STD_CUB43PCIEXP_V2.0E

STEP TECHNICA CO.,LTD.

STEP TECHNICA

CUnet (MKY43) PCI Express Board

CUB-43PCIEXP
User's Manual

Revision history

Date	Version	Content	Note
AUG 2018	2.0E	Issued the first edition	

Preface

This manual describes CU-43PCIEXP, PCI Express board with MKY43 which is a kind of CUnet family IC.
Be sure to read "CUnet Introduction Guide (A Guide to the CUnet Protocol)" in advance to use CUB-43PCIEXP and understand this manual.

● Target readers
・Those who first build a CUnet
・Those who first use StepTechnica's CUB-43PCIEXP 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 (A Guide to the CUnet Protocol)
・CUnet Technical Guide (For Network)
・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).

Table of contents

Figures

Tables

Chapter 1　Product outline

This chapter describes product outline of CUB-43PCIEXP.

1.1　Features

CUB-43PCIEXP is a PCI expansion bus supported CUnet communication board with MKY43 chip.
You can easily use MKY43 functions with manufacturer provided API for Windows library.

1.2　Specifications

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

Table 1-1　Specifications

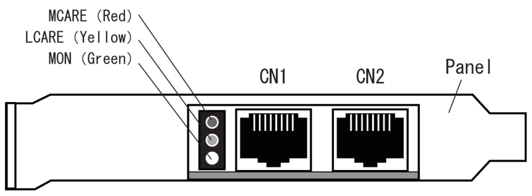
Board name	CUB-43PCIEXP
CUnet device	MKY43 1pc
CUnet communication mode	Half-duplex
CUnet transfer rate	12M/6M/3Mbps（Set in MKY43 register）
Connector	CUnet communication connector（RJ-45 type 2 pcs）
Supported bus	PCI Express x1 Gen1 supported expansion bus
Owned resource	16KB serial memory area（Automatically allocated by PnP）
Interrupt	1 line used（Automatically allocated by PnP）
Maximum concurrent use	4 units
Supported OS	Windows8.1 （64bit/32bit） Windows8 （64bit/32bit） Windows7 （64bit/32bit）
Power supply	DC +3.3V
Consumption current	500mA or less
Atmospheric conditions	Temperature 0 ～ 50℃ Humidity 20 ～ 90%（Without condensation）
External dimensions	119.9mm（W）× 68.9mm（H） ※ Not including a panel
Accesorries	LOW profile bracket
Provided software	Driver for Windows API CUeditor43J_PCI

Chapter 2 Hardware

This chapter describes hardware of CUB-43PCIEXP.

2.1 Connector specifications

The panel side view of CUB-43PCIEXP and details are shown in Fig.2-1.



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

MON	LED connected to MON pin of MKY43 is lit green when it has been successfully communicating with other CUnet stations.
LCARE	LED connected to LCARE pin of MKY43 is lit yellow when LCARE has occurred.
MCARE	LED connected to MCARE pin of MKY43 is lit red when MCARE has occurred.
CN1、CN2	CUnet communication line Commercial CAT5 or more newer straight-through cable for 100BASE-TX can be used.
Panel	Panel is connected to a PC case front panel. PC case is connected to F.G. in accordance with the PC manual. Metal shell of CN1 / CN2, and #8 pin can be connected to the panel by making PAD1 and PAD2 on the board shorted.

Fig.2-1 Panel view

CN1 / CN2 connector peripheral circuit is shown in Fig.2-2.

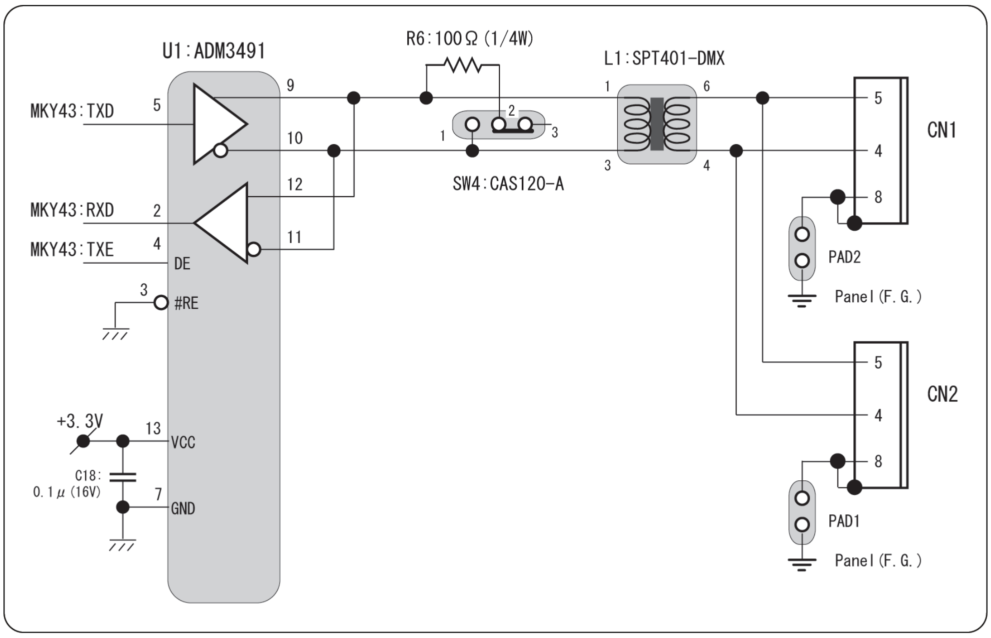
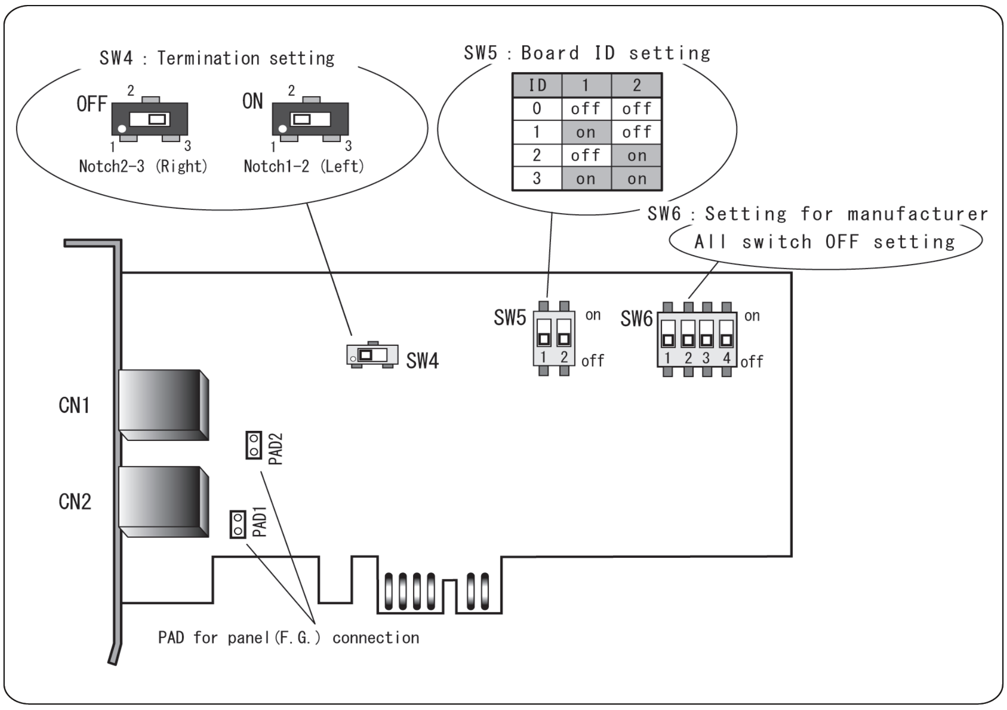


Fig.2-2 Connector peripheral circuit

2.2 DIP switches

DIP switch settings of CUB-43PCIEXP are shown in Fig.2-3.



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

Fig.2-3 CUB-43PCIEXP board settings

2.2.1　Board ID switch（SW5）

If two or more CUB-43PCIEXP devices connected to one PC, set SW5 board IDs to individual number of each boards.
（Factory setting board ID : 0）

2.2.2　Termination setting switch（SW4）

If CUB-43PCIEXP is the termination of CUnet communication line, switch the termination ON（enable）.
If the termination is ON, communication line is terminated in 100 Ω .
If CUB-43PCIEXP is not a termination, always set this switch OFF so that it is not to be terminated.
（Factory setting : Termination OFF）

2.2.3　Manufacturer's setting switch（SW6）

These setting switches are only for manufacturer. Always set these switches OFF when using.

2.3 Memory map

The memory map of CUB-43PCIEXP is shown in Table 2-1.
The addresses in memory map are the relative value from starting address.
Actual address will be the value which is added the starting address value of the board automatically allocated by PCI BIOS.

Table 2-1　Memory map

Address	Description
000H ～ 5FFH	MKY43
600H ～ BFFH	Unused
C00H	Chip Reset Register
C02H ～ DFFH	Unused
E00H	Board ID Register
E02H ～ FFFH	Unused

2.3.1　MKY43

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

2.3.2　Unique register of CUB-43PCIEXP

C00H and E00H in memory map of Fig.2-1 are the unique registers of CUB-43PCIEXP.

Chip Reset Register　　Address：C00H

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

[Function]　Add reset signal to RST pin of MKY43 by writing "1" to CRST（Chip ReSeT）bit.
The reset period of RST pin is 280ns. This register is write-only, so the data will be undefined value when you read this register.

Board ID Register　　Address：E00H

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

[Function]　The value of board IDs set in SW5 can be obtained by reading BID0,1（Board ID）bit.
See "2.2 DIP switches" for details.

Caution

Do not access to unused area（"600H to BFFH, C02H to DFFH, E02H to FFFH"）shown in Table 2-1.
It can make the whole system unstable.

2.4 To access without attached driver software

When directly access to CUB-43PCIEXP without StepTechnica's driver software, note the following point.

Always use 32bit access to access to CUB-43PCIEXP. 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 instance, in order to read 200H address of MKY43, make the lower 16bit in 400H of CUB-43PCI-EXP obtain 2 bytes data in 200H of MKY43 by executing 32bit Read.
This access method applies to CUB-43PCIEXP unique register.

Chapter 3　Software

This chapter describes API provided by StepTechnica.
This manual is edited on the basis of Major number "2" or newer API version.
When using this product, check the latest information on our website.

3.1　Outline

StepTechnica's API enables to access CUB-43PCIEXP from user application in 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 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-43PCIEXP are licenced to copy or use of these works of Step Technica 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 R&D department.

3.3 File structure

Files stored in "DLL" folder are the following.

【cub43pciexp.dll】
DLL body :
Use this within Windows system folder or user program folder using this DLL.

【cub43pciexp.lib】
Import library for Microsoft Visual C/C++

【cub43pciexp.h】
Header file for API :
Include this after than Windows.h when using.

3.4 Restrictions

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

3.4.1 Multi thread

API in DLL 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-43PCIEXP does not support for power saving mode of operating systems.
Stop sleep mode of OS before using. When OS goes into sleep mode, power supply to MKY43 chip mounted on the board will shut down and the communication will be stopped.
Also, make sure that MKY43 will be reset at recovering from power saving mode, so each registers will be initialized and global memory, mail sending buffer, mail receiving buffer 0 and mail receiving buffer 1 area will be undefined.

3.4.3　Interrupt handling

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

① Set the interrupt factor information in interrupt factor register.
(The past interrupt factor remains until be cleared by interrupt factor register from user application.)
② Increment the value of interrupt count register.
③ Release MKY43 interrupt by writing "1" to the bit which is set "1" in 0-15 bit of INT0SR.

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

(1) Obtain interrupt count from interrupt count register.
The internal driver retains interrupt count of each INT0, INT1 registers from MKY43.
This API function obtains the information from interrupt count register.
(2) Clear the interrupt count register (CubClearInterrupt0Count, CubClearInterrupt1Count) .
This API clears the interrupt count register.
(3) Obtain interrupt factor register (CubGetInterrupt0StatusInfo,CubGetInterrupt1StatusInfo) .
The internal driver retains interrupt factor of each INT0, INT1 registers from MKY43.
This API obtains the information from interrupt factor register.
(4) Clear function in interrupt factor register (CubClearInterrupt0StatusInfo, CubClearInterrupt1StatusInfo)
Clear the interrupt factor specified by interrupt factor register.

In user application, use these functions to check the interrupt count and interrupt factor from MKY43.

3.5 API specifications

The supported API functions are listed in the Table 3-1.
The API functions described hereafter are recorded in cub43pciexp.h.

Table 3-1 API functions

Functions	Description
CubGetVersion	Obtains API version number
CubGetLastError	Obtains the termination status of API functions
CubCountDevice	Obtains the number of CUB-43PCIEXP board connected to PC
CubBoardID	Obtains the board ID
CubResetBoard	Resets MKY43
CubSearchBoard	Obtains the number of CUB-43PCIEXP board connected to PC and its board ID
CubOpenHandle	Obtains the handle value of CUB-43PCIEXP
CubCloseHandle	Closes the handle obtained by CubOpenHandle
CubReadByte	Reads 1 byte data from specified address
CubWriteByte	Writes 1 byte data from specified address
CubReadWord	Reads 2 bytes data from specified address
CubWriteWord	Writes 2 bytes data from specified address
CubGetInterrupt0Count CubGetInterrupt1Count	Obtains INT0, INT1 interrupt count retained in the internal driver
CubClearInterrupt0Count CubClearInterrupt1Count	Clears INT0, INT1 interrupt count retained in the internal driver
CubGetInterrupt0StatusInfo CubGetInterrupt1StatusInfo	Clears INT0, INT1 interrupt factor retained in the internal driver
CubClearInterrupt0StatusInfo CubClearInterrupt1StatusInfo	Clears specified INT0, INT1 interrupt factor retained in the internal driver

3.5.1　CubGetVersion

Format

UINT CubGetVersion（void）;

Function

Obtains API version number

Parameter

None

Return value

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

Error code

The error code and error factor returned by CubGet LastError after executing this function is as follows.

CUB_SUCCESS	Terminated normally

Note

The configuration of API version number is as 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

Obtains the termination state of the API called last time

Parameter

None

Return value

Returns the error code defined in cub43pciexp.h.

Note

The error codes defined in cub43pciexp.h are shown in Table 3-3.

Table 3-3 Error code list

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

3.5.3 CubCountDevice

Format

INT CubCountDevice（void）;

Function

Obtains the number of CUB-43PCIEXP connected to PC

Parameter

None

Return value

Returns the number of CUB-43PCIEXP connected to PC.

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

Error code

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

CUB_SUCCESS	Terminated normally

Note

No more than five units can be connected to PC.

3.5.4　CubBoardID

Format

INT CubBoardID（HANDLE CUBHandle）；

Function

Obtains the board ID of CUB-43PCIEXP

Parameter

HANDLE CUBHandle	Handle value of CUB-43PCIEXP

Return value

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

Failed : -1 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	Called with invalid handle
CUB_ERR_FAILED	The process failed due to unknown reason.

3.5.5　CubResetBoard

Format

BOOL CubResetBoard（HANDLE CUBHandle）；

Function

Resets MKY43 of specified CUB-43PCIEXP board.

Parameter

HANDLE CUBHandle	Handle value of CUB-43PCIEXP

Return value

Succeeded : TRUE is returned.

Failed : FALSE is returned.

Error code

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

CUB_SUCCESS	Terminated normally
CUB_ERR_INVALIDPARAM	Called with invalid handle
CUB_ERR_FAILED	The process failed due to unknown reason.

3.5.6 CubSearchBoard

Format

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

Function

Obtains the number of CUB-43PCIEXP connected to PC and its board ID list.

Parameter

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

Return value

Succeeded : TRUE is returned.

Failed : FALSE 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	*board_num is NULL.
CUB_ERR_FAILED	The process failed due to unknown reason.

Note

Board ID is set by SW5.
If two or more CUB-43PCIEXP are connected to PC, each unit can be distinguished by board IDs.
This API can distinguish up to four CUB-43PCIEXP devices.
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-43PCIEXP units are connected to PC, and each board IDs are set in sequence ; 1st board ID = 0, 2nd board ID = 1, 3rd board ID = 2.

If the units 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 handles to CUB-43PCIEXP

Parameter

int index_no	Index number You can specify an index number from 0 to 3. If just one CUB-43PCIEXP is connected to PC, set '0'. For more information, see "Note".

Return value

Succeeded : 1 or greater value is returned.

Failed : -1（INVALID_HANDLE_VALUE）is returned.

Error code

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

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

Note

Close handle by CubCloseHandle at finishing the program.

It's not necessary to run CubSearchBoard when just one CUB-43PCIEXP is connected to PC.
If two or more CUB-43PCIEXP are connected to PC, you must run "CubSearchBoard" in advance to check which CUB-43PCIEXP to manipulate.

As an example, three CUB-43PCIEXP units are connected to PC, and each board IDs are set in sequence ;
1st board ID = 0, 2nd board ID = 1, 3rd board ID = 2.
To obtain the handle value of Board ID = 2, operate as follows.

```
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.

3.5.8 CubCloseHandle

Format

BOOL CubCloseHandle（HANDLE CUBHandle）;

Function

Closes handle obtained by CubOpenHandle.

Parameter

HANDLE CUBHandle	Handle value of CUB-43PCIEXP

Return value

Succeeded : TRUE is returned.
Failed : FALSE 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	Called with invalid handle
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 of data from the specified address.

Parameter

HANDLE CUBHandle	The handle value of CUB-43PCIEXP
const ULONG Adr	Address value Input condition is the following ・Input range : 0x0000 - 0x0FFF
BYTE *Dat	The storage address of read data

Return value

Succeeded : TRUE is returned.
Failed : FALSE 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	Called with invalid handle 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	Handle value of CUB-43PCIEXP
const ULONG Adr	Address value Input condition is the following. ・Input range : 0x0000 - 0x0FFF
const WORD Dat	Write data

Return value

Succeeded : TRUE is returned.

Failed : FALSE is returned.

Error code

The error codes and error factors returned by CubGetLastError after executing this function are the following.

CUB_SUCCESS	Terminated normally
CUB_ERR_INVALIDPARAM	Called with invalid handle 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	Handle value of CUB-43PCIEXP
const ULONG Adr	Address value Input conditions are the following. ・Multiples of 2 ・Input range : 0x0000 - 0x0FFF
WORD *Dat	The storage address of read data

Return value

Succeeded : TRUE is returned.

Failed : FALSE s 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	Called with invalid handle Adr is out of range. Adr value is not 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	Handle value of CUB-43PCIEXP
const ULONG Adr	Address value Input conditions are the following. ・Multiples of 2 ・Input range : 0x0000 - 0x0FFE
const WORD Dat	Write data

Return value

Succeeded : TRUE is returned.

Failed : FALSE is returned.

Error code

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

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

3.5.13 CubGetInterrupt0Count、CubGetInterrupt1Count

Format

BOOL CubGetInterrupt0Count（HANDLE CUBHandle, BYTE *int0Counter）;

BOOL CubGetInterrupt1Count（HANDLE CUBHandle, BYTE *int1Counter）;

Function

Obtains the information of INT0, 1 interrupt count register retained in internal driver.

Interrupt count is incremented from 0 to 255（0xFF）.

Parameter

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

Return value

Succeeded : TRUE is returned.

Failed : FALSE is returned.

Error code

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

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

3.5.14　CubClearInterrupt0Count、CubClearInterrupt1Count

Format

BOOL CubClearInterrupt0Count（HANDLE CUBHandle）；
BOOL CubClearInterrupt1Count（HANDLE CUBHandle）；

Function

Clears the information of INT0, 1 interrupt count register retained in internal driver.

Parameter

HANDLE CUBHandle	Handle value of CUB-43PCIEXP

Return value

Succeeded : TRUE is returned
Failed : FALSE is returned

Error code

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

CUB_SUCCESS	Terminated normally
CUB_ERR_INVALIDPARAM	Called with invalid handle
CUB_ERR_FAILED	The process failed due to unknown reason.

3.5.15　CubGetInterrupt0StatusInfo、CubGetInterrupt1StatusInfo

Format

BOOL CubGetInterrupt0StatusInfo（HANDLE CUBHandle,WORD *int0Info）；
BOOL CubGetInterrupt1StatusInfo（HANDLE CUBHandle,WORD *int1Info）；

Function

Obtains the information of INT0, 1 interrupt factor retained in internal driver.

Parameter

HANDLE CUBHandle	Handle value of CUB-43PCIEXP
WORD * int0Info、*int1Info	The storage address of interrupt factor information.

Return value

Succeeded : TRUE is returned
Failed : FALSE is returned

Error code

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

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

Note

The configuration of parameters set to int0Info, int1Info is shown in Table 3-4.

When interrupt has occurred, the bit which corresponds to its factor turned to "1".

The arrangements of interrupt factors are same as INT0SR and INT1SR of MKY43.

Table 3-4 Internal configuration of int0Info and int1Info

bit	Interrupt factor
15	An interrupt occurs when a jammer is detected.
14	An interrupt occurs when a PING instruction is received from other CUnet stations.
13	An interrupt occurs when a resize overlap occurs.
12	An interrupt occurs when break packets sent from other CUnet stations are received.
11	An interrupt occurs by the result of “Link NG（No Good）”.
10	An interrupt occurs by the result of “Link OK”.
9	An interrupt occurs when the number of bits at “1” in the MFR（Member Flag Register）increases or decreases.
8	An interrupt occurs when the phase changes to the RUN phase.
7	An interrupt occurs when the network stops.
6	An interrupt occurs when the resizing of a self-station requested from other CUnet stations is completed.
5	An interrupt occurs by the result of “MGR > MFR”.
4	An interrupt occurs by the result of “MGR ≠ MFR”.
3	An interrupt occurs when mail sending is terminated（correctly or incorrectly）.
2	An interrupt occurs when mail reception is completed.
1	An interrupt occurs when the data transition of the Memory Block（MB）corresponding to the detection bit preset to the DRCR（Data Renewal Check Register）is detected at the time prespecified to the IT0CR（Interrupt Timing 0 Control Register）.
0	An interrupt occurs when the station time during cycles reaches the time prespecified to the IT0CR（Interrupt Timing 0 Control Register）.

3.5.16　CubClearInterrupt0StatusInfo、CubClearInterrupt1StatusInfo

Format

BOOL CubClearInterrupt0StatusInfo（HANDLE CUBHandle, WORD clearInt0Info）；
BOOL CubClearInterrupt1StatusInfo（HANDLE CUBHandle, WORD clearInt1Info）；

Function

Clears the specified interrupt factor of INT0, 1 interrupt factors retained by internal driver.

Parameter

HANDLE CUBHandle	Handle value of CUB-43PCIEXP
WORD clearInt0Info、clearInt1Info	Specify the interrupt factor to be cleared.

Return value

Succeeded : TRUE is returned.
Failed : FALSE 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	Called with invalid handle.
CUB_ERR_FAILED	The process failed due to unknown reason.

Note

Interrupt factors and configuration of setting values are shown in Table 3-5.
Set the value which is corresponded to the interrupt factors to clearInt0Info, clearInt1Info.
To clear the multiple interrupt factors, set the logical disjunction（OR）of each setting values.

Table 3-5 Interrupt factors to be cleared and setting values

Interrupt factor	Setting value
Clear an interrupt by jammer detect	0x8000
Clear an interrupt by PING instruction receiving	0x4000
Clear an interrupt by resize overlap occurrence	0x2000
Clear an interrupt by break packet receiving	0x1000
Clear an interrupt by the result of "LINK NG"	0x0800
Clear an interrupt by the result of "LINK OK"	0x0400
Clear an interrupt by incleasing or decleasing of MFR bit number "1"	0x0200
Clear an interrupt by transfer to RUN phase	0x0100
Clear an interrupt by network stop	0x0080
Clear an interrupt by resize complete	0x0040
Clear an interrupt by the result of MGR > MFR	0x0020
Clear an interrupt by the result of MGR ≠ MFR	0x0010
Clear an interrupt by mail sending completion	0x0008
Clear an interrupt by mail receiving completion	0x0004
Clear an interrupt by data renewal	0x0002
Clear an interrupt by ALM	0x0001

3.6 Sample program

3.6.1 Access sample to MKY43

The following describes a sample program to initialize, set CUnet communication mode and obtain the value of global memory using this API.

```
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 < 0x200 || api_version > 0x299) {
        printf (" This version of cub43pciexp.dll has no compatibility. \n") ;
        exit (1) ;
    }

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

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

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

    // (3) setting the communication mode
    // (3-1) Switching ON the GMM function to write to BCR
    CubWriteWord (CUBHandle, 0x366, 0x8000) ;
    // (3-2) Set the communication mode to BCR.
    //    In this sample program, SA=0, OWN=1, BPS=6Mbps are set to BCR.
    CubWriteWord (CUBHandle, 0x356, 0x0180) ;
    // (3-3) Switch OFF the GMM function.
    CubWriteWord (CUBHandle, 0x366, 0x0000) ;
    /** Start CUnet communication */
    CubWriteWord（CUBHandle, 0x366, 0x0100）;

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

        }
        // Reading SA1 global memory
        CubReadWord（CUBHandle, 0x0008, &sa1_gm[0]）;
        CubReadWord（CUBHandle, 0x000A, &sa1_gm[1]）;
        CubReadWord（CUBHandle, 0x000C, &sa1_gm[2]）;
        CubReadWord（CUBHandle, 0x000E, &sa1_gm[3]）;
        // Reading data of SA63 global memory
        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　Interrupt handling sample

The following describes a sample program to check the interrupt setting and interrupt occurrence 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 a 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 registers */
   CubClearInterrupt0StatusInfo（CUBHandle, 0xffff）;

   /* Clear the interrupt count register */
   CubClearInterrupt0Count（CUBHandle）;
   int0_lastTime_numOfOccurr = 0;      // Interrupt count is 0.

   /* Set the interrupt factor. INT0 interrupt occurs when network has stopped. */
   CubWriteWord（CUBHandle, 0x358, 0x0080）;

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

   while (1) {
      /* Obtain the information of interrupt count register */
     CubGetInterrupt0Count（CUBHandle, &int0_current_numOfOccurr）;
      /* An interrupt is occurring if it does not match with previous interrupt count. */
      if (int0_lastTime_numOfOccurr != int0_current_numOfOccurr) {
         /* Copy current value to the previous value */
         int0_lastTime_numOfOccurr = int0_current_numOfOccurr;
         /* Obtain the information of an interrupt factor register. */
         CubGetInterrupt0StatusInfo（CUBHandle, &int0_factor）;
         /* Make sure the interrupt factor is CHECK-1. */
         if ((int0_factor & 0x0080) == 0x0080) {
            /* --- Here descrices the process when the network stop has occured. --- */
            /* Clear INT0 interrupt factor register */
            CubClearInterrupt0StatusInfo（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 Express board

CUB-43PCIEXP

User's Manual

Document No.：STD_CUB43PCIEXP_V2.0E
Issued：Aug 2018