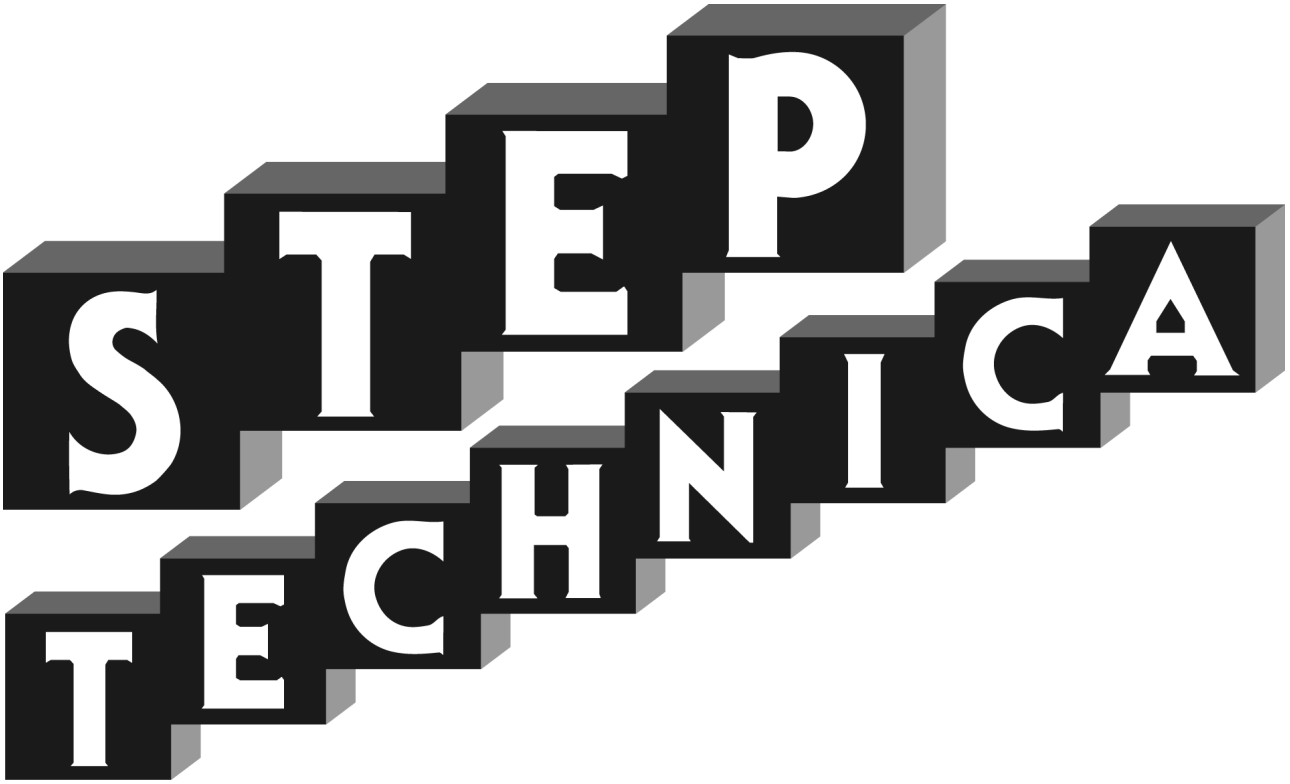
STD_CUB43PCI1_V1.0J

STEP TECHNICA CO.,LTD.



CUnet（MKY43）搭載 PCI ボード

CUB-43PCI1

ユーザーズマニュアル

ご注意

1. 本書に記載された内容は、将来予告なしに変更する場合があります。本製品をご使用になる際には、本書が最新の版数であるかをご確認ください。

2. 本書において記載されている説明や回路例などの技術情報は、お客様が用途に応じて本製品を適切にご利用をいただくための参考資料です。実際に本製品をご使用になる際には、基板上における本製品の周辺回路条件や環境を考慮の上、お客様の責任においてシステム全体を十分に評価し、お客様の目的に適合するようシステムを設計してください。当社は、お客様のシステムと本製品との適合可否に対する責任を負いません。

3. 本書に記載された情報、製品および回路等の使用に起因する損害または特許権その他権利の侵害に関して、当社は一切その責任を負いません。

4. 本製品および本書の情報や回路などをご使用になる際、当社は第三者の工業所有権、知的所有権およびその他権利に対する保証または実施権を許諾致しません。

5. 本製品は、人命に関わる装置用としては開発されておりません。人命に関わる用途への採用をご検討の際は、当社までご相談ください。

6. 本書の一部または全部を、当社に無断で転載および複製することを禁じます。

はじめに

本マニュアルは、CUnet 専用 IC である MKY43 を 1 個搭載した PCI ボードの CUB-43PCI1 について記述します。
CUB-43PCI1 の利用および本マニュアルの理解に先駆けて、“CUnet 導入ガイド ” を必ずお読みください。

●対象読者
・CUnet を初めて構築する方
・CUnet を構築するために、CUB-43PCI1 を初めてご利用になる方

●読者が必要とする知識
・ネットワーク技術に関する標準的な知識
・半導体製品（特にマイクロコントローラおよびメモリ）に関する標準的な知識

●関連マニュアル
・CUnet　導入ガイド
・CUnet　テクニカルガイド
・CUnet　MKY43 ユーザーズマニュアル

【注意事項】
本書において記載されている一部の用語は、弊社の Web および営業用ツール（総合カタログ等）において記載されている用語とは異なっています。営業用ツールにおいては、様々な業界において弊社製品をご理解いただけるよう、一般的用語を用いています。
CUnet ファミリに関する専門知識は、技術ドキュメント（マニュアル等）を基にご理解ください。

改訂履歴

Ver	日 付	改 訂 内 容	
		ページ	説 明
Ver1.0J	2018/11/9	-	初版発行

目　次

第１章　製品概要

第２章　ハードウェア

第３章　ソフトウェア

図　目　次

表　目　次

第 1 章　製品概要

本章は、CUB-43PCI1 の製品概要について記述します。

1.1 特徴

CUB-43PCI1 は、ステップテクニカ製 MKY43 チップを搭載した PCI 拡張バス対応の CUnet 通信ボードです。ステップテクニカ提供の Windows 用の API と併せて利用することにより、MKY43 の機能を簡単に利用することが可能です。

1.2 仕様

CUB-43PCI1 の仕様を、表 1-1 に示します。

表 1-1　仕様

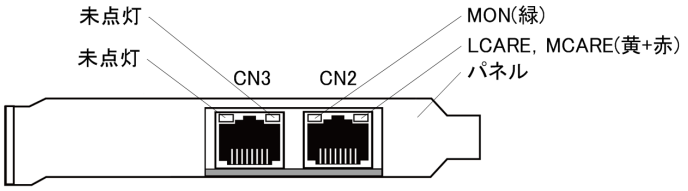
CUnet デバイス	MKY43　1 個
CUnet 通信方式	半二重通信
CUnet 通信速度	12M/6M/3Mbps（MKY43 レジスタにて設定）
CUnet 通信コネクタ	RJ45 タイプ（8pin モジュラー）× 2 個
対応バス	PCI Ver2.2 準拠した、32 ビット・33MHz 拡張バス 5.0V/3.3V 対応
占有リソース	16KB の連続したメモリエリア（PnP にて自動割当）
割り込み	1 ライン使用（PnP にて自動割当）
対応 OS	Windows10　（64bit/32bit） Windows8.1　（64bit/32bit） Windows8　（64bit/32bit） Windows7　（64bit/32bit）
電源	DC +5.0V
消費電流	500mA 以下
動作環境	温度　0 ～ 50℃　湿度　20 ～ 90%（非結露）
保存環境	温度　0 ～ 80℃　湿度　0 ～ 90%（非結露）
外形寸法	119.9mm(W) × 64.4mm(D) ※パネル部含まず（Low Profile 対応）
付属品	Low Profile 用ブラケット

第２章　ハードウェア

本章は、CUB-43PCI1 のハードウェアについて記述します。

2.1 コネクタ仕様

CUB-43PCI1 のパネル面とその詳細を図 2-1 に示します。



ピン番号	機　能
4	TRX-
5	TRX+
8	シールド
1,2,3,6,7	未使用

MON	MKY43 の MON 端子に接続した LED（緑） 他の CUnet ステーションと通信が成立している間点灯
LCARE MCARE	MKY43 の #LCARE 端子に接続した LED（黄）または #MCARE 端子に接続した LED（赤）を表示 LCARE または MCARE 発生時に点灯 LCARE と MCARE が同時に発生した場合は LED（赤）が点灯
CN2、CN3	CUnet 通信ライン 市販の 100BASE-TX 用 CAT-5 以上のストレートケーブルが使用可能
パネル	パネルは、PC 本体の筐体と接触 PC 本体の筐体は PC 取扱説明書に従って F.G へ接続 基板上の R33 パットおよび R43 パットをショートさせることにより、CN2 および CN3 の金属シェルおよび 8 番端子をパネルへ接続することが可能（図 2-2、図 2-3 参照）

図 2-1　パネル概観

CN2、CN3 コネクタ周辺回路を図 2-2 に示します。

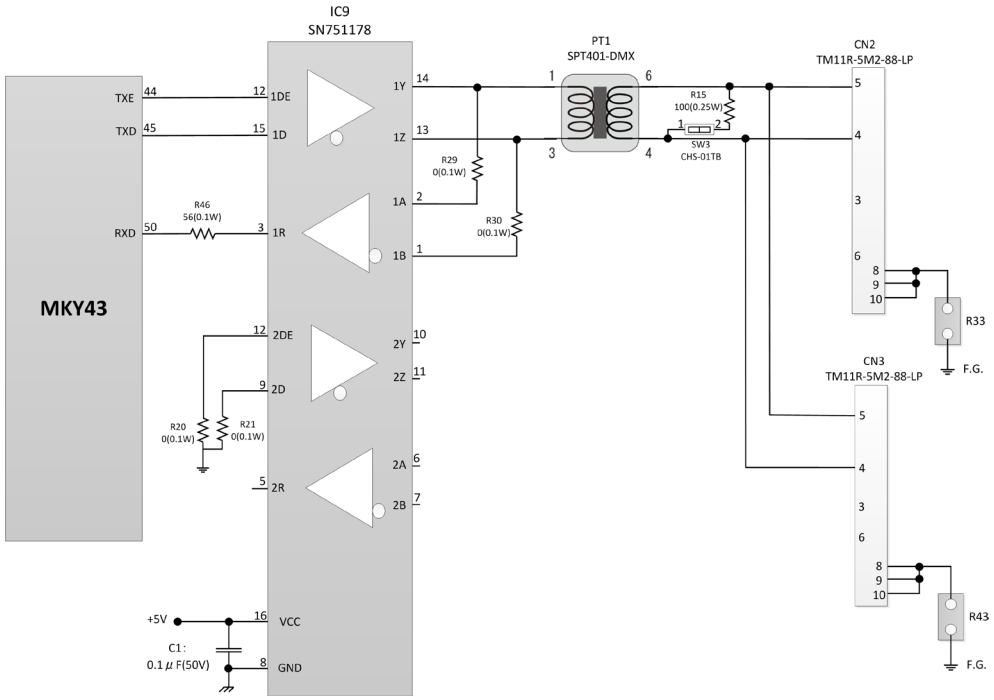


図 2-2　コネクタ周辺回路

2.2 ディップスイッチ

CUB-43PCI1 のディップスイッチの設定を図 2-3 に示します。

複数の CUB-43PCI1 を同一の機器に搭載する場合には、SW9 のボード ID を設定してください。このボード ID によって、ソフトウェアから特定の CUB-43PCI1 を見つけることができます。（工場出荷時設定 ボード ID：0）

CUB-43PCI1 がマルチドロップ接続の終端位置（通信ケールの端）になる場合は、SW3 を ON に設定し、ターミネーションを有効にしてください。
（工場出荷時設定 ターミネーション：OFF）

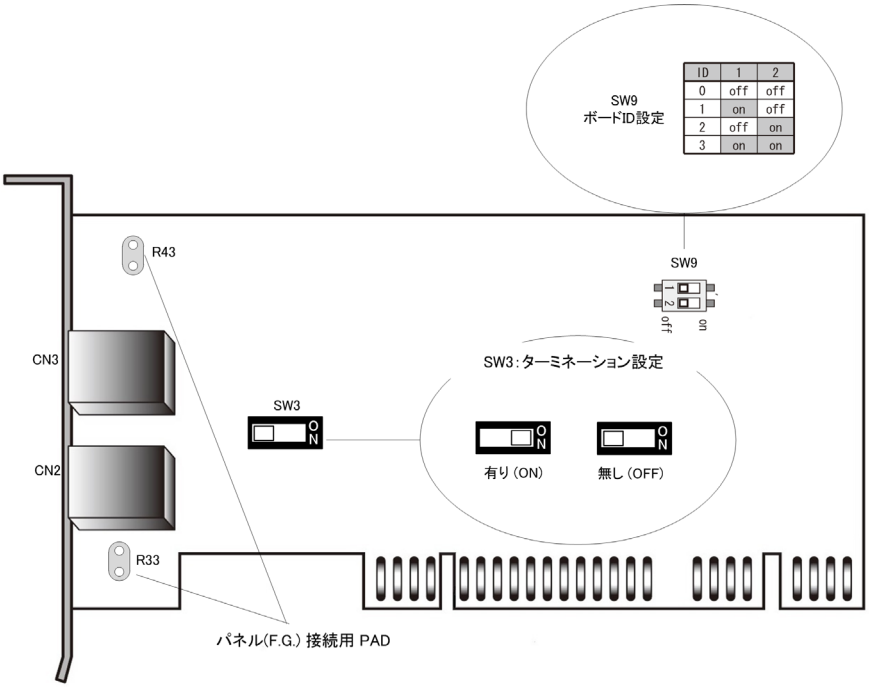


図 2-3　CUB-43PCI1 ボードの設定

2.3 メモリマップ

CUB-43PCI1 のメモリマップを表 2-1 に示します。メモリマップ中のアドレスは CUB-43PCI1 の先頭アドレスからの相対値であり、実際のアドレスはボードの先頭アドレス値を加算したアドレスになります。

表 2-1　メモリマップ

アドレス	概　要
000H ～ 5FFH	MKY43
600H ～ EFFH	未使用
F00H	Chip Reset Register
F02H	Board ID Register
F04H ～ FFFH	未使用

2.3.1　MKY43

MKY43 のメモリマップについては「MKY43 ユーザーズマニュアル」の「第 4 章 MKY43 のソフトウエア」、「4.1.1 メモリマップ」をご参照ください。

2.3.2 CUB-43PCI1 独自のレジスタ

表 2-1 のメモリマップに示された F00H および F02H のレジスタは、CUB-43PCI1 独自のレジスタです。
以下に、そのレジスタの詳細を記載します。

Chip Reset Register　　アドレス：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
機能	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	CRST

[機能説明] CRST（Chip ReSeT ）へ “1” をライトすることにより､MKY43 の RST 端子へリセット信号を印加します。RST 端子へのリセット期間は、100ms です。また、本レジスタは書き込み専用レジスタの為、読み込みを行った場合のデータは不定です。

Board ID Register　　アドレス：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
機能	--	--	--	--	--	--	--	--	--	--	--	--	--	--	BID1	BID0

[機能説明] BID0,1（Board ID）ビットをリードすることにより、SW9 によって設定されたボード ID の値を取得することができます。詳細については、“2.2 ディップスイッチ ” を参照ください。

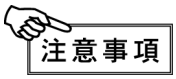
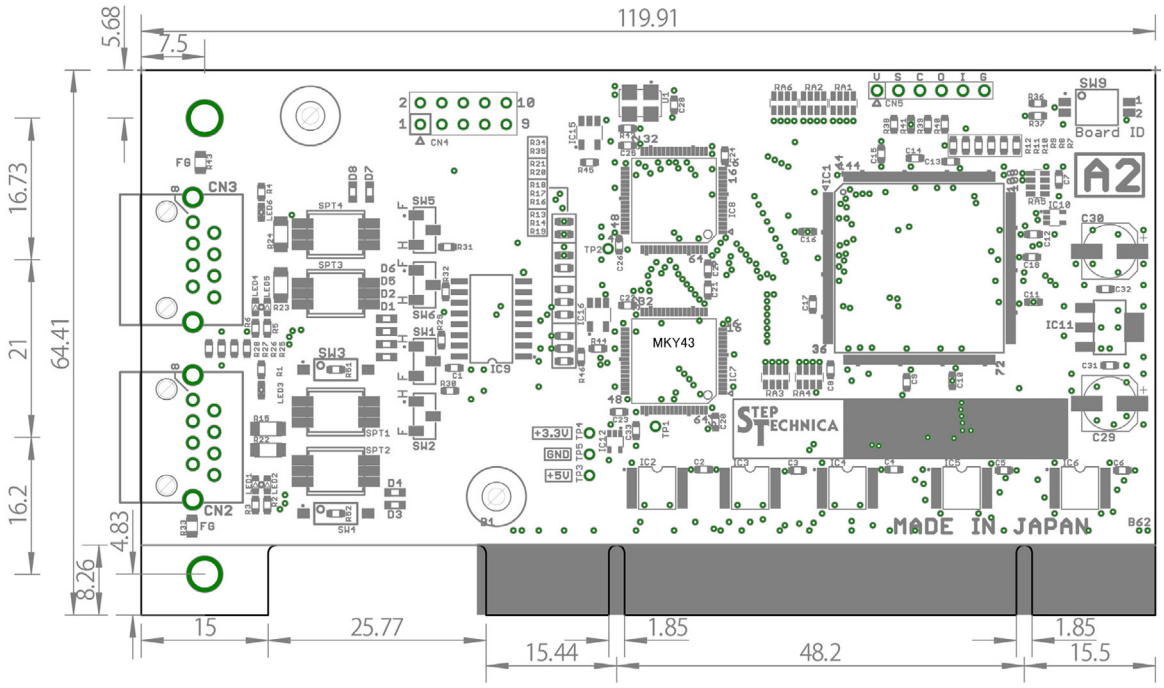


表 2-1 のメモリマップに示されている未使用領域（“600H ～ EFFH” , ”FO4H ～ FFFH”）はアクセスしないでください。システムを不安定にする可能性があります。

2.4 寸法図


119.91
5.68
7.5
16.73
21
64.41
16.2
4.83
8.26
15
25.77
15.44
1.85
48.2
1.85
15.5
CN3
CN2
CN4
CN5
SW9
Board ID
A2
SW1
SW2
SW3
SW4
SW5
SW6
MKY43
MADE IN JAPAN
+3.3V
GND
+5V

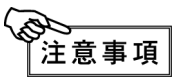
第３章　ソフトウェア

本章は、ステップテクニカ社提供の API について記述します。
本マニュアルは、API バージョン「1.0.0」に基づいて記述しています。
製品のご使用にあたっては、弊社ホームページにより最新の情報をご確認してください。

3.1 概要

ユーザアプリケーションから CUB-43PCI1 へアクセスするための API を用意しています。
下記ステップテクニカ社のダウンロードページより API をダウンロードしてください。
URL：http://www.steptechnica.com/jp/download/index.html
対応 OS は
・Windows10 （64bit/32bit）
・Windows 8.1（64bit/32bit）
・Windows 8 （64bit/32bit）
・Windows 7 （64bit/32bit）
です。
提供している API は、Microsoft Visual Studio や VB6 などから呼び出すことができます。

3.2 著作権・免責



① 弊社 web ページより入手した全てのソフトウェアの使用による、いかなる結果に対しても弊社は一切責任を負いません。
② API は、説明に従って正しくお使いください。
③ 仕様・内容は、将来予告無く変更になる場合があります。弊社は、将来への互換性について、一切保証いたしません。
④ OS や開発環境等に関するお問い合わせはサポートできません。
⑤ バグ・不具合などを発見された方は、弊社システム開発部までご連絡ください。

3.3 ファイル構成

“DLL” フォルダに収められたファイルは以下のとおりです。

【cub43pci1.dll】

DLL 本体です。Windows のシステムフォルダ、もしくは、本 DLL を使用するユーザプログラムと同じディレクトリにコピーしてお使いください。

【cub43pci1.lib】

インポートライブラリです。

【cub43pci1.h】

DLL のヘッダファイルです。Windows.h より後ろにインクルードしてください。

3.4 制限事項

ここでは、本 API を使用してアプリケーションを作成する際の制限事項について記します。

3.4.1 マルチスレッド

本 API 関数は複数スレッドから同時に使用することはできません。
アプリケーションをマルチスレッド構成にする場合、同時呼び出しが起こらないように配慮して下さい。

3.4.2 省電力モードについて

CUB-43PCI1 は、省電力モード機能に対応しておりません。
OS のスリープ機能を停止したうえでご使用ください。スリープに入った場合には、搭載されている MKY43 への電源供給が遮断され、通信が停止します。また、省電力モードからの復帰時には、リセットがかかる為、各レジスタは初期化され、GM、MSB、MRB0、MRB1 領域は不定の状態になりますのでご注意ください。

3.4.2 割り込み処理

MKY43 では、INT0SR と INT1SR レジスタにより割込み発生状況の確認が行えます。
ドライバ内部では、割込み発生時の INT0SR、INT1SR の情報を保持するレジスタ（割込み発生要因レジスタ）と INT0、INT1 それぞれで割込みが発生した回数を保持するレジスタ（割込み発生回数レジスタ）を管理しています。
ドライバ内部では、割込み発生時にこれらのレジスタを使用して次の処理を行います。
（ここでは、INT0 での割込み発生した場合の説明を記します。）

① 割込み発生要因レジスタに割込み発生要因情報をセットします。
（ユーザアプリケーションから割込み発生要因レジスタのクリア指示があるまで過去の割込み発生要因が残った状態でセットされます。）
② 割込み発生回数レジスタの値をインクリメントする。
③ INT0SR の 0~15bit の内で “1” となっている箇所に “1” をライトし、割込み発生要因のクリアを行います。

割込み発生要因レジスタと割込み発生回数レジスタから情報の取得とクリアを行う API 関数を用意しています。

(1) 割込み発生回数レジスタから割り込み発生回数を取得（CubGetInterrupt0Count, CubGetInterrupt1Count）
ドライバ内で MKY43 からの INT0、INT1 それぞれの割込み発生回数を割込み発生回数レジスタで保持しています。
本 API 関数では、割り込み発生回数レジスタから情報を取得します。

(2) 割込み発生回数レジスタのクリア（CubClearInterrupt0Count, CubClearInterrupt1Count）
割込み発生回数レジスタをクリアします。

(3) 割込み発生要因レジスタの情報を取得（CubGetInterrupt0StatusInfo, CubGetInterrupt1StatusInfo）
ドライバ内部で MKY43 からの INT0、INT1 それぞれの割込み発生要因を割込み発生要因レジスタで保持しています。
本 API 関数では、割込み発生要因レジスタから情報を取得します。

(4) 割込み発生要因レジスタのクリア関数（CubClearInterrupt0StatusInfo, CubClearInterrupt1StatusInfo）
指定した割込み発生要因を割込み発生要因レジスタからクリアします。

ユーザアプリケーションでは、これらの関数を使用して MKY43 からの割込み発生回数と割込み発生要因の確認を行ってください。

3.4.3 提供ドライバを使用しない場合のアクセスについて

ステップテクニカ社製ドライバを使用せずに CUB-43PCI1 へ直接アクセスする場合には、以下の点について注意が必要です。

CUB-43PCI1 には、常に 32bit アクセスを行ってください。その時、下位 16bit データが有効となり、上位 16bit は使用されません。その為にアクセスするアドレスは上記メモリマップの 2 倍を指定する必要があります。
例えば、MKY43 のアドレス 200H を Read する場合、CUB-43PCI1 の 400H を 32bit Read することで、その下位 16bit に MKY43 の 200H の 2 バイトデータが取得できます。CUB-43PCI1 独自レジスタに関しても同様のアクセスが必要です。

3.5 API 仕様

API の仕様について記述します。
本 API は、CUB-43PCI1 をユーザアプリケーションから簡単に操作することを目的として用意しています。

API 関数の一覧を表 3-1 に示します。

表 3-1　API 関数

関　数	機能概要
CubGetVersion	API のバージョン番号を取得
CubGetLastError	API 関数の終了状態を取得
CubCountDevice	パソコンに装着している CUB-43PCI1 の枚数を取得
CubBoardID	ボード ID を取得
CubResetBoard	指定した MKY43 へリセットを指示
CubSearchBoard	CUB-43PCI1 の枚数とそのボード ID を取得
CubOpenHandle	CUB-43PCI1 のハンドルをオープン
CubCloseHandle	CUB-43PCI1 のハンドルをクローズ
CubReadByte	指定したアドレスから 1 バイトのデータ読み込み
CubWriteByte	指定したアドレスへ 1 バイトのデータ書き込み
CubReadWord	指定したアドレスから 2 バイトのデータ読み込み
CubWriteWord	指定したアドレスへ 2 バイトのデータ書き込み
CubGetInterrupt0Count CubGetInterrupt1Count	ドライバ内部で保持している INT0、1 割込み発生回数を取得
CubClearInterrupt0Count CubClearInterrupt1Count	ドライバ内部で保持している INT0、1 割込み発生回数をクリア
CubGetInterrupt0StatusInfo CubGetInterrupt1StatusInfo	ドライバ内部で保持している INT0、1 割込み発生要因を取得
CubClearInterrupt0StatusInfo CubClearInterrupt1StatusInfo	指定した割込み発生要因をドライバ内部で保持している INT0、1 割込み要因情報からクリア

3.5.1 CubGetVersion

書式

UINT CubGetVersion(void);

機能

API のバージョン番号を取得します。

パラメータ

なし

リターンパラメータ

API のバージョン番号（BCD コード 16 進数）
（メジャー番号 + マイナー番号 + アップデート番号）

エラーコード

本関数実行後に CubGetLastError が返すエラーコードとエラー発生要因は以下の通りです。
CUB_SUCCESS　　正常終了

注釈

API のバージョン番号は、表 3-2 の構成です。
バージョン番号がアップされる原因は、下記のとおりです。

メジャー番号　　：API の仕様変更など、互換性を保てなかった変更がなされた時に変わります。
マイナー番号　　：API 関数の追加など、互換性を保ったままの変更がなされた時に変わります。
アップデート番号：不具合修正など、仕様に影響が無く修正された時に変わります。

表 3-2　バージョン番号の構成

戻り値（例）	メジャー番号（ビット 15 ～ 8）	マイナー番号（ビット 7 ～ 4）	アップデート番号（ビット 3 ～ 0）
0x0102	1	0	2
0x1398	13	9	8

3.5.2 CubGetLastError

書式

UINT CubGetLastError(void);

機能

最後に呼び出された API 関数の終了状態を取得します。

パラメータ

なし

リターンパラメータ

cub43pci1.h で定義しているエラーコードを返します。

注釈

表 3-3 に cub43pci1.h で定義しているエラーコードを記します。

表 3-3 エラーコードリスト

文字定数	値	内 容
CUB_SUCCESS	0	正常終了
CUB_ERR_DEVICENOTEXIST	1	デバイスが存在しない
CUB_ERR_ALREADYOPENED	2	すでにオープンされている
CUB_ERR_CLOSED	3	CubOpenHandle が一度もコールされていない
CUB_ERR_INVALIDPARAM	4	パラメータが間違っている
CUB_ERR_NORESOUCE	5	実行に必要なリソースが足りない
CUB_ERR_FAILED	6	原因不明により処理が遂行されなかった
CUB_NOTCALLYET	99	API 関数が一度もコールされていない

3.5.3 CubCountDevice

書式

INT CubCountDevice(void);

機能

パソコンに装着している CUB-43PCI1 の枚数を返します。

パラメータ

なし

リターンパラメータ

CUB-43PCI1 の枚数を返します。

-1	： 5 枚以上
0	： 1 枚も装着されていない
1 ～ 4	： 1 枚～ 4 枚

エラーコード

本関数実行後に CubGetLastError が返すエラーコードとエラー発生要因は以下の通りです。

CUB_SUCCESS 正常終了

注釈

パソコンに 5 枚以上装着することはできません。

3.5.4 CubBoardID

書式

INT CubBoardID(HANDLE CUBHandle);

機能

CUB-43PCI1 のボード ID を取得します。

パラメータ

HANDLE CUBHandle	CUB-43PCI1 のハンドル値

リターンパラメータ

正常終了時はボード ID（0 ～ 3）を返します。失敗時は -1 を返します。

エラーコード

本関数実行後に CubGetLastError が返すエラーコードとエラー発生要因は以下の通りです。

CUB_SUCCESS	正常終了
CUB_ERR_INVALIDPARAM	ハンドル値が無効
CUB_ERR_FAILED	原因不明により処理が遂行されなかった

3.5.5 CubResetBoard

書式

BOOL CubResetBoard (HANDLE CUBHandle);

機能

MKY43 をリセットします。

パラメータ

HANDLE CUBHandle	CUB-43PCI1 のハンドル値

リターンパラメータ

正常終了時は TRUE(1)、失敗時は FALSE(0) を返します。

エラーコード

本関数実行後に CubGetLastError が返すエラーコードとエラー発生要因は以下の通りです。

CUB_SUCCESS	正常終了
CUB_ERR_INVALIDPARAM	ハンドル値が無効
CUB_ERR_FAILED	原因不明により処理が遂行されなかった

補足

リセット後、100ms 以上待ってから MKY43 へのアクセスを行ってください。

3.5.6 CubSearchBoard

書式

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

機能

パソコンに装着されている CUB-43PCI1 の枚数とそのボード ID リストを返します。

パラメータ

*board_num	ボード枚数がセットされるバイト型変数へのアドレスを指定します。 セットされた値の意味は以下の通りです。 ・-1 ： 5 枚以上確認された ・0 ： 1 枚もない ・1 ～ 4 ： 認識したボード枚数
*board_id_list	ボード ID を受け取る為に、バイト型を 4 要素持つ配列のポインタを指定します。または NULL を指定することも可能です。 NULL が指定された場合は、ボード枚数のみを返します。 セットされた値の意味は以下の通りです。 ・0x00 ～ 0x03 ： ボード ID ・0xFF ： 認識できなかった

リターンパラメータ

正常終了時は TRUE(1)、失敗時は FALSE(0) を返します。

エラーコード

本関数実行後に CubGetLastError が返すエラーコードとエラー発生要因は以下の通りです。

CUB_SUCCESS	正常終了
CUB_ERR_INVALIDPARAM	*board_num に NULL が指定された
CUB_ERR_FAILED	原因不明により処理が遂行されなかった

補足

ボード ID は、SW9 により設定します。
CUB-43PCI1 が複数枚装着している場合は、ボード ID により識別することができます。
本 API 関数では、最大 4 枚までの CUB-43PCI1 を識別できます。
下記のようにバイト型配列をパラメータとして指定してください。

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

動作例として、パソコンに 3 枚の CUB-43PCI1 が装着されており、それぞれのボード ID が
1 枚目：ボード ID=0、2 枚目：ボード ID=1、3 枚目：ボード ID=2 と設定されています。
パソコンが認識した順番が 1 枚目、3 枚目、2 枚目となっている状況で CubSearchBoard が実行された場合

board_num = 3;
board_id_list[0] = 0、board_id_list[1] = 2、board_id_list[2] = 1、board_id_list[3] = 0xFF

と返します。

3.5.7 CubOpenHandle

書式

HANDLE CubOpenHandle(int index_no);

機能

CUB-43PCI1 のハンドルをオープンします。

パラメータ

int index_no	インデックス番号 インデックス番号には、0 ～ 3 が指定できます。 CUB-43CPI1 が 1 枚の場合は、0 をセットしてください。 詳しくは、“ 補足 “ を参照してください。

リターンパラメータ

正常終了時は、1 以上の値を返します。失敗時は -1（INVALID_HANDLE_VALUE）を返します。

エラーコード

本関数実行後に CubGetLastError が返すエラーコードとエラー発生要因は以下の通りです。

CUB_SUCCESS	正常終了
CUB_ERR_DEVICENOTEXIST	デバイスが存在しない
CUB_ERR_FAILED	原因不明により処理が遂行されなかった

補足

CUB-43PCI1 が 1 枚の場合は、CubSearchBoard を実行する必要がありません。
CUB-43PCI1 が複数枚装着している場合は、“CubSearchBoard” を先に実行し、操作を行う対象の CUB-43PCI1 を確認しておく必要があります。
例として、パソコンに 3 枚の CUB-43PCI1 が装着されており、それぞれのボード ID が
1 枚目：ボード ID=0、2 枚目：ボード ID=1、3 枚目：ボード ID=2
と設定されています。ここでボード ID=2 のハンドル値を取得する為には

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

を実行した結果、

board_id_list [0] = 0、board_id_list [1] = 2、board_id_list [2] = 1、board_id_list [3] = 0xFF

になったと仮定します。
この場合、インデックス番号 1 がボード ID=2 であることが確認できます。
つまり CubOpenHandle のパラメータであるインデックス番号は、 1 になります。
プログラム終了時には、CubCloseHandle によりハンドルをクローズしてください。

3.5.8 CubCloseHandle

書式

BOOL CubCloseHandle(HANDLE CUBHandle);

機能

CubOpenHandle によって取得したハンドルを閉じます。

パラメータ

HANDLE CUBHandle	CUB-43PCI1 のハンドル値

リターンパラメータ

正常終了時は TRUE(1)、失敗時は FALSE(0) を返します。

エラーコード

本関数実行後に CubGetLastError が返すエラーコードとエラー発生要因は以下の通りです。

CUB_SUCCESS	正常終了
CUB_ERR_INVALIDPARAM	ハンドル値が無効
CUB_ERR_FAILED	原因不明により処理が遂行されなかった

3.5.9 CubReadByte

書式

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

機能

指定したアドレスから 1 バイトのデータを読み込みます。

パラメータ

HANDLE CUBHandle	CUB-43PCI1 のハンドル値
const ULONG Adr	アドレス値 入力条件は以下の通り ・入力範囲：0x0000 ～ 0x0FFE
BYTE *Dat	読み込みデータ格納先のアドレス

リターンパラメータ

正常終了時は TRUE(1)、失敗時は FALSE(0) を返します。

エラーコード

本関数実行後に CubGetLastError が返すエラーコードとエラー発生要因は以下の通りです。

CUB_SUCCESS	正常終了
CUB_ERR_INVALIDPARAM	ハンドル値が無効 Adr が範囲外 *Dat に NULL が指定された
CUB_ERR_FAILED	原因不明により処理が遂行されなかった

3.5.10 CubWriteByte

書式

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

機能

指定したアドレスへ 1 バイトのデータを書き込みます。

パラメータ

HANDLE CUBHandle	CUB-43PCI1 のハンドル値
const ULONG Adr	アドレス値 入力条件は以下の通り ・入力範囲：0x0000 ～ 0x0FFE
const WORD Dat	書き込みデータ

リターンパラメータ

正常終了時は TRUE(1)、失敗時は FALSE(0) を返します。

エラーコード

本関数実行後に CubGetLastError が返すエラーコードとエラー発生要因は以下の通りです。

CUB_SUCCESS	正常終了
CUB_ERR_INVALIDPARAM	ハンドル値が無効 Adr が範囲外
CUB_ERR_FAILED	原因不明により処理が遂行されなかった

3.5.11 CubReadWord

書式

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

機能

指定したアドレスから 2 バイトのデータを読み込みます。

パラメータ

HANDLE CUBHandle	CUB-43PCI1 のハンドル値
const ULONG Adr	アドレス値 入力条件は以下の通り ・2 の倍数 ・入力範囲：0x0000 ～ 0x0FFE
WORD *Dat	読み込みデータ格納先のアドレス

リターンパラメータ

正常終了時は TRUE(1)、失敗時は FALSE(0) を返します。

エラーコード

本関数実行後に CubGetLastError が返すエラーコードとエラー発生要因は以下の通りです。

CUB_SUCCESS	正常終了
CUB_ERR_INVALIDPARAM	ハンドル値が無効 Adr が範囲外 Adr が 2 の倍数でない *Dat に NULL が指定された
CUB_ERR_FAILED	原因不明により処理が遂行されなかった

3.5.12 CubWriteWord

書式

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

機能

指定したアドレスへ 2 バイトのデータを書き込みます。

パラメータ

HANDLE CUBHandle	CUB-43PCI1 のハンドル値
const ULONG Adr	アドレス値 入力条件は以下の通り ・2 の倍数 ・入力範囲：0x0000 ～ 0x0FFE
const WORD Dat	書き込みデータ

リターンパラメータ

正常終了時は TRUE(1)、失敗時は FALSE(0) を返します。

エラーコード

本関数実行後に CubGetLastError が返すエラーコードとエラー発生要因は以下の通りです。

CUB_SUCCESS	正常終了
CUB_ERR_INVALIDPARAM	ハンドル値が無効 Adr が範囲外 Adr が 2 の倍数でない
CUB_ERR_FAILED	原因不明により処理が遂行されなかった

3.5.13 CubGetInterrupt0Count、CubGetInterrupt1Count

書式

BOOL CubGetInterrupt0Count(HANDLE CUBHandle, BYTE *int0Counter);
BOOL CubGetInterrupt1Count(HANDLE CUBHandle, BYTE *int1Counter);

機能

ドライバ内で保持している INT0、1 割込み発生回数レジスタ情報を取得します。
割込み発生回数は、0 から 255（0xFF）までインクリメントし、0 に戻ります。

パラメータ

HANDLE CUBHandle	CUB-43PCI1 のハンドル値
BYTE *int0Counter、*int1Counter	割込み発生回数を格納領域のアドレス

リターンパラメータ

正常終了時は TRUE(1)、失敗時は FALSE(0) を返します。

エラーコード

本関数実行後に CubGetLastError が返すエラーコードとエラー発生要因は以下の通りです。

CUB_SUCCESS	正常終了
CUB_ERR_INVALIDPARAM	ハンドル値が無効 *int0Counter、*int1Counter に NULL が指定された
CUB_ERR_FAILED	原因不明により処理が遂行されなかった

3.5.14 CubClearInterrupt0Count、CubClearInterrupt1Count

書式

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

機能

ドライバ内で保持している INT0、1 割込み発生回数レジスタ情報をクリアします。

パラメータ

HANDLE CUBHandle	CUB-43PCI1 のハンドル値

リターンパラメータ

正常終了時は TRUE(1)、失敗時は FALSE(0) を返します。

エラーコード

本関数実行後に CubGetLastError が返すエラーコードとエラー発生要因は以下の通りです。

CUB_SUCCESS	正常終了
CUB_ERR_INVALIDPARAM	ハンドル値が無効
CUB_ERR_FAILED	原因不明により処理が遂行されなかった

3.5.15 CubGetInterrupt0StatusInfo、CubGetInterrupt1StatusInfo

書式

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

機能

ドライバ内で保持している INT0、1 割込み発生要因情報を取得します。

パラメータ

HANDLE CUBHandle	CUB-43PCI1 のハンドル値
WORD * int0Info、*int1Info	割込み発生要因情報を格納する領域のアドレス

リターンパラメータ

正常終了時は TRUE(1)、失敗時は FALSE(0) を返します。

エラーコード

本関数実行後に CubGetLastError が返すエラーコードとエラー発生要因は以下の通りです。

CUB_SUCCESS	正常終了
CUB_ERR_INVALIDPARAM	ハンドル値が無効 * int0Info、* int1Info に NULL が指定された
CUB_ERR_FAILED	原因不明により処理が遂行されなかった

注釈

int0Info、int1Info へセットされるパラメータの構成を表 3-4 に記します。

割込みが発生した場合、発生要因に対応した bit が “1” となります。

割込み発生要因の配置は、MKY43 の INT0SR、INT1SR と同等です。

表 3-4　intOInfo、int1Info の内部構成

bit	割込み発生要因
15	ジャマー検出による割込み発生
14	PING 命令の受信による割込み発生
13	リサイズオーバラップの発生による割込み発生
12	ブレークパケット受信による割込み発生
11	“ リンク NG” の判定結果による割込み発生
10	“ リンク OK” の判定結果による割込み発生
9	MFR の “1” であるヒット数の増減による割込み発生
8	ランフェーズへ遷移したことによる割込み発生
7	ネットワーク停止による割込み発生
6	リサイズ完了による割込み発生
5	MGR>MFR の判定結果による割込み発生
4	MGR ≠ MFR 判定結果による割込み発生
3	メール送信終了による割込み発生
2	メール受信完了による割込み発生
1	データリニューアルによる割込み発生
0	ALM による割込み発生

3.5.16 CubClearInterrupt0StatusInfo、CubClearInterrupt1StatusInfo

書式

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

機能

指定した割込み要因をドライバ内部で保持している INT0、INT1 割込み発生要因情報から
クリアします。

パラメータ

HANDLE CUBHandle	CUB-43PCI1 のハンドル値
WORD clearInt0Info、clearInt1Info	クリアした割込み発生要因を指定

リターンパラメータ

正常終了時は TRUE(1)、失敗時は FALSE(0) を返します。

エラーコード

本関数実行後に CubGetLastError が返すエラーコードとエラー発生要因は以下の通りです。

CUB_SUCCESS	正常終了
CUB_ERR_INVALIDPARAM	ハンドル値が無効
CUB_ERR_FAILED	原因不明により処理が遂行されなかった

注釈

割込み発生要因と設定値の構成を表 3-5 に記します。

割込み発生要因に対応した設定値を clearInt0Info、clearInt1Info へセットしてください。

複数の割込み発生要因をクリアする場合は、各設定値の論理和を求めてセットしてください。

表 3-5　クリアする割込み発生要因と設定値

割込み発生要因	設定値
ジャマー検出による割込みをクリア	0x8000
PING 命令の受信による割込みをクリア	0x4000
リサイズオーバラップの発生による割込みをクリア	0x2000
ブレークパケット受信による割込みをクリア	0x1000
“ リンク NG” の判定結果による割込みをクリア	0x0800
“ リンク OK” の判定結果による割込みをクリア	0x0400
MFR の “1” であるヒット数の増減による割込みをクリア	0x0200
ランフェーズへ遷移したことによる割込みをクリア	0x0100
ネットワーク停止による割込みをクリア	0x0080
リサイズ完了による割込みをクリア	0x0040
MGR>MFR の判定結果による割込みをクリア	0x0020
MGR ≠ MFR 判定結果による割込みをクリア	0x0010
メール送信終了による割込みをクリア	0x0008
メール受信完了による割込みをクリア	0x0004
データリニューアルによる割込みをクリア	0x0002
ALM による割込みをクリア	0x0001

3.6 サンプルプログラム

3.6.1 MKY43 へのアクセスサンプル

本 API を使用して MKY43 の初期化、CUnet 通信設定、グローバルメモリの値を取得までのサンプルプログラムを記します。

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


    /** API のバージョンチェック */
    api_version = CubGetVersion();
    if (api_version < 0x100 || api_version > 0x199) {
        printf(" 互換性の無いバージョンの cub43pci1.dll です。\n");
        exit(1);
    }

    /** ハンドル生成 */
    CUBHandle = CubOpenHandle(0);
    if (CUBHandle == INVALID_HANDLE_VALUE) {
        exit(1);
    }

    /** MKY43 を初期化 */
    // (1) メモリマップ内の 0x000 ～ 0x2FF(GM + MSB) を 0x00 でライト
    for (i=0;i<0x300;i+=2) {
        CubWriteWord(CUBHandle, i, 0);
    }

    // (2) メモリマップ内の 0x400 ～ 0x5FF(MRB0 + MRB1) を 0x00 でライト
    for (i=0x400;i<0x600;i+=2) {
        CubWriteWord(CUBHandle, i, 0);
    }

    // (3) 通信設定を行う
    // (3-1) BCR へライトするために GMM 機能を ON
    CubWriteWord(CUBHandle, 0x366, 0x8000);
    // (3-2) BCR へ通信条件を設定
    //    サンプルでは、SA=0,OWN=1,BPS=6Mbps と BCR に設定
    CubWriteWord(CUBHandle, 0x356, 0x0180);
    // (3-3) GMM 機能を OFF
    CubWriteWord(CUBHandle, 0x366, 0x0000);
```

```
    /** CUnet の起動 */
    CubWriteWord(CUBHandle, 0x366, 0x0100);

    /** 本サンプルプログラムでは、CUB-43PCI1 とは別に 2 つの Unet ステーション (SA1 と SA63) と
     * リンクが成立していると仮定し、SA1 と SA63 のグローバルメモリのデータ読込みを行っています。
     */
    while(1) {
        /** CUnet のネットワーク状態を確認 */
        CubReadWord(CUBHandle, 0x366, &mky43_scr);
        if ((mky43_scr&0x0100)==0) {
            CubWriteWord(CUBHandle, 0x366, 0x0100);  // ネットワークが停止中なら再起動

        }
        // SA1 のグローバルメモリ読込み
        CubReadWord(CUBHandle, 0x0008, &sa1_gm[0]);
        CubReadWord(CUBHandle, 0x000A, &sa1_gm[1]);
        CubReadWord(CUBHandle, 0x000C, &sa1_gm[2]);
        CubReadWord(CUBHandle, 0x000E, &sa1_gm[3]);
        // SA63 のグローバルメモリ読込み
        CubReadWord(CUBHandle, 0x01f8, &sa63_gm[0]);
        CubReadWord(CUBHandle, 0x01fA, &sa63_gm[1]);
        CubReadWord(CUBHandle, 0x01fC, &sa63_gm[2]);
        CubReadWord(CUBHandle, 0x01fE, &sa63_gm[3]);
    }
    /* 生成したハンドルを閉じる */
    CubCloseHandle(CUBHandle);
    return 0;
}
```

3.6.2 割込み処理サンプル

本 API を使用して MKY43 へ割込み設定と割込み発生を確認する、サンプルプログラムを記します。

```
int main(int argc, char *argv[])
{
   HANDLE CUBHandle;
   BYTE int0_current_numOfOccurr;                  // 現在の INT0 割込み発生回数
   BYTE int0_lastTime_numOfOccurr;                 // 前回の INT0 割込み発生回数
   WORD int0_factor;                                       // INT0 割込み発生要因

   /* ハンドル生成 */
   CUBHandle = CubOpenHandle(0);
   /* 生成されたハンドルをチェック */
   if (CUBHandle == INVALID_HANDLE_VALUE) {
         exit(1);
   }

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

   /* 割込み発生要因レジスタをクリア */
   CubClearInterrupt0StatusInfo(CUBHandle, 0xffff);

   /* 割込み発生回数レジスタをクリア */
   CubClearInterrupt0Count(CUBHandle);
   int0_lastTime_numOfOccurr = 0;     // 割込み発生回数は 0 です。

   /* 割込み発生要因をセット。ネットワーク停止時に INT0 割り込みが発生します。 */
   CubWriteWord(CUBHandle, 0x358, 0x0080);

   /* ネットワーク開始指示 */
   CubWriteWord(CUBHandle, 0x366, 0x0100);

   while (1) {
      /* 割込み発生回数レジスタの情報を取得 */
      CubGetInterrupt0Count(CUBHandle, &int0_current_numOfOccurr);
      /* 前回の割込み発生回数と比較し一致しなければ割込みが発生しています */
      if (int0_lastTime_numOfOccurr != int0_current_numOfOccurr) {
         /* 現在値を前回値にコピー */
         int0_lastTime_numOfOccurr = int0_current_numOfOccurr;
         /* 割込み発生要因レジスタの情報を取得 */
         CubGetInterrupt0StatusInfo(CUBHandle, &int0_factor);
         /* 割込み発生要因が CHECK-1 か確認をする */
         if ((int0_factor & 0x0080) == 0x0080) {
            /* --- ネットワーク停止が発生したときの処理を書きます --- */
            /* INT0 割込み発生要因レジスタをクリア */
            CubClearInterrupt0StatusInfo(CUBHandle, 0x0080);
         }
      }
   }
   /* 生成したハンドルを閉じます */
   CubCloseHandle(CUBHandle);

   return 0;
}
```

■開発・製造
株式会社ステップテクニカ
〒 358-0011 埼玉県入間市下藤沢 757-3
TEL: 04-2964-8804
http://www.steptechnica.com/
info@steptechnica.com

CUnet（MKY43）搭載 PCI ボード
CUB-43PCI1
ユーザーズマニュアル

ドキュメント No.：STD_CUB43PCI1_V1.0J
発行年月日：2018 年 11 月