to #RST pin is 100ms. This register is write-only, so data will be undefined when reading the register.

Board ID Register Address : F02H

bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R/W	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
Function	--	--	--	--	--	--	--	--	--	--	--	--	--	--	BID1	BID0

[Function] The value of board ID set in SW9 can be acquired by reading BID0,1 (Board ID) bits.
For details, refer to "2.2 DIP switches".

Caution

Do not access to unused area ("600H to EFFH" , "F04H to FFFH")shown in Table 2-1 Memory map.
It can make a whole system unstable.

2.4 External dimensions

External dimensions of CUB-43PCI1 are shown in Fig.2-4.

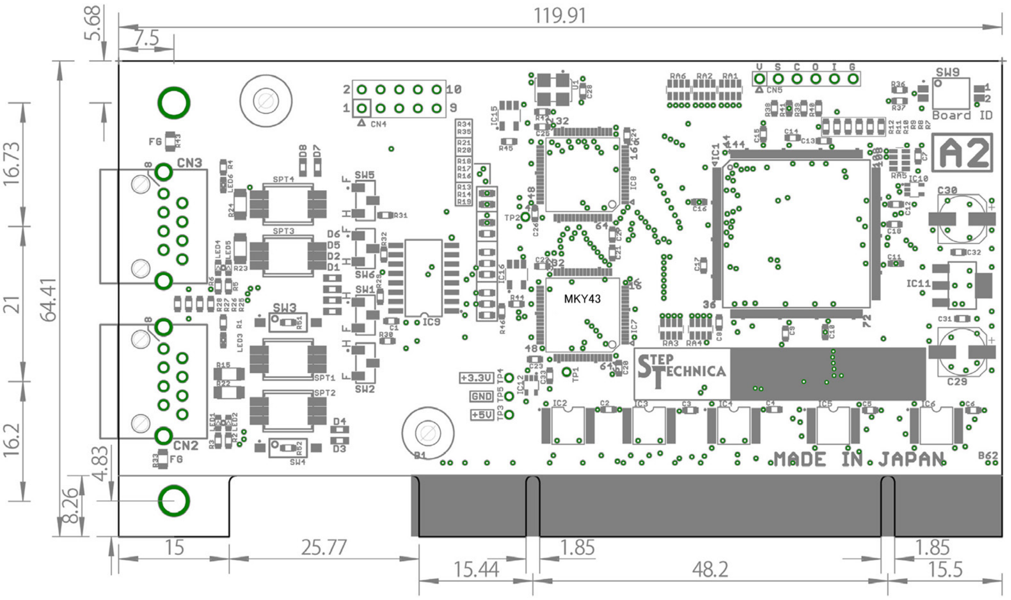


Fig. 2-4 External dimensions of CUB-43PCI1

Chapter 3 Software

This chapter describes API provided by StepTechnica.
The description in this manual is based on API version "1.0.0".
Please check the latest information on our website at using the product.

3.1 Outline

StepTechnica provides API to optimize the access to CUB-43PCI1 from user application on Windows.
You can download the API from our website below.
URL：http://www.steptechnica.com/en/download/index.html
The supported operating systems are as follows.
・Windows 10 (64bit/32bit)
・Windows 8.1 (64bit/32bit)
・Windows 8 (64bit/32bit)
・Windows 7 (64bit/32bit)
Provided API can be called from Microsoft Visual Studio and VB6.

3.2 Copyright and disclaimer

All documents, programs and program sources are belong to StepTechnica Co., Ltd.
The individuals, companies or other parties only who accept the cautions written below and use our CUB-43PCI1 are licenced to copy or use of these works of StepTechnica Co., Ltd.
You can not be revised and re-distribution or duplication and use some or all of the work other than this product without prior notice.

Caution

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

② Use API in proper ways with its instructions.

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

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

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

3.3 File structure

Files stored in "DLL" folder are the following.

【cub43pci1.dll】

DLL body :
Use this within Windows system folder or the same directory where the user program using this DLL is stored.

【cub43pci1.lib】

Import library

【cub43pci1.h】

Header file for DLL : Include this after Windows.h.

3.4 Restrictions

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

3.4.1 Multi-thread

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

3.4.2 Power saving mode

CUB-43PCI1 does not support for power saving mode of PC.
Use the board after stop the sleeping function of OS. When the PC has gone into the sleep mode, the power supply to MKY43 shuts down and the communication is stopped.
In addition, take care that each registers are initialized and GM, MSB, MRB0, MRB1 areas will be undefined state due to resetting at recovering from power saving mode.

3.4.3 Interrupt handling

INT0SR and INT1SR registers enable to check the status of interrupt occurrence of MKY43.
The internal driver has registers called interrupt factor register which retains the information in INT0SR and INT1SR at interrupt occurrence and interrupt count register which retains the interrupt occurrence count of each INT0SR and INT1SR.
The internal driver process the following procedure using these registers at interrupt occurrence.
(For instance, below describes when interrupt occurred at INT0.)

① Set the interrupt factor information in interrupt factor register
(Previous interrupt factor remains until it is being cleared by interrupt factor register from user application.)
② Increment the value of interrupt count register
③ Clear interrupt factor by writing "1" to the bit which is set "1" in 0-15bit of INT0SR

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

(1) Acquires the interrupt count from interrupt count register (CubGetInterrupt0Count, CubGetInterrupt1Count)
The internal driver retains interrupt count of each INT0, INT1 registers from MKY43.
This API function acquires the data from interrupt count register.

(2) Clears interrupt count register (CubClearInterrupt0Count, CubClearInterrupt1Count)
Clears the data of interrupt count register

(3) The internal driver retains interrupt factor of each INT0, INT1 registers from MKY43 in interrupt factor register.
The data is acquired from interrupt factor register in this API function.

(4) Clear function of interrupt factor register (CubClearInterrupt0StatusInfo, CubClearInterrupt1StatusInfo)
Clears the specified interrupt factor from interrupt factor register

Check the interrupt factor and interrupt count from MKY43 using these functions in user application.

3.4.4 The access without StepTechnica-provided driver

When you directly access to CUB-43PCI1 without StepTechnica-provided driver, note the following point.

Always use 32bit-access to access CUB-43PCI1. At that time, lower 16 bit data will be used and upper 16 bit will be unused.
For this reason, address to access needs to be specified 2 times longer than memory map mentioned above.
For example, in order to read 200H address of MKY43, make the lower 16bit in 400H of CUB-43PCI1 to acquire 2 bytes data in 200H of MKY43 by executing 32bit Read.
This access method applies to CUB-43PCI1 unique register.

3.5 API specifications

This chapter describes the API specifications.
The API functions are provided to optimize the operation of CUB-43PCI1 from user application.

The supported API functions are listed in Table 3-1.

Table 3-1 API functions

Function	Description
CubGetVersion	Acquires API version number
CubGetLastError	Acquires the termination status of API function
CubCountDevice	Acquires the number of CUB-43PCI boards connected to PC
CubBoardID	Acquires the board ID
CubResetBoard	Resets the specified MKY43
CubSearchBoard	Acquires the number of CUB-43PCI1 boards and its board IDs
CubOpenHandle	Opens the handle of CUB-43PCI1
CubCloseHandle	Closes the handle of CUB-43PCI1
CubReadByte	Data read of 1 byte from the specified address
CubWriteByte	Data write of 1 byte to the specified address
CubReadWord	Data read of 2 bytes from the specified address
CubWriteWord	Data write of 2 bytes to the specified address
CubGetInt0Counter CubGetInt1Counter	Acquires the interrupt count of INT0, 1 retained in the internal driver
CubClearInt0Counter CubClearInt1Counter	Clears the interrupt count of INT0, 1 retained in the internal driver
CubGetInt0StatusInfo CubGetInt1StatusInfo	Acquires the interrupt factor of INT0, 1 retained in the internal driver
CubClearInt0StatusInfo CubClearInt1StatusInfo	Clears the specified interrupt factor from INT0, 1 interrupt factor data retained in the internal driver

3.5.1　CubGetVersion

Format

UINT CubGetVersion(void);

Function

Acquires the API version number

Parameter

None

Return value

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

Error code

The error codes and error factors returned by CubGetLastError after executing this function is as follows.

CUB_SUCCESS	Terminated normally

Note

Version numbering of API is shown in Table 3-2.
The reasons for updating the version number are as follows.

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

Table 3-2　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

3.5.2 CubGetLastError

Format

UINT CubGetLastError(void);

Function

Acquires the termination status of API function called last time

Parameter

None

Return value

Returns an error code defined in cub43pci1.h.

Note

The error codes defined by cub43pci1.h are shown in Table 3-3.

Table 3-3 Error code list

Character constant	Value	Description
CUB_SUCCESS	0	Terminated normally
CUB_ERR_DEVICENOTEXIST	1	Device does not exist.
CUB_ERR_ALREADYOPENED	2	Handle has already opened.
CUB_ERR_CLOSED	3	CubOpenHandle has never been called.
CUB_ERR_INVALIDPARAM	4	Called with invalid parameter
CUB_ERR_NORESOUCE	5	No resource to execute the process
CUB_ERR_FAILED	6	The process failed due to unknown reason.
CUB_NOTCALLYET	99	API function has never been called.

3.5.3 CubCountDevice

Format

INT CubCountDevice(void);

Function

Returns the number of CUB-43PCI1 devices connected to PC.

Parameter

None

Return value

Returns the number of CUB-43PCI devices connected to PC

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

Error code

The error code and error factor returned by CubGetLastError after executing CubGetLastError is as follows.

CUB_SUCCESS	Terminated normally

Note

More than 5 devices cannot be connected to a PC.

3.5.4 CubBoardID

Format

INT CubBoardID(HANDLE CUBHandle);

Function

Acquires the board IDs of CUB-43PCI1.

Parameter

HANDLE CUBHandle	The handle value of CUB-43PCI1

Return value

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

Error code

The error codes and error factors returned by CubGet LastError are as follows.

CUB_SUCCESS	Terminated normally
CUB_ERR_INVALIDPARAM	Invalid handle value
CUB_ERR_FAILED	The process failed due to unknown reason.

3.5.5 CubResetBoard

Format

BOOL CubResetBoard (HANDLE CUBHandle);

Function

Resets MKY43

Parameter

HANDLE CUBHandle	The handle value of CUB-43PCI1

Return value

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

Error code

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

CUB_SUCCESS	Terminated normally
CUB_ERR_INVALIDPARAM	Invalid handle value
CUB_ERR_FAILED	The process failed due to unknown reason.

Addendum

To access MKY43, wait 100ms or longer after resetting

3.5.6 CubSearchBoard

Format

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

Function

Returns the number of CUB-43PCI1 devices connected to PC and its board ID list.

Parameter

*board_num — Specify an address to byte-type variable to which the number of boards are set.
The descriptions of set value are the following.

・-1	：	Five or more
・0	：	Not connected
・1 ～ 4	：	Number of boards identified

*board_id_list — To receive the board ID, specify a ponter which has an array of four elements of byte-type. It is also possible to specify NULL.
If NULL has been specified, only the number of boards are returned.
The descriptions of set value are the following.

・0x00 ～ 0x03	：	Board ID
・0xFF	：	Not identified

Return value

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

Error code

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

CUB_SUCCESS	Terminated normally
CUB_ERR_INVALIDPARAM	NULL has been specified to *board_num
CUB_ERR_FAILED	The process failed due to unknown reason.

Addendum

Board ID is set by SW9.
If two or more CUB-43PCI1 devices are connected to a PC, you can distinguish them by its board IDs.
IN this API, you can distinguish up to four CUB-43PCI1 boards.
Specify the byte-type array as a parameter as shown below.

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

As an example, three CUB-43PCI1 boards are connected to a PC, and each board IDs are set in sequence ;
1st board ID = 0, 2nd board ID = 1, 3rd board ID = 2
If the boards have been identified by the PC in sequence with first, third, and second, and run CubSearchBoard, 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
```

3.5.7 CubOpenHandle

Format

HANDLE CubOpenHandle(int index_no);

Function

Opens the handle of CUB-43PCI1

Parameter

int index_no	Index number 0 to 3 can be specified as an index number. If only one CUB-43PCI1 is connected, set '0'. For details, refer to "Addendum".

Return value

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

Error code

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

CUB_SUCCESS	Terminated normally
CUB_ERR_DEVICENOTEXIST	Device does not exist.
CUB_ERR_FAILED	The process failed due to unknown reason.

Addendum

CIf only one CUB-43PCI1 board is connected, it's not necessary to execute CUBSearchBoard.
If two or more CUB-43PCI1 boards are connected to a PC, execute "CubSearchBoard" to check which CUB-43PCI1 to manipulate.
As an example, three CUB-43PCI1 boards are connected to a PC, and each board IDs are set in sequence ;
1st board ID = 0, 2nd board ID = 1, 3rd board ID = 2.To acquire the handle value of board ID=2, execute the following.

```
BYTE board_num;
BYTE board_id_list[4];
CubSearchBoard(&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 CubOpenHandle.
Close the handle with CubCloseHandle at finishing the program.

3.5.8 CubCloseHandle

Format

BOOL CubCloseHandle(HANDLE CUBHandle);

Function

Closes the handle which is acquired by CubOpenHandle

Parameter

HANDLE CUBHandle	The handle value of CUB-43PCI1

Return value

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

Error code

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

CUB_SUCCESS	Terminated normally
CUB_ERR_INVALIDPARAM	Invalid handle value
CUB_ERR_FAILED	The process failed due to unknown reason.

3.5.9 CubReadByte

Format

BOOL CubReadByte(HANDLE CUBHandle,const ULONG Adr,BYTE *Dat);

Function

Reads 1 byte data from the specified address

Parameter

HANDLE CUBHandle	The handle value of CUB-43PCI1
const ULONG Adr	Address value Input condition is the following. ・Input range : 0x0000 to 0x0FFE
BYTE *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 CubGetLastError after executing this function are as follows.

CUB_SUCCESS	Terminated normally
CUB_ERR_INVALIDPARAM	Invalid handle value Adr is out of range. *NULL has been specified to *Dat.
CUB_ERR_FAILED	The process failed due to unknown reason.

3.5.10 CubWriteByte

Format

BOOL CubWriteByte(HANDLE CUBHandle, const ULONG Adr, const BYTE Dat);

Function

Writes 1 byte data to the specified address

Parameter

HANDLE CUBHandle	The handle value of CUB-43PCI1
const ULONG Adr	Address value Input condition is the following. ・Input range : 0x0000 to 0x0FFE
const 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 CubGetLastError after executing this function are as follows.

CUB_SUCCESS	Terminated normally
CUB_ERR_INVALIDPARAM	Invalid handle value Adr is out of range.
CUB_ERR_FAILED	The process failed due to unknown reason.

3.5.11 CubReadWord

Format

BOOL CubReadWord(HANDLE CUBHandle,const ULONG Adr,WORD *Dat);

Function

Reads 2 bytes data from the specified address.

Parameter

HANDLE CUBHandle	The handle value of CUB-43PCI1
const ULONG Adr	Address value Input condition is the following. ・Multiples of 2 ・Input range : 0x0000 to 0x0FFE
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 CubGetLastError after executing this function are as follows.

CUB_SUCCESS	Terminated normally
CUB_ERR_INVALIDPARAM	Invalid handle value Adr is out of range. Adr is not a multiple of 2. NULL has been specified to *Dat.
CUB_ERR_FAILED	The process failed due to unknown reason.

3.5.12 CubWriteWord

Format

BOOL CubWriteWord(HANDLE CUBHandle, const ULONG Adr, const WORD Dat);

Function

Writes 2 bytes data to the specified address

Parameter

HANDLE CUBHandle	The handle value of CUB-43PCI1
const ULONG Adr	Address value Input condition is the following. ・Multiples of 2 ・Input range : 0x0000 to 0x0FFE
const 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 CubGetLastError after executing this function are as follows.

CUB_SUCCESS	Terminated normally
CUB_ERR_INVALIDPARAM	Invalid handle value Adr is out of range. Adr is not a multiple of 2.
CUB_ERR_FAILED	The process failed due to unknown reason.

3.5.13 CubGetInt0Counter , CubGetInt1Counter

Format

BOOL CubGetInt0Counter(HANDLE CUBHandle, BYTE *int0Counter);
BOOL CubGetInt1Counter(HANDLE CUBHandle, BYTE *int1Counter);

Function

Acquires the data of INT0, 1 interrupt count register retained by internal driver.
Interrupt count increments from 0 to 255 (0xFF) and returns to 0.

Parameter

HANDLE CUBHandle	The handle value of CUB-43PCI1
BYTE *int0Counter、int1Counter	The storage address of interrupt count

Return value

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

Error code

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

CUB_SUCCESS	Terminated normally
CUB_ERR_INVALIDPARAM	Invalid handle value NULL has been specified to *int0Counter, *int1Counter.
CUB_ERR_FAILED	The process failed due to unknown reason.

3.5.14 CubClearInt0Counter , CubClearInt1Counter

Format

```
BOOL CubClearInt0Counter (HANDLE CUBHandle);
BOOL CubClearInt1Counter (HANDLE CUBHandle);
```

Function

Clears the data of INT0, INT1 interrupt counter register retained by internal driver

Parameter

HANDLE CUBHandle	The handle value of CUB-43PCI1

Return value

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

Error code

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

CUB_SUCCESS	Terminated normally
CUB_ERR_INVALIDPARAM	Invalid handle value
CUB_ERR_FAILED	The process failed due to unknown reason.

3.5.15 CubGetInt0StatusInfo , CubGetInt1StatusInfo

Format

BOOL CubGetInt0StatusInfo (HANDLE CUBHandle,WORD *int0Info)
BOOL CubGetInt1StatusInfo (HANDLE CUBHandle,WORD *int1Info)

Function

Acquires the data of INT0, 1 interrupt factor retained by internal driver

Parameter

HANDLE CUBHandle	The handle value of CUB-43PCI1
WORD * int0Info,*int1Info	The storage address of interrupt factor

Return value

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

Error code

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

CUB_SUCCESS	Terminated normally
CUB_ERR_INVALIDPARAM	Invalid handle value NULL has been specified to *int0Info, *int1Info.
CUB_ERR_FAILED	The process failed due to unknown reason.

Note

The configuration of parameter set to int0Info, int1Info are described in Table 3-4.

If the interrupt has occurred, "1" is set to the bit corresponding to interrupt factor.

The arrangement of interrupt factors equals to INT0SR, INT1SR of MKY43.

Table 3-4　Internal configuration of int0Info , int1Info

bit	Interrupt factor
15	Interrupt by jammer detect
14	Interrupt by receiving of PING instruction
13	Interrupt by resize overlap
12	Interrupt by receiving break packet
11	Interrupt by "Link NG" judgement
10	Interrupt by "Link OK" judgement
9	Interrupt by increasing and decreasing of MFR bits to which "1" is set
8	Interrupt by entering RUN phase
7	Interrupt by network stop
6	Interrupt by resize complete
5	Interrupt by MGR > MFR judgement
4	Interrupt by MGR ≠ MFR judgement
3	Interrupt by finishing mail sending
2	Interrupt by finishing mail receiving
1	Interrupt by data renewal
0	Interrupt by ALM

3.5.16 CubClearInt0StatusInfo , CubClearInt1StatusInfo

Format

BOOL CubClearInt0StatusInfo (HANDLE CUBHandle, WORD clearInt0Info);
BOOL CubClearInt1StatusInfo (HANDLE CUBHandle, WORD clearInt0Info);

Function

Clears the specified interrupt factor from INT0, INT1 interrupt factor data retained by internal driver

Parameter

HANDLE CUBHandle	The handle value of CUB-43PCI1
WORD clearInt0Info、clearInt0Info	Specifies the cleared interrupt factor

Return value

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

Error code

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

CUB_SUCCESS	Terminated normally
CUB_ERR_INVALIDPARAM	Invalid handle value
CUB_ERR_FAILED	The process failed due to unknown reason.

Note

Interrupt factors and its setting values are shown in Table 3-5.

Set the setting value which are corresponded to each interrupt factors to clearInt0Info, clearInt1Info.

Set the logical or of each setting values to clear multiple interrupt factors.

Table 3-5 Interrupt factors to clear and its setting values

Interrupt factor	Setting value
Clear the interrupt by Jammer detect	0x8000
Clear the interrupt by receiving PING instruction	0x4000
Clear the interrupt by occurrence of resize overlap	0x2000
Clear the interrupt by receiving BREAK packet	0x1000
Clear the interrupt by "Link NG" judgement	0x0800
Clear the interrupt by "Link OK" judgement	0x0400
Clears the interrupt by increasing and decreasing of MFR bits to which are set "1"	0x0200
Clears the interrupt by entering RUN phase	0x0100
Clears the interrupt by network stop	0x0080
Clear the interrupt by finishing resizing	0x0040
Clear the interrupt by MGR > MFR judgement	0x0020
Clear the interrupt by MGR ≠ MFR judgement	0x0010
Clear the interrupt by finishing mail sending	0x0008
Clear the interrupt by finishing mail receiving	0x0004
Clear the interrupt by data renewal	0x0002
Clear the interrupt by ALM	0x0001

3.6 Sample program

3.6.1 Access sample to MKY43

The sample program that works for initializing MKY43, setting CUnet communication mode, acquiring the value of global memory with this API is described in the following.

```
int main(int argc, char *argv[])
{
    HANDLE CUBHandle;
    WORD   mky43_scr;
    WORD   sa1_gm[4];
    WORD   sa63_gm[4];
    int    i;
    UINT   api_version;


    /** Checking the version of API */
    api_version = CubGetVersion();
    if (api_version < 0x100 || api_version > 0x199) {
        printf(" This version of cub43pci1.dll is not compatible.\n");
        exit(1);
    }

    /** Generating handle */
    CUBHandle = CubOpenHandle(0);
    if (CUBHandle == INVALID_HANDLE_VALUE) {
        exit(1);
    }

    /** Initializing MKY43 */
    // (1) Write 0x00 to 0x000 to 0x2FF(GM + MSB) in memory map
    for (i=0;i<0x300;i+=2) {
        CubWriteWord(CUBHandle, i, 0);
    }

    // (2) Write 0x00 to 0x400 to 0x5FF(MRB0 + MRB1) in memory map
    for (i=0x400;i<0x600;i+=2) {
        CubWriteWord(CUBHandle, i, 0);
    }

    // (3) Set communication mode
    // (3-1) Set GMM function ON to write to BCR
    CubWriteWord(CUBHandle, 0x366, 0x8000);
    // (3-2) Set the network condition to BCR
    //     Set BCR as follows in this sample program: SA=0, OWN=1, BPS=6Mbps
    CubWriteWord(CUBHandle, 0x356, 0x0180);
    // (3-3) GMM function OFF
    CubWriteWord(CUBHandle, 0x366, 0x0000);
    /** Start CUnet */
    CubWriteWord(CUBHandle, 0x366, 0x0100);

     /** In this sample program, executing data read of SA1 SA63 global memory assuming that the link is
established between two CUnet station (SA1 and SA63) other than CUB-43PCI1.
      */
    while(1) {
        /** Checking the state of CUnet network*/
        CubReadWord(CUBHandle, 0x366, &mky43_scr);
        if ((mky43_scr&0x0100)==0) {
            CubWriteWord(CUBHandle, 0x366, 0x0100);  // Restart if network has been stopping

        }
        // Read global memory of SA1
        CubReadWord(CUBHandle, 0x0008, &sa1_gm[0]);
        CubReadWord(CUBHandle, 0x000A, &sa1_gm[1]);
        CubReadWord(CUBHandle, 0x000C, &sa1_gm[2]);
        CubReadWord(CUBHandle, 0x000E, &sa1_gm[3]);
        // Read global memory of SA63
        CubReadWord(CUBHandle, 0x01f8, &sa63_gm[0]);
        CubReadWord(CUBHandle, 0x01fA, &sa63_gm[1]);
        CubReadWord(CUBHandle, 0x01fC, &sa63_gm[2]);
        CubReadWord(CUBHandle, 0x01fE, &sa63_gm[3]);
    }
    /* Close the generated handle */
    CubCloseHandle(CUBHandle);
    return 0;
}
```

3.6.2 Sample program of interrupt handling

This chapter describes the sample program to check the setting and occurrence of interrupt to MKY43 using this API.

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

   /* Generating the handle */
   CUBHandle = CubOpenHandle(0);
   /* Checking the generated handle */
   if (CUBHandle == INVALID_HANDLE_VALUE) {
           exit(1);
   }

   // MKY43 START = 0
   CubWriteWord(CUBHandle, 0x366, 0x0000);

   /* Clear the interrupt factor register */
   CubClearInt0StatusInfo(CUBHandle, 0xffff);

   /* Clear the interrupt count register */
   CubClearInt0Counter(CUBHandle);
   int0_lastTime_numOfOccurr = 0;      // Interrupt count : 0

   /* Set the interrupt factor. INT0 interrupt is occurred at network stopping. */
   CubWriteWord(CUBHandle, 0x358, 0x0080);

   /* Network start instruction*/
   CubWriteWord(CUBHandle, 0x366, 0x0100);

   while (1) {
      /* Acquire the data of interrupt count register */
      CubGetInt0Counter(CUBHandle, &int0_current_numOfOccurr);
      /* Interrupt is occurring if the count is not equal to the previous interrupt count. */
      if (int0_lastTime_numOfOccurr != int0_current_numOfOccurr) {
         /* Copy the current value to the previous value */
         int0_lastTime_numOfOccurr = int0_current_numOfOccurr;
         /* Acquire the data of interrupt factor register*/
         CubGetInt0StatusInfo(CUBHandle, &int0_factor);
         /* Confirm that the interrupt factor is CHECK-1 or not */
         if ((int0_factor & 0x0080) == 0x0080) {
            /* ---The process when the network stop has been occurred is described here. --- */
            /*Clear INT0 interrupt factor register*/
            CubClearInt0StatusInfo(CUBHandle, 0x0080);
         }
      }
   }
   /*Close the generated handle*/
   CubCloseHandle(CUBHandle);

   return 0;
}
```

■ Developed and manufactured by

StepTechnica Co., Ltd.

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

CUnet (MKY43) PCI board
CUB-43PCI1
User's Manual

Document No. : STD_CUB43PCI1_V1.0E
Issued : February 2019