



STEP
TECHNICA

CUnet (MKY43) 搭載 PCI ボード
CUB-43PCI-LP
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 を搭載した PCI ボードの CUB-43PCI-LP について記述します。CUB-43PCI-LP 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、弊社の CUB-43PCI-LP 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
- ・ 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
Ver1.1J	2013年1月	--	初版発行
Ver1.3J	2014年6月	--	文章構成を変更・誤字、脱字の修正
Ver2.0J	2018年11月	--	Windows8/8.1/10に対応
		1-1	「表 1-1 ボード仕様」の構成を変更
		3-2	「3.4 制限事項」を追加
		--	API関数に MKY43 からの割込み発生回数、割込み要因情報を取得・クリア関数を追加

目次

第1章 製品概要

1.1 特徴.....	1-1
1.2 仕様.....	1-1

第2章 ハードウェア

1.1 特徴.....	1-1
1.2 仕様.....	1-1
2.1 コネクタ仕様.....	2-1
2.2 ディップスイッチ.....	2-2
2.2.1 ボードIDスイッチ (SW5).....	2-2
2.2.2 ターミネーション設定スイッチ (SW1).....	2-2
2.3 メモリマップ.....	2-3
2.3.1 MKY43.....	2-3
2.3.2 CUB-43PCI-LP 独自のレジスタ.....	2-4
2.4 寸法図.....	2-5

第3章 ソフトウェア

3.1 概要.....	3-1
3.2 著作権・免責.....	3-1
3.3 ファイル構成.....	3-2
3.4 制限事項.....	3-2
3.4.1 マルチスレッド.....	3-2
3.4.2 省電力モードについて.....	3-2
3.4.3 割り込み処理.....	3-3
3.4.4 ドライバを使用しない場合のアクセス方法.....	3-3
3.5 API仕様.....	3-4
3.5.1 CubGetVersion.....	3-5
3.5.2 CubGetLastError.....	3-6
3.5.3 CubCountDevice.....	3-6
3.5.4 CubBoardID.....	3-7
3.5.5 CubResetBoard.....	3-7
3.5.6 CubSearchBoard.....	3-8
3.5.7 CubOpenHandle.....	3-9
3.5.8 CubCloseHandle.....	3-10
3.5.9 CubReadByte.....	3-10
3.5.10 CubWriteByte.....	3-11
3.5.11 CubReadWord.....	3-11
3.5.12 CubWriteWord.....	3-12
3.5.13 CubGetInterrupt0Count、CubGetInterrupt1Count.....	3-12
3.5.14 CubClearInterrupt0Count、CubClearInterrupt1Count.....	3-13
3.5.15 CubGetInterrupt0StatusInfo、CubGetInterrupt1StatusInfo.....	3-13
3.5.16 CubClearInterrupt0StatusInfo、CubClearInterrupt1StatusInfo.....	3-15
3.6 サンプルプログラム.....	3-17
3.6.1 MKY43 へのアクセスサンプル.....	3-17
3.6.2 割り込み処理サンプル.....	3-19

図 目 次

図 2-1	パネル外観	2-1
図 2-2	コネクタ周辺回路	2-1
図 2-3	CUB-43PCI-LP ボードの設定	2-2

表 目 次

表 1-1	仕様	1-1
表 2-1	メモリマップ	2-3
表 3-1	API 関数	3-4
表 3-2	バージョン番号の構成	3-5
表 3-3	エラーコードリスト	3-6
表 3-4	int0Info、int1Info の内部構成	3-14
表 3-5	クリアする割込み発生要因と設定値	3-16

第 1 章 製品概要

本章は、CUB-43PCI-LP の製品概要について記述します。

1.1 特徴

CUB-43PCI-LP は、MKY43 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-43PCI-LP 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 に準拠した、32 ビット・33MHz 拡張バス
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 +5V
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(H) ※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-43PCI-LP 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-43PCI-LP のパネル面とその詳細を図 2-1 に示します。

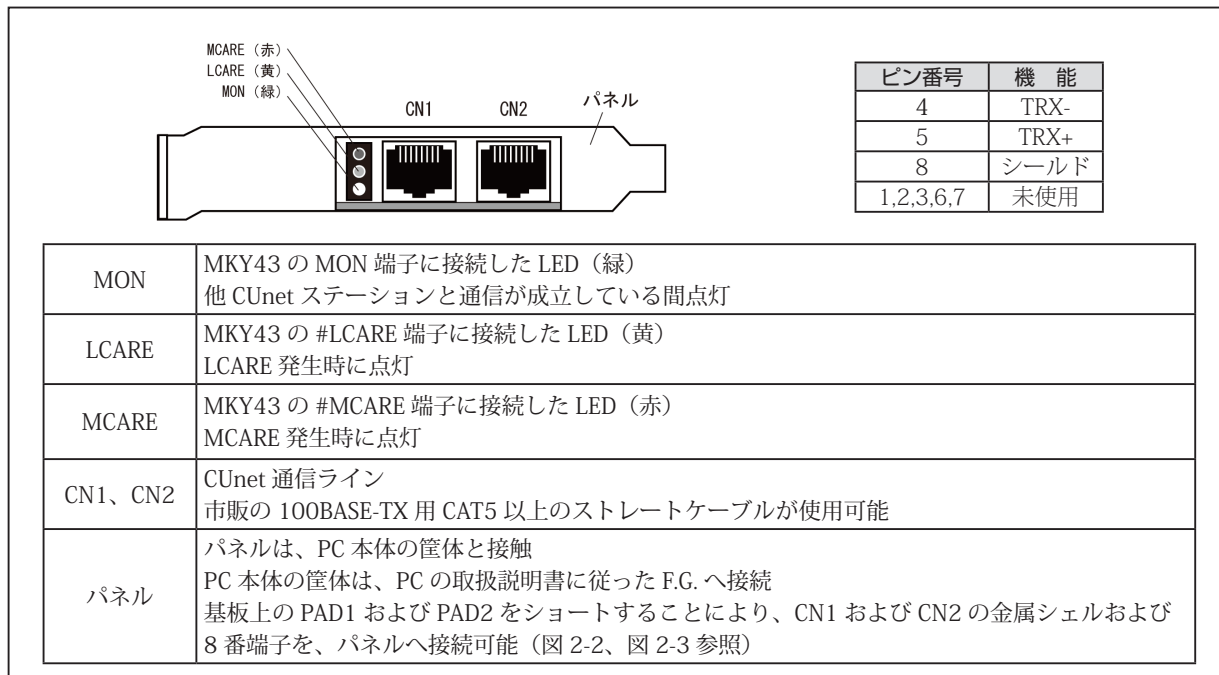


図 2-1 パネル外観

CN1、CN2 コネクタ周辺回路を図 2-2 に示します。

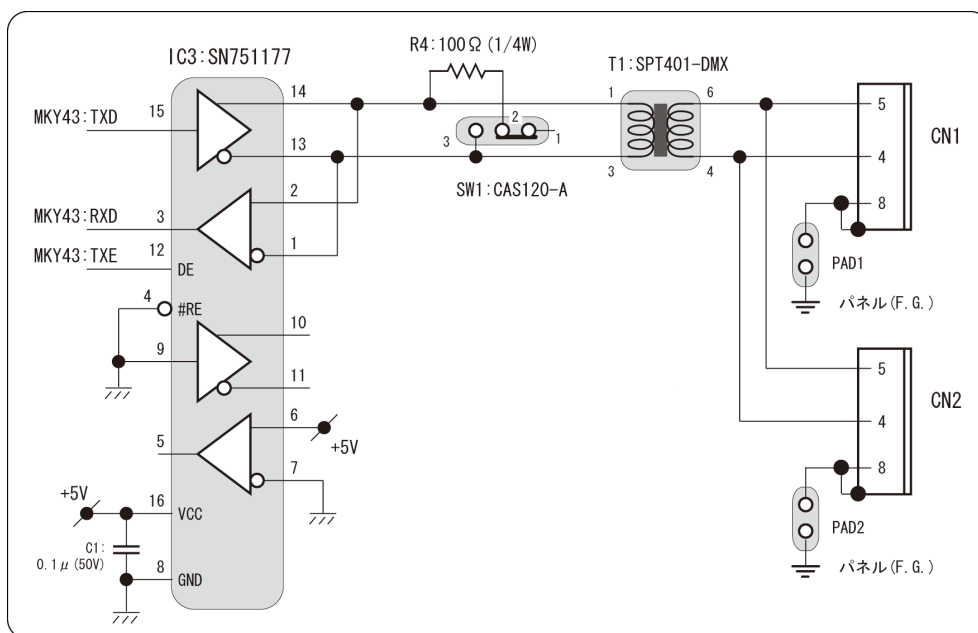


図 2-2 コネクタ周辺回路

2.2 ディップスイッチ

CUB-43PCI-LP のディップスイッチの設定を図 2-3 に示します。

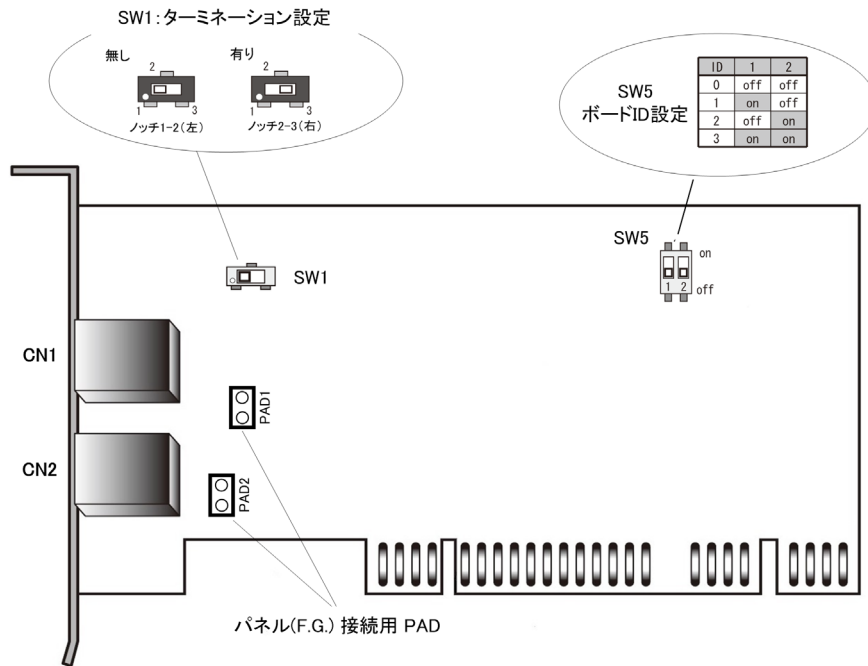


図 2-3 CUB-43PCI-LP ボードの設定

2.2.1 ボード ID スイッチ (SW5)

複数 CUB-43PCI-LP を同一の機器に搭載する場合には、SW5 のボード ID を各ボード個別の番号に設定してください。
(工場出荷時設定 ボード ID : 0)

2.2.2 ターミネーション設定スイッチ (SW1)

CUB-43PCI-LP が CUnet 通信ラインの末端になる場合は、ターミネーションを ON (有効) にしてください。
ターミネーションを ON に設定すると通信経路が 100 Ω で終端されます。
それ以外の場合は、OFF (無効) にしてください。OFF に設定すると終端されません。
(工場出荷時設定 ターミネーション : 無効)

2.3 メモリマップ

CUB-43PCI-LP 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-43PCI-LP の先頭アドレスからの相対値であり、実際のアドレスは PCI BIOS から自動的に割り振られた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 ~ BFFH	未使用
C00H	Chip Reset Register
C02H ~ DFFH	未使用
E00H	Board ID Register
E02H ~ 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 のソフトウェア」、「4.1.1 メモリマップ」をご参照ください。

2.3.2 CUB-43PCI-LP 独自のレジスタ

表 2-1 のメモリマップに示された C00H および E00H のレジスタは、CUB-43PCI-LP 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 : 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
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、280ns 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 : E00H

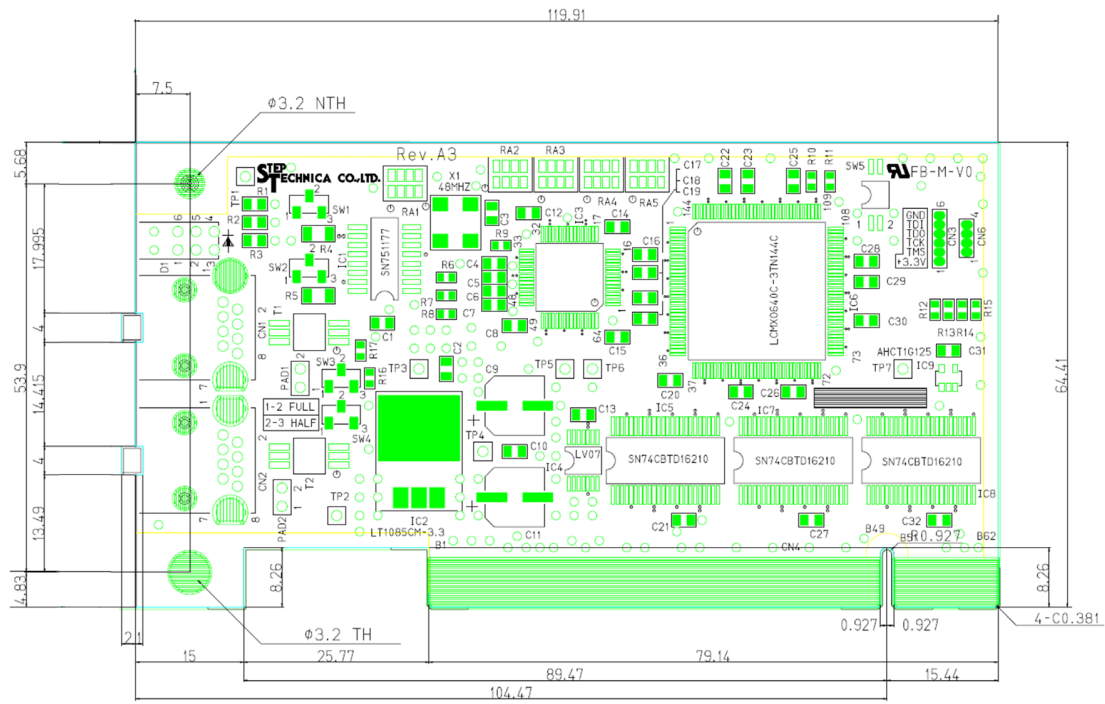
bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R/W	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
機能	--	--	--	--	--	--	--	--	--	--	--	--	--	--	BID1	BID0

[機能説明] BID0,1 (Board ID) ビットをリードすることにより、SW5 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 ~ BFFH、C02H ~ DFFH、E02H ~ 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



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 バージョンのメジャー番号「2」以上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

Windows 上のユーザアプリケーションからの CUB-43PCI-LP への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 をダウンロードできます。

URL : <http://www.steptechnica.com/jp/download/index.html>

対応 OS は

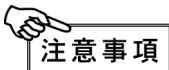
- Windows10 (64bit/32bit)
- Windows8.1 (64bit/32bit)
- Windows 8 (64bit/32bit)
- Windows 7 (64bit/32bit)

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可能です。

3.2 著作権・免責

全てのドキュメント・プログラム・プログラムソースの著作権は、株式会社ステップテクニカが所有しています。
株式会社ステップテクニカは、以下の注意事項を了承された個人・法人、その他の団体が CUB-43PCI-LP を利用する場合に限り、これら著作物の複製・利用をする権利をライセンスするものです。株式会社ステップテクニカに断り無く、CUB-43PCI-LP 以外でこれら著作物の一部または全部を改訂・再配布や複製・利用することはできません。



注意事項

- 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、一切保証致しません。
- ④ OS や開発環境等に関するお問い合わせはサポート致しかねます。
- 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。

【cub43pcilp.dll】

DLL 本体です。Windows のシステムフォルダか、本 DLL を使用するユーザプログラムと同じフォルダにコピーしてお使いください。

【cub43pcilp.lib】

Microsoft Visual C/C++ 用のインポートライブラリです。

【cub43pcilp.h】

API のヘッダファイルです。ご使用の際は、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

DLL 内の API 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-43PCI-LP は、省電力モード機能に対応していません。

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、グローバルメモリ、メール送信バッファ、メール受信バッファ 0、1 領域は不定の状態になりますのでご注意ください。

3.4.3 割り込み処理

MKY43 では、INTOSR と INT1SR レジスタにより割り込み発生状況の確認が行えます。

ドライバ内部には、割り込み発生時の INTOSR、INT1SR の情報を保持するレジスタ（割り込み発生要因レジスタ）と INTO、INT1 それぞれで割り込みが発生した回数を保持するレジスタ（割り込み発生回数レジスタ）があります。ドライバ内部では、割り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 での割り込み発生した場合の説明を記します。）

- ① 割り込み発生要因レジスタに割り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り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り込み発生要因が残った状態でセットされます。）
- ② 割り込み発生回数レジスタの値をインクリメントする。
- ③ INTOSR の 0~15bit の内で“1”となっている箇所に“1”をライトし、MKY43 の割り込みを解除します。

割り込み発生要因レジスタと割り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) 割り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)
ドライバ内にて MKY43 からの INTO、INT1 それぞれの割り込み発生回数を割り込み発生回数レジスタに保持しています。
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) 割り込み発生回数レジスタのクリア (CubClearInterrupt0Count, CubClearInterrupt1Count)
割り込み発生回数レジスタをクリアします。
- (3) 割り込み発生要因レジスタの情報を取得 (CubGetInterrupt0StatusInfo, CubGetInterrupt1StatusInfo)
ドライバ内部にて MKY43 からの INTO、INT1 それぞれの割り込み発生要因を割り込み発生要因レジスタに保持しています。
本 API 関数では、割り込み発生要因レジスタから情報を取得します。
- (4) 割り込み発生要因レジスタのクリア関数 (CubClearInterrupt0StatusInfo, CubClearInterrupt1StatusInfo)
割り込み発生要因レジスタの指定した割り込み発生原因をクリアします。

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 からの割り込み発生回数と割り込み発生要因の確認を行ってください。

3.4.4 ドライバを使用しない場合のアクセス方法

ステップテクニカ社製ドライバを使用せずに CUB-43PCI-LP 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-43PCI-LP 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-43PCI-LP 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-43PCI-LP 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

表 3-1 にサポートしている API 関数の一覧を示します。

以降に説明する API 関数は、cub43pcilp.h に収録されている関数です。

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	CUB-43PCI-LP ボードの枚数を取得
CubBoardID	ボード ID を取得
CubResetBoard	MKY43 のリセットを指示
CubSearchBoard	CUB-43PCI-LP ボードの枚数とそのボード ID を取得
CubOpenHandle	CUB-43PCI-LP のハンドル値を取得
CubCloseHandle	CubOpenHandle で取得した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 INTO、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 INTO、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 INTO、1 割込み発生要因を取得
CubClearInterrupt0StatusInfo CubClearInterrupt1StatusInfo	ドライバ内部で保持している INTO、1 割り込み要因の内、指定した割り込み要因をクリア

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

戻り値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

注釈表 3-2 に、バージョン番号の構成を示します。
バージョン番号が変更される原因は、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

戻り値

cub43pcilp.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 cub43pcilp.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	ハンドルが無効
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実行できなかった
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-43PCI-LP ボードの枚数を返します。

パラメータ

なし

戻り値

CUB-43PCI-LP ボードの枚数を返します。

-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-43PCI-LP ボードのボード ID を取得します。

パラメータ

HANDLE CUBHandle CUB-43PCI-LP のハンドル

戻り値

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	ハンドルが無効
CUB_ERR_FAILED	原因不明により処理が実行できなかった

3.5.5 CubResetBoard

書式

BOOL CubResetBoard (HANDLE CUBHandle);

機能

指定された CUB-43PCI-LP ボードの MKY43 をリセットします。

パラメータ

HANDLE CUBHandle CUB-43PCI-LP のハンドル

戻り値

正常終了時は TRUE、失敗時は FALSE 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	ハンドルが無効
CUB_ERR_FAILED	原因不明により処理が実行できなかった

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-43PCI-LP ボード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。 <ul style="list-style-type: none">• -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 要素持つ配列のアドレスを指定します。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。 <ul style="list-style-type: none">• 0x00 ~ 0x03 : ボード ID• 0xFF : 認識できなかった

戻り値

正常終了時は TRUE、失敗時は FALSE 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実行できなかった

注釈

ボード ID は、SW5 により設定します。
CUB-43PCI-LP ボード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-43PCI-LP ボードを識別できます。
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-43PCI-LP ボードが装着されており、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態で 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-43PCI-LP のハンドルを取得します。

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-43PCI-LP ボードが1枚の場合は、0をセットしてください。 詳しくは、“注釈”を参照してください。
--------------	---

戻り値

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実行できなかった

注釈

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。

CUB-43PCI-LP ボードが1枚の場合は、CubSearchBoard を実行する必要がありません。
CUB-43PCI-LP ボード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-43PCI-LP 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-43PCI-LP ボードが装着されており、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。

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-43PCI-LP のハンドル

戻り値

正常終了時は TRUE、失敗時は FALSE 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	ハンドルが無効
CUB_ERR_FAILED	原因不明により処理が実行できなかった

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-43PCI-LP のハンドル
const ULONG Adr	アドレス値 入力条件は以下の通り ・入力範囲：0x0000 ~ 0x0FFF
BYTE *Dat	読み込みデータ格納先のアドレス

戻り値

正常終了時は TRUE、失敗時は FALSE 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実行できなかった

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-43PCI-LP のハンドル
const ULONG Adr	アドレス値 入力条件は以下の通り ・入力範囲：0x0000 ~ 0x0FFF
const WORD Dat	書き込みデータ

戻り値

正常終了時は TRUE、失敗時は FALSE 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	ハンドルが無効 Adr が範囲外
CUB_ERR_FAILED	原因不明により処理が実行できなかった

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-43PCI-LP のハンドル
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

戻り値

正常終了時は TRUE、失敗時は FALSE 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実行できなかった

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-43PCI-LP のハンドル
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

戻り値

正常終了時は TRUE、失敗時は FALSE 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実行できなかった

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 INTO、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ます。

パラメータ

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

戻り値

正常終了時は TRUE、失敗時は FALSE 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実行できなかった

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 INTO、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-43PCI-LP のハンドル

戻り値

正常終了時は TRUE、失敗時は FALSE 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	ハンドルが無効
CUB_ERR_FAILED	原因不明により処理が実行できなかった

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 INTO、1 割込み発生要因情報を取得します。

パラメータ

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

戻り値

正常終了時は TRUE、失敗時は FALSE 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	ハンドルが無効
CUB_ERR_FAILED	*int0Info、*int1Info に NULL が指定された 原因不明により処理が実行できなかった

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 INTOSR、INT1SR と同等です。

表 3-4 int0Info、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	リサイズオーバーラップの発生による割込み発生
12	ブ레이크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

ドライバ内部で保持している INTO、1 割り込み要因の内、指定した割り込み要因をクリアします。

パラメータ

HANDLE CUBHandle CUB-43PCI-LP のハンドル
WORD clearInt0Info、clearInt1Info クリアする割り込み発生要因を指定

戻り値

正常終了時は TRUE、失敗時は FALSE 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	ハンドルが無効
CUB_ERR_FAILED	原因不明により処理が実行できなかった

注釈

割り込み発生要因と設定値の構成を表 3-5 に記します。

割り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。

複数の割り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
リサイズオーバー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する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 < 0x200 || api_version > 0x299) {
        printf(" 互換性の無いバージョンの cub43pcilp.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 ヘライトするために 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-43PCI-LP とは別に 2 つの CUnet 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み込み
    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み込み
    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 INTO 割込み発生回数
    BYTE int0_lastTime_numOfOccurr;        // 前回の INTO 割込み発生回数
    WORD int0_factor;                       // INTO 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 INTO 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 --- */
                /* INTO 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.steptecnica.com/>

info@steptecnica.com

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

ドキュメント No. : STD_CUB43PCILP_V2.0J

発行年月日 : 2018 年 11 月