DIO シリーズ・32IN/32OUT
USB, Ethernet, WiFi インターフェース

HUSB-DIO464U
HETN-DIO864T
HWIF-DIO864W

ユーザーズマニュアル

〈ソフトウェア・USB 編
Windows 版〉

NC ボードシリーズ

絶縁型入出力ボード

株式会社ハイバーテック
http://www.hivertec.co.jp/

この説明書は次のデバイス(ボード)に適応しています.

USB	HUSB-DIO464U HUSB- DIO464U(D)
USB + Ethernet	HETN- DIO864T HETN- DIO864T(D)
USB + WiFi	HWIF- DIO864W HWIF- DIO864W (D)

株式会社 ハイパーテック
東京都江東区新大橋 1-8-11
大樹生命新大橋ビル
TEL 03-3846-3801
FAX 03-3846-3773
sales@hivertec.co.jp

第 1.00 版 2019 年 04 月 24 日発行

本製品をご使用される前に「注意事項」を必ずご一読の上ご利用をお願い致します.

目　　次

図　表　目　次

■ 注意事項

■ 保証範囲

1. 本製品の保証期間は，お買い上げ頂いた日より 3 年間です．保証期間中に弊社の判断により欠陥が判明した場合には，本製品を弊社に引き取り，修理または交換を行います．

2. 保証期間内外に関わらず，弊社製品の使用，供給(納期)または故障に起因する，お客様及び第三者が被った，直接，間接，二次的な損害あるいは，遺失利益の損害に付いて，弊社は本製品の販売価格以上の責任を負わないものとしますので，予めご了承ください．

■ 免責事項

1. 本書に記載された内容に沿わない，製品の取付，接続，設定，運用により生じた損害に対しましては，一切の責任を負いかねますので，予めご了承ください．

2. 本製品は，一般電子機器用(工作機械・計測機器・FA/OA 機器・通信機器等)に製造された半導体製品を使用していますので，その誤作動や故障が直接，生命を脅かしたり，身体・財産等に危害を及ぼしたりする恐れのある装置(医療機器・交通機器・燃焼機器・安全装置等)に適用できるような設計，意図，または，承認，保証もされていません．
   ゆえに本製品の安全性，品質および性能に関しては，本書(またはカタログ)に記載してあること以外は明示的にも黙示的にも一切保証するものではありませんので，予めご了承ください．

3. 保証期間内外に関わらず，お客様が行った弊社の承認しない製品の改造または，修理が原因で生じた損害に対しましては，一切の責任を負いかねますので，予めご了承ください．

4. 本書に記載された内容について，弊社もしくは，第三者の特許権，著作権，商標権，その他の知的所有権の権利に対する保証または実施権の許諾を行うものではありません．
   また本書に記載された情報を使用したことにより第三者の知的所有権等の権利に関わる問題が生じた場合，弊社は，その責任を負いかねますので，予めご了承ください．

■ 安全にお使い頂くために

この度は，弊社 NC ボードシリーズ製品をご採用頂きまして，誠に有り難う御座います．本書は，本製品をご使用して頂く場合の取扱い，留意点に付いて記入してありますので，必ずご一読の上ご利用をお願い致します．
尚，本書は，本書が添付された製品常設箇所付近の分かりやすい場所に常時保管し，必要に応じて適宜参照・確認頂きますよう，お願い致します．

安全上の注意

本製品のご使用前に，必ずこのユーザーズマニュアル及び付属書類を全て熟読し，内容を理解してから正しくご使用下さい．本製品の知識，安全の情報及び注意事項の全てに付いて習熟してからご使用下さい．
本ユーザーズマニュアルでは，安全注意事項のランクを「警告」，「注意」として区分してあります．

⚠ **警 告**	この表示を無視して，誤った取扱いをすると，人が死亡または重傷を負う可能性が想定される内容を示しています．

⚠ **注 意**	この表示を無視して，誤った取扱いをすると，人が傷害を負う可能性または物的損害が想定される内容を示しています．

■ 対象ユーザー

⚠ 注 意

本製品およびマニュアルは，以下の様な，ユーザーを対象としています．

・拡張用デバイスの増設および配線に付いて基本的な知識を有している方．
・制御用電子機器およびパソコン等に付いて基本的な知識を有している方．

■ 適合 Bus

⚠ 警 告

本製品は，USB インターフェース部は **Universal Serial Bus 2.0** に適合しています．
また Ethernet インターフェース部は IEEE802.3i/u(10Base-T/100Base-TX),
WiFi は IEEE802.11b/g/n に準拠しています

■ 環境条件

⚠ 警 告

本製品は，下記の環境条件下で保管・ご使用下さい．

- 動作周囲温度　0℃ ～ +50℃
- 動作周囲湿度　20%RH ～ 85%RH(但し結露せぬこと)
- 保存周囲温度　-15℃ ～ +75℃
- 保存周囲湿度　10%RH ～ 90%RH(但し結露せぬこと)
- 雰 囲 気　　腐食性ガス・引火性ガス・オイルミスト・塵埃のないこと
- 標 高　　海抜 3000m 以下(300m 毎に 2℃の上限値を下げた範囲で使用して下さい)

■ 運搬・取り付け

⚠ 警 告

本製品にふれる前に，金属に触り身体の静電気を取り除いて下さい．
静電気は，本製品の故障の原因になります．

本製品を静電気の帯びやすい梱包材(エアーキャップなど)でくるまないで下さい．
静電気は，本製品の故障の原因になります．

本製品の上に重いものを載せないで下さい．重いものを乗せますと，部品が損傷し故障の原因になります．

本製品のジャンパ及びディップスイッチの設定は，パソコン等に接続する前に行って下さい．
電源が ON の状態で設定しますと，設定を正しく認識しないで誤動作の原因になります．

本製品のジャンパ及びディップスイッチの設定は，正しく行って下さい．
設定を間違えますと誤動作の原因になります．

USB コネクタは本来 ホットプラグインが許されていますが，本製品への接続には電源 ON 状態での USB コネクタの抜き挿しは避けてください．

本製品をパソコン等と接続する時は，コネクタを深くしっかりと挿入し，ケーブルの直近部分を固定する等して，動作中に抜ける，または接触不良等が発生しない様な措置を施して下さい．
動作中に抜ける，または接触不良等が発生すると，誤動作，故障の原因となります．

⚠ 注 意

本製品を落とすなど乱暴に扱わないで下さい．衝撃や振動が故障の原因となります．

本製品の半田面を手で直接触らないで下さい．部品の突起などにより怪我をする恐れがあります．

■ **添付ソフトウェア適合 OS**

⚠ 注 意

⚠ 本製品の標準添付ソフトウェアは Windows10, Windows 8.1, Windows 7 SP1 に対応しております．マイクロソフト社 OS サポートのライフサイクル期間が終了した OS の対応については，マイクロソフト社 OS サポートのライフサイクル期間に確認したものであり，本マニュアル発行時点での動作を保証するものではありません．

■ **サンプルプログラム開発環境**

⚠ 注 意

⚠ 本製品のサンプルプログラムは Microsoft Visual Studio のプロジェクト及びソースコードを添付しています．マイクロソフト社製品サポートのライフサイクル期間が終了した Microsoft Visual Studio の各バージョンについては，マイクロソフト社製品サポートのライフサイクル期間に確認したものであり，本マニュアル発行時点での動作を保証するものではありません．

■ **ユーザプログラム**

⚠ 警 告

⚠ 本製品に添付されるサンプルプログラムまたはマニュアル内のコード例は，本製品の機能・動作をソフトウェア面から理解して頂く為のものです．故に使用される機器毎に固有な安全対策処理・エラー処理・例外処理・排他処理等は省略されています．実際にプログラムを作成する場合は，十分に安全対策等を考慮し，必要な処理を追加してください．

■ 配線

警 告
 外線用コネクタへの配線作業や外線用コネクタの着脱は，パソコン等の電源を OFF にし，電源コードを抜いてから行って下さい. 電源コードを抜かないで作業を行った場合，故障の原因になります．また，装置が思わぬ動作をすることがあります.
 外線用コネクタへの配線は，コネクタ信号表などをよく確認し，正しく配線して下さい. 間違った配線をしますと，故障・焼損の原因になります.
 外部から供給する電源は，必ず定格以内でご使用下さい．定格以外で使用されますと，故障・焼損・誤動作の原因となります.
 入出力回路に接続する回路は，必ず定格電流・電圧以内でご使用下さい．定格以外で使用されますと，故障・焼損・誤動作の原因となります.
 外部配線用コネクタは，推奨のコネクタをご使用下さい．推奨以外のコネクタを使用されますと，接触不良などにより誤動作の原因となります.
 外部配線用コネクタは，必ずロックしてご使用下さい．ロックしないで使用されますと，コネクタが外れる，または接触不良等により誤動作の原因となります.
 外部配線用ケーブルは，引っ張る，または重い荷重を掛ける等しないで下さい. コネクタが外れる，または接触不良等により誤動作の原因となります.
 外部配線用ケーブルは，モータの配線やAC電源ケーブルなど，ノイズの多い配線とは出来るだけ離して下さい．配線が近いとノイズが誤動作の原因となります.

■ 廃棄

警 告
 本製品を廃棄する時は，関連する法律・規則に従って処理して下さい.

■ DIO シリーズのマニュアル構成

DIO シリーズ製品のマニュアルは

(1) ユーザーズマニュアル　<ハードウェア編>
(2) ユーザーズマニュアル　<ソフトウェア・USB 編 Windows 版> ※本書
(3) ユーザーズマニュアル　<ソフトウェア・Ethernet/WiFi 編 Windows 版>

の 3 部構成です.

各マニュアルの内容は以下の通りです.

ユーザーズマニュアル　＜ハードウェア編＞
ー主として配線担当者向け

- ブロック図
- デバイス各部名称
- 入出力ポート構成
- デバイス上の設定
- 入出力回路
- 外部との接続
- DIO464v2 との相違点について

ユーザーズマニュアル　＜ソフトウェア・USB 編 Windows 版＞
ー主としてソフトウェア開発者向け（USB インターフェース使用時）

- ソフトウェアの種類と対応 OS
- 添付ソフトウェアの構成
- インストールとアンインストール
- デバイスへのアクセスとボードID
- デバイスドライバI／F関数
- サンプルプログラム
- USB コマンド資料

ユーザーズマニュアル　＜ソフトウェア・Ethernet/WiFi 編 Windows 版＞
ー主としてソフトウェア開発者向け
（Ethernet または WiFi インターフェース使用時）

- ソフトウェアの種類と対応 OS
- 添付ソフトウェアの構成
- インストールとアンインストール
- デバイスへのアクセスとボードID
- デバイスドライバI／F関数
- サンプルプログラム
- Ethernet/WiFi コマンド資料

1. はじめに

本製品は DIO シリーズ・USB/ Ethernet/ WiFi インターフェース・DI32 点／DO32 点の入出力として，次の 3 製品×2 タイプで構成される製品グループです．

HUSB-DIO464U ・・・ USB (標準タイプ)
HUSB-DIO464U(D) ・・・ USB＋DIN 取付台
HETN- DIO864T ・・・ USB+Ethernet (標準タイプ)
HETN- DIO864T (D) ・・・ USB+Ethernet＋DIN 取付台
HWIF- DIO864W ・・・ USB+WiFi (標準タイプ)
HWIF- DIO864W (D) ・・・ USB+ WiFi ＋DIN 取付台

このマニュアルは上記製品のソフトウェアマニュアル（USB インターフェース使用／Windows 版）です．
ハードウェアの結線やコネクタピンアサイン等は別冊の「DIO ボードシリーズ ユーザーズマニュアル<ハードウェア編>」を併せてお読みください．また，Ethernet/WiFi インターフェース使用時のソフトウェアに」関しては別冊「DIO ボードシリーズ ユーザーズマニュアル<ソフトウェア・Ethernet/WiFi 編 Windows 版>」を併せてお読みください．
本書では以降，本製品を総称して DIO あるいは DIO464 と呼称する場合があります．

2. ソフトウェア概要

本製品の使用には，デバイスドライバおよびDLLのインストールが必要です．さらに製品用アプリケーションの開発では所定の手順にてプログラムを作成する必要があります．このため製品に必要なデバイスドライバ，DLL，サンプルプログラム等のソフトウェアが添付CDに収納されています．本書ではこれらについての関連する情報や使い方などについて説明します．

3. ソフトウェアの種類と対応 OS

本製品では以下2種類のドライバおよび1つの DLL が必要です.
各ソフトウェアは Windows7 以降の 32bit 版および 64bit 版 Windows に対応しています.
なお本製品のドライバおよび DLL をインストールする際，従来製品 HUSB-DIO464V2 ドライバが既にインストールされている場合（CD バージョン 6.x.x.x 以前でインストールしたドライバ）は先にアンインストールしてから行う必要があります.

(1) VCP ドライバ

本製品は USB インターフェースに Sillicon Labs 社の USB-Serial Bridge コントローラを使用しており，HUSB-DIO464U/ HETN-DIO864T/ HWIF-DIO864W で USB インターフェースを使用する場合はこの仮想 COM ドライバが必要です.
このドライバはPCに搭載される OS によって使用するインストーラが異なります.
◇Winsow10 – 64bit 版用
・・CD フォルダ Usb¥vcp¥CP210x_Universal_Windows_Driver 内 CP210xVCPInstaller_x64.exe
◇Winsow10 – 32bit 版用
・・CD フォルダ Usb¥vcp¥CP210x_Universal_Windows_Driver 内 CP210xVCPInstaller_x86.exe
◇Winsow8.1/8/7 - 64bit 版用
・・CD フォルダ Usb¥vcp¥CP210x_Windows_Drivers 内 CP210xVCPInstaller_x64.exe
◇Winsow8.1/8/7 - 32it 版用
・・CD フォルダ Usb¥vcp¥CP210x_Windows_Drivers 内 CP210xVCPInstaller_x86.exe
インストーラにより silabser.sys がインストールされます.

(2) デバイスドライバ I/F 用 DLL

本デバイスで使用する DLL は，従来製品である HUSB-DIO464v2 用 DLL とファイル名，関数名，関数仕様で互換性を持ちます. そのため従来製品を含む DIO464 全てで使用することが可能です.
ただし， HUSB-DIO464v2 に添付された CD からインストールした同名 DLL は，本ボートインストーラでインストールした DLL と互換性がありません. そのためそのまま使用した場合，本デバイスは正常に動作しませんので，既にインストールされた HUSB-DIO464v2 用ドライバが存在する場合は，本デバイスインストール前にアンインストールが必要です.
インストールは 32bit 版 Windows または 64bit 版 Windows 毎に用意されたセットアッププログラムを使って行います.
◇Windows 7 以降 - 64bit 版用・・CD フォルダ Usb¥dll¥x64 内 setup.exe
◇Windows 7 以降 - 32bit 版用・・CD フォルダ Usb¥dll¥x86 内 setup.exe
インストーラにより hudio464.dll がインストールされます.

(3) デバイスドライバ（HUSB-DIO464v2 用）

このドライバは従来製品の HUSB-DIO464v2 を本デバイスと併用する場合に必要です. HUSB-DIO464v2 を使用しない場合は本デバイスドライバのインストールは不要です.
このデバイスドライバをインストールする際は，既にインストール済みの HUSB-DIO464v2 用デバイスドライバを先にアンインストールする必要があります.
インストールはドライバパッケージインストーラを使って行います.
◇Windows 7 以降 - 64bit 版用・・CD フォルダ Usb¥dpi¥x64 内 dpinst.exe
◇Windows 7 以降 - 32bit 版用・・CD フォルダ Usb¥dpi¥x86 内 dpinst.exe
インストーラにより hd464wdm.sys がインストールされます.

4. 添付ソフトウェアの構成

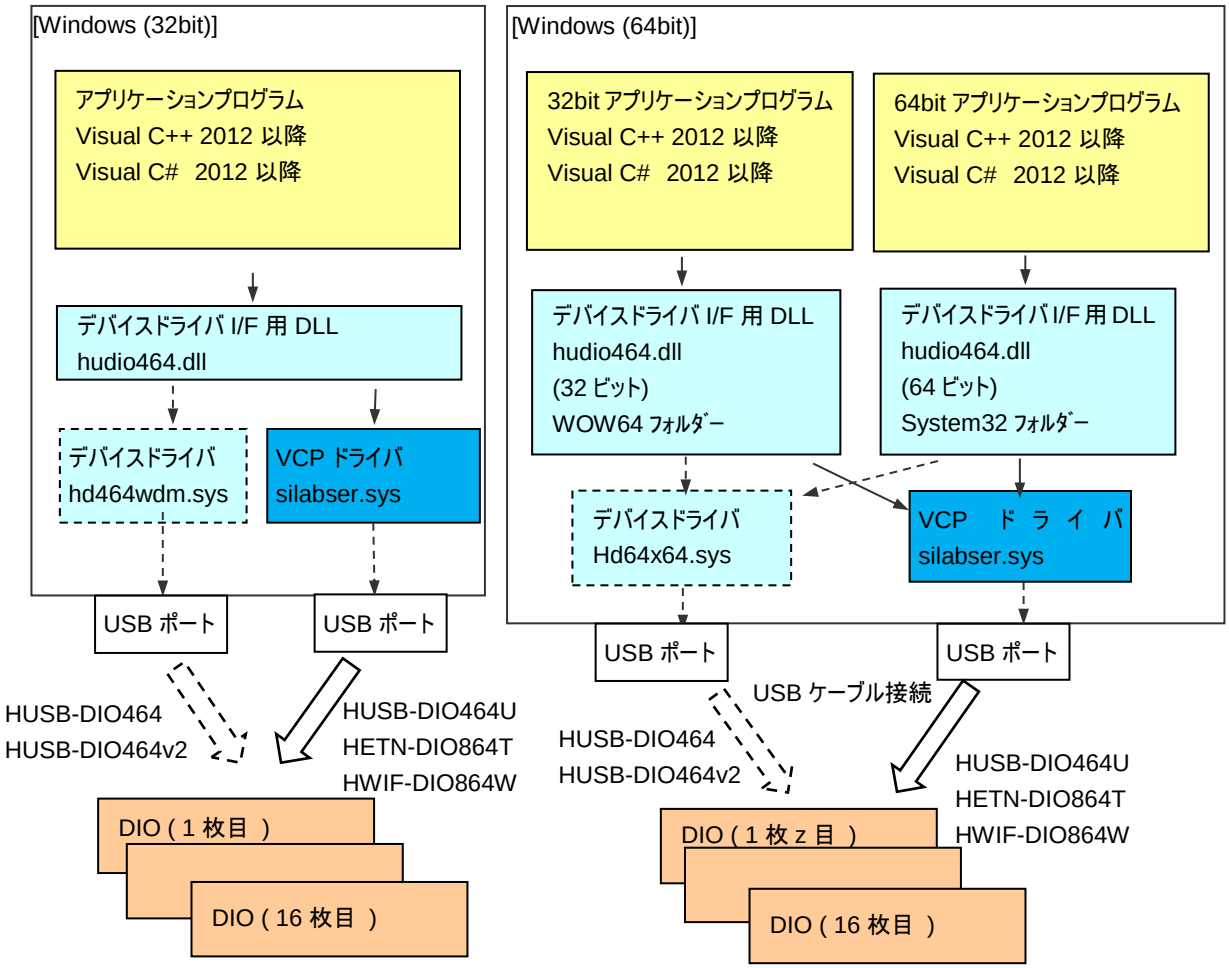

図 4.1-1 ソフトウェア構成

5. インストールとアンインストール

DIO464 を使用するには，VCP(仮想 COM ドライバおよびドライバ I/F 用 DLL のインストールが必要です．
また従来製品 HUSB-DIO464v2 を併用する場合は，デバイスドライバのインストールが必要になります．
インストールは本製品に添付されている CD（バージョン 7.0.0.0 以降）を使用しなければなりません．従来製品のドライバが既にインストールされている場合は，本製品各ドライバ，DLL のインストール前に必ずアンインストールしてください．
なお VCP ドライバおよびデバイスドライバのインストールが終了するまでデバイスは USB ポートに接続しない下さい．

インストール時の注意

本製品のドライバおよび DLL のインストール時，従来製品 HUSB-DIO464V2 のドライバが既にインストールされている場合は必ず先にそれらドライバのアンインストールを行ってから，本製品のドライバ，DLL のインストールを行って下さい．上書インストールした場合，デバイスが正常に動作しない場合があります．

5.1 VCP ドライバのインストール

DIO464 は仮想 COM ポートを経由して USB 通信を行います．そのため本製品を使用するには VCP ドライバが必要です．VCP ドライバは Sillicon Labs 社製のものを使用します．

(1) Windows10 へのインストール

① パソコンの電源を ON にして Windows を起動します．
② CD ドライブ “¥Usb¥vcp¥ CP210x_Universal_Windows_Driver”を開きます．
③ OS が 32bit 版の場合は“CP210xVCPInstaller_x86.exe“を実行します．
OS が 64bit 版の場合は“CP210xVCPInstaller_x64.exe“を実行します．
④ 画面の指示に従ってインストールを進めます
⑤ 正常に終了したら VCP ドライバのインストールは終了です．

(2) Windows8.1/8/7 へのインストール

① パソコンの電源を ON にして Windows を起動します．
② CD ドライブ “¥Usb¥vcp¥ CP210x_ Windows_Driver”を開きます．
③ OS が 32bit 版の場合は“CP210xVCPInstaller_x86.exe”を実行します．
OS が 64bit 版の場合は“CP210xVCPInstaller_x64.exe”を実行します．
④ 画面の指示に従ってインストールを進めます
⑤ 正常に終了したら VCP ドライバのインストールは終了です．

※ネットワーク上の CD イメージからインストーラを実行した時，以下エラーダイアログが表示される場合があります．
この場合はインストーラをローカルディスク上で実行してください

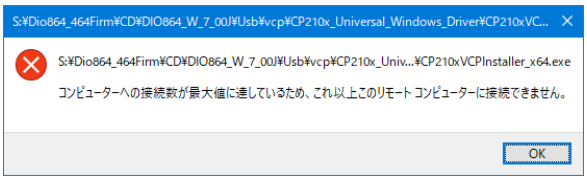

図 5.1-1 ネットワーク経由インストール時エラー

5.2 デバイスドライバI／F用 DLL のインストール

ドライバ I/F 用 DLL は，VCP ドライバが生成した仮想 COM ポートを経由して USB 通信を行ない，DIO464 にアクセスするための関数群です．
このドライバ I/F 用 DLL は従来製品の HUSB-DIO464v2 に対しても互換性があります．

① 従来製品に添付されている CD（CD バージョン 6.x.x.x.以前）によって古いドライバが既にインストールされている場合，これらをアンインストールします．
② OS が 32bit 版の場合，CD ドライブ“¥Usb¥dll¥x86”を開きます．
OS が 64bit 版の場合，CD ドライブ“¥Usb¥dll¥x64”を開きます．
③ setup.exe”を実行します．
④ 画面の指示に従ってインストールを進めます．
⑤ 正常に終了したらドライバ I/F 用 DLL のインストールは終了です．

5.3 デバイスドライバのインストール

従来製品 HUSB-DIO464v2 を DIO464 と共に使用する場合，従来製品用のデバイスドライバも別途インストールする必要があります.

(1) Windows7 以降 32bit 版 へのインストール

① CD ドライブ “¥Usb¥dpi¥x86¥”開きます.
② ”dpinst.exe”を実行します.
③ 画面の指示に従ってインストールを進めます
④ 「ドライバーソフトウェアの発行元を検証できません」とのメッセージが出る場合がありますが，
「このドライバーソフトウェアをインストールします」をクリックします.

(2) Windows7 以降 64bit 版 へのインストール

① CD ドライブ “¥Usb¥dpi¥x64¥”開きます.
② ”dpinst.exe”を実行します.
③ 画面の指示に従ってインストールを進めます

5.4 ドライバ・DLL のアンインストール

DIO464 に添付された CD からインストールしたドライバおよび DLL のアンインストールは，Windows 設定メニュの「コントロールパネル」ー「プログラムのアンインストール」または「アプリと機能」から行います．

Windows7 以降のアンインストール

① デバイスを取り外し，デバイスの電源を OFF します．
② スタートメニューからコントロールパネルを起動します．
③ “プログラムの追加と削除”あるいは“アプリと機能”を選択します．
④ プログラムのアンインストールから以下プログラムをアンインストールします．

プログラムの種類	プログラム登録名（名称の先頭部分のみ抜粋） ※
VCP ドライバ	"Windows ドライバ パッケージ- Silicon Laboratories Inc. (silabser) Ports…”
デバイスドライバ I/F 用 DLL	"Hivertec Library Package for HUSB-DIO464…”
デバイスドライバ	"Windows ドライバ パッケージ - Hivertec HUSB-DIO464 …”
デバイスドライバ	"Windows ドライバ パッケージ - Hivertec HUSB-DIO464v2 …”

※ドライバおよび DLL インストール時，Windows ロケール設定が日本語以外だった場合，プログラム登録名は設定したロケールに従った言語表示になるのでご注意下さい

表 5.4-1 “アプリと機能”上でのアプリケーション登録名

図 5.4-1 “アプリと機能”画面

アンインストール時の注意

デバイスドライバのアンインストール時は必ずデバイスを取り外し，DIO464 の電源を OFF にして行ってください．
電源を OFF にせずアンインストールを行うと，OS の再インストールが必要になる場合があります．

6. デバイスへのアクセスとボード ID

6.1 デバイスを複数枚使用する場合

DIO464 を同一コンピュータに複数枚接続し，それぞれのデバイスと外部の接続を1対1に対応させたい場合について説明します．

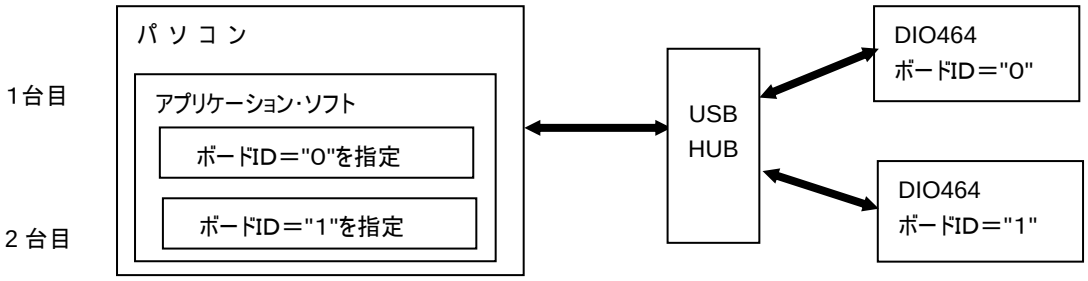

図 6.1-1 PC と複数枚デバイスの接続例

(1) HUSB-DIO464 の指定
USB ケーブルで接続された2台以上の DIO464 は，パソコンによって内部で固定の番号が割り振られていますが，この番号をソフトウェアで利用できません．そのため，同一型名のデバイスを識別するために，各 DIO464 にそれぞれ固有な「ボード ID No.」を設定し識別します．
(ボード ID は 0~15)

(2) ボード ID の確認方法
添付 CD 内のVCサンプル「¥Usb¥sample¥Vc¥Release¥sud46400.exe」を実行し，画面のボード ID を確認します．

(3) ボード ID の設定方法
ボード ID の設定については ハードウェア編を参照してください．
※ ボード ID の設定は DIO464，DIO464v2 で共通です．
※ 1 枚でも複数枚でも DIO464 は複数プロセスから制御できません．

6.2 デバイスの着脱と認識について

DIO464 の接続，切り離しを行った時のデバイスの認識および動作は次のようになります．
アプリケーション起動後の着脱やデバイス電源を入れたままの着脱は避けてください．

	項　目	動　作
1	DIO464 の認識	アプリケーションプログラム起動時に行われます
2	起動時以降に接続されたデバイス	認識されません
3	起動時以降に切り離されたデバイス	制御不能となります．PC 自体の動作にも影響が出る場合があります．
4	アプリケーション実行中に任意の USB 接続	アプリケーションの USB 通信に影響を与える場合があります （通信停止が発生するなど）
5	DIO464（デバイス）の着脱	PC 電源を投入したままのデバイス接続あるいはデバイス電源を投入した場合は 5 秒以上経過してから使用（通信）します．

表 6.2-1 USB ポートとデバイスの着脱と認識

6.3 デバイスアクセス方法

PC に接続された任意の DIO464 にアクセスするためには，どのようなハードウェアリソースを持つデバイスをオープンするかという情報が必要です．このハードウェアリソースをデバイス情報と呼びます．このデバイス情報を取得し，デバイス情報を使用してデバイスをオープンし，アクセスするためのデバイスハンドル値を取得します．

6.3.1 デバイス（ボード）認識用のデータ構造体

デバイス認識のために次に示す HUSBDEVINF 型構造体が用意されています．
従来製品とデータ互換性、共用性を持たせるため，構造体およびメンバー名が同じになっていますが，
メンバーのデータ内容は異なります．

[Visual C++]

```
typedef struct _HUSBDEVINF {      // デバイス情報
        WORD        BrdID;          // ボード ID( 0～15 )
        WORD        EPcnt;          // 1 固定
        WORD        EPspc[8];       // 配列[0]のみ使用．ﾃﾞﾊﾞｲｽﾘｽﾄｲﾝﾃﾞｯｸｽ番号(0～)
        WORD        EPsiz[8];       // 配列[0]のみ使用．仮想 COM ポート番号(1～65535)
        WORD        EPopt[8];       // 0 固定
        HANDLE      EPhdl[8];       // 配列[0]のみ使用．ﾃﾞﾊﾞｲｽﾊﾝﾄﾞﾙ
} HUSBDEVINF, *PHUSBDEVINF;
```

[Visual C#]

```
[StructLayout(LayoutKind.Sequential)]
public struct HUSBDEVINF
{       // デバイス情報
        Public      ushort  BrdID;          // ボード ID( 0～15 )
        public      ushort  EPcnt;          // 1 固定
        public      ushort  EPspc0;         // EPspc0 のみ使用．ﾃﾞﾊﾞｲｽﾘｽﾄｲﾝﾃﾞｯｸｽ番号(0～)              /
        public      ushort  EPspc1;
        public      ushort  EPspc2;
        public      ushort  EPspc3;
        public      ushort  EPspc4;
        public      ushort  EPspc5;
        public      ushort  EPspc6;
        public      ushort  EPspc7;
        public      ushort  EPsiz0;         // EPsiz0 のみ使用．仮想 COM ポート番号(1～65535)
        public      ushort  EPsiz1;
        public      ushort  EPsiz2;
        public      ushort  EPsiz3;
        public      ushort  EPsiz4;
        public      ushort  EPsiz5;
        public      ushort  EPsiz6;
        public      ushort  EPsiz7;
        public      ushort  EPopt0;         // 0 固定
        public      ushort  EPopt1;
        public      ushort  EPopt2;
        public      ushort  EPopt3;
        public      ushort  EPopt4;
        public      ushort  EPopt5;
        public      ushort  EPopt6;
        public      ushort  EPopt7;
        public      IntPtr  EPhdl0;         // EPhdl0 のみ使用．ﾃﾞﾊﾞｲｽﾊﾝﾄﾞﾙ
        public      IntPtr  EPhdl1;
        public      IntPtr  EPhdl2;
        public      IntPtr  EPhdl3;
        public      IntPtr  EPhdl4;
        public      IntPtr  EPhdl5;
        public      IntPtr  EPhdl6;
        public      IntPtr  EPhdl7;
}
```

6.3.2　ドライバ関数の使用

(1)　Visual C++ によるアプリケーションの構築

次のファイルをプロジェクトへ追加します.

■プロジェクト追加ファイル

◇ドライバ　関数用・・hudio464.lib　・・　ドライバ関数インポートライブラリ

■インクルードファイル

◇ドライバ　関数用・・**hudio464.h**　・・　ドライバ関数結合用ヘッダーファイル

(2)　Visual C# によるアプリケーションの構築

次のファイルをプロジェクトへ追加します.

◇ドライバ　関数用・・**hudio464.cs**　・・　ドライバI／F用DLL関数定義標準モジュールファイル

このファイルに外部関数宣言（DllImport），及びユーザー定義型宣言が記述されています.

6.3.3　C++アプリケーションでの使用

ドライバ関数は**「C 言語」**で作成されています. これらの関数をC++アプリケーションで使用できるように,
ドライバ関数のヘッダーファイル内で以下のように「C言語」として明示的な定義をしています.

```
//-------------------------------------------------------
//        関数プロトタイプ宣言
//-------------------------------------------------------
#ifdef __cplusplus
extern "C"
{
#endif
        個々の関数のプロトタイプ宣言
#ifdef __cplusplus
}
#endif
```

(1)　#ifdef __cplusplus ・・Visual C++ 用の定義です
　　#endif　　C++コーディングでは明示的にライブラリ関数が”Cモジュール”として解釈されます.

(2)　extern "C"　・・ C モジュールで定義されている関数を表します.

7. デバイスドライバI／F関数

7.1 関数一覧

No	関数名称	機　能	備考
1	dio464_GetDeviceCount	デバイス枚数の取得	
2	dio464_GetDeviceInfo	デバイス情報の取得	
3	dio464_OpenDevice	デバイスオープン	
4	dio464_CloseDevice	デバイスクローズ	
5	dio464_rInB	入力ポートからの 1 バイトデータ読出	
6	dio464_rOutB	出力ポートからの 1 バイトデータ読出	
7	dio464_wOutB	出力ポートへの　1 バイトデータ書込	
8	dio464_rInW	入力ポートからの 2 バイトデータ読出	
9	dio464_rOutW	出力ポートからの 2 バイトデータ読出	
10	dio464_wOutW	出力ポートへの　2 バイトデータ書込	
11	dio464_rInDW	入力ポートからの 4 バイトデータ読出	
12	dio464_rOutDW	出力ポートからの 4 バイトデータ読出	
13	dio464_wOutDW	出力ポートへの　4バイトデータ書込	
14	dio464_InOutDW	出力ポート 4 バイト書込／入力ポート 4 バイト読出	
15	dio464_SetFilter	入力ポートのフィルタの設定 （HUSB-DIO464v2 互換）	
16	dio464_SetFilterEx	入力ポートのフィルタの設定	DIO464V2 非互換
17	dio464_GetBoardCode	デバイス固有コードの取得	DIO464V2 非互換
18	dio464_GetBoardType	デバイスタイプの取得	DIO464V2 非互換
19	dio464_rVersion	デバイスバージョンの取得	DIO464V2 非互換
20	dio464_UsbSend	USB 送信	DIO464V2 非互換
21	dio464_UsbRecv	USB 受信	DIO464V2 非互換

表 7.1-1　関数一覧

7.2 デバイスとの通信時間

ドライバ関数の概略所要時間を次の表に示します．下記計測値は個々のパソコンおよび使用する USB ポートの種類によって若干の差がありますのでご参考用としてご使用ください．

No	関数名称	通信時間（マイクロ秒 ）	参考値 HUSB-DIO464v2 通信時間（ マイクロ秒 ）
1	dio464_GetDeviceCount	----	----
2	dio464_GetDeviceInfo	----	----
3	dio464_OpenDevice	----	----
4	dio464_CloseDevice	----	----
5	dio464_rInB	3000	500
6	dio464_rOutB	3000	500
7	dio464_wOutB	200	250
8	dio464_rInW	3000	500
9	dio464_rOutW	3000	500
10	dio464_wOutW	200	250
11	dio464_rInDW	3000	500
12	dio464_rOutDW	3000	500
13	dio464_wOutDW	200	250
14	dio464_InOutDW	4000	500
15	dio464_SetFilter	500	250
16	dio464_SetFilterEx	500	
17	dio464_GetBoardCode	----	
18	dio464_GetBoardType	----	
19	dio464_rVersion	3000	
20	dio464_UsbSend	200	
21	dio464_UsbSend +dio464_UsbRecv	4000	

表 7.2-1 関数通信時間

注意 1 ：項目 17,18 は DLL 内部で管理するデータを使用して値を返している関数のため通信時間は 0 になります．
注意 2 ：項目 20，21 はデータ長によって時間が変わります．（項目 20 は 10Byte のデータを送信した時の時間）
注意 3 ：項目 21 は単独での計測ができないため項目 20 をセットにした送受信の通信時間を計測
（送信 10 Byte，受信 6Byte を送受信した時の通信時間 → 項目 14 の機能を本関数で実行）

7.3 デバイスアクセスの準備手順

(1) 使用する全デバイスのデバイス情報の取得

"HUSBDEVINF"型構造体エリア（の配列）内に，全 DIO464 のデバイス情報をまず取得します．

◆ dio464_GetDeviceCount()・・・デバイス枚数の確認

◆ dio464_GetDeviceInfo() ・・・全デバイスのデバイス情報を取得

(2) ボード毎にデバイスオープン

ある1つの DIO464 のデバイス情報をデバイスオープン関数に渡します．

この結果その DIO464 がオープンされ，デバイスオープン関数はこのボードにアクセスするためのデバイスハンドル値を返してきます．

ボード枚数が2枚以上の場合には，個々のボード毎にこの処理を行います．

特定のボードを選択する場合には，ボードID値をチェックして下さい．

◆ dio464_OpenDevice() ・・・デバイスのオープン処理

以降は，この「デバイスハンドル」を使用し，そのボードにアクセスすることができるようになります．

(3) 全ての処理が終了してアプリケーションを終了する場合には，オープンしたデバイスの「クローズ処理」を行って下さい．

◆ dio464_CloseDevice() ・・・デバイスのクローズ処理（1 枚分）

***** デバイス判別の特例 *****

デバイス情報取得時に，デバイスが認識されない，デバイス設定のボードIDが見つからない場合があります．

■次の場合には DIO デバイスとしては認めていません．（デバイス情報なし）

◆ベンダ ID・プロダクト ID が一致しても，デバイスからの情報取得に失敗．

◆デバイスのエンドポイント数が不足． （入出力不可）

◆各エンドポイントの仕様が一致しない．（入出力不可）

■上記の条件を満足しても，次の場合にはボード ID＝99 として，ドライバは処理しています．

◆DIO464 の持つ一部の機能が認識できない．

7.4 アプリケーション作成上の注意

本製品には使用方法の参考となるようにサンプルプログラムが添付されていますが，ここには表現されていない注意点があります．アプリケーションを作成する場合はこれらの点に留意して戴き，必要に応じて適切な処理を行ってください．

(1) 排他について

Windows ではマルチスレッドがサポートされていますが，マルチスレッドを使用し複数のスレッドからデバイスにアクセスする場合は同期(排他処理)が必要です．

DIO では各デバイスに送信用エンドポイントと受信用エンドポイントを一つずつ持ち，データ送受信を行っています．

データ受信時にはデータ受信要求送信後，データ受信を行いますが，この順序が異なると正しく通信できません．

例えば二つのスレッドで制御している場合，一つのスレッドがデータ受信要求送信を行い，データ受信を完了する前に，他のスレッドがデータ受信要求送信を行うと，データ受信できなくなり，タイムアウトエラーが発生します．

同様に一つのスレッドがデータ受信要求送信を行い，データ受信を完了する前に，他のスレッドがデータ送信を行った場合もデータ受信できなくなり，タイムアウトエラーが発生します．

この様な状況を避けるため，排他処理が必要なります．

具体的にはマルチスレッドを使用し複数のスレッドからデバイスにアクセスする場合はミューテックスやクリティカルセクションなどを使用し，使用するAPI関数の排他処理が必要となります．

<イメージ>

```
Thread1()
{
   Lock();
   dio464_rInDW( hDev1, &dt );;          //デバイス 1 の DI ﾎﾟｰﾄ読出
   UnLock();
}

Thread2()
{
   Lock();
   dio464_rInDW( hDev2, &dt );;          //デバイス 2 の DI ﾎﾟｰﾄ読出
   UnLock();
}
```

7.5 関数の戻り値

ドライバの関数を使用する時，関数の戻り値が異常値（'0'以外）であった場合には，異常内容に対応した処理を行います．
通常，この異常が発生した場合にはアプリケーションプログラムの続行は困難であり，プログラム内容の再検討が必要に
なります．

No	戻り値			異 常 内 容 と 確 認 項 目
	記号表記	16 進数表記		
		VC++, VC#	VB, VB.NET	
1	NO_ERROR	0x000	&H0	正　常
				異常は発生していません
2	NOT_FOUND	0x001	&H1	デバイスドライバが存在しない
				◎デバイスドライバがインストールされていない ◎デバイスドライバが所定のフォルダに格納されていない
3	ALREADY_OPENED	0x002	&H2	既にオープン済のデバイスをオープン
				◎オープン済みデバイスに更にオープン指令 ◇オープンしたデバイスはクローズするまで使用 （多重のオープンは禁止） ◎ボード2枚以上使用する場合,オープンするデバイス情報の更新を確認します.
4	NOT_MEMORY	0x004	&H4	デバイス情報格納メモリが不足
				◎アプリケーション用のメモリ不足 ◇パソコン主記憶メモリの不足 ◎システムリソース（OS用メモリ）の不足 ◇多数のアプリケーション起動 ◇1度に多数のウインドウを開いた
5	INVALID_HANDLE	0x008	&H8	無効なデバイスハンドルを指定
				◎デバイスオープンで得られた"デバイスハンドル"の不使用 ◎このデバイスは既にクローズされている
6	NOT_READY	0x010	&H10	デバイスの入出力ポートが使用できない
				◎デバイス（ボード）内部の入出力ポートがない システムとの不整合等が考えられますので，弊社サポートまでお問い合わせください
7	ILLEGAL_DEVICE	0x020	&H20	デバイス固有情報が不正
				◎デバイス情報とデバイス機能が一致しない ◇アプリケーション起動後の USB ケーブル抜き差しや差し替え.
8	USB_TIMEOUT	0x080	&H80	USB 受信でﾀｲﾑｱｳﾄ発生
				◎USB からデータが受信できない ◇ケーブルが外れた ◇アプリケーションで不正な受信を行おうとした
9	ILLEGAL_PARAM	0x100	&H100	関数の引数の値が異常
				◇引数の設定値を確認（マニュアル照合）
10	DEVICE_STALL	0x80000000	&H80000000	通信異常
				◎USB 通信中に異常が発生 デバイスまたは PC 側の不具合が考えられます．ﾃﾞﾊﾞｲｽおよび PC の再起動を試行してください

表 7.5-1　関数の戻り値

7.6 ドライバ I/F 用 DLL 関数詳細

7.6.1 dio464_GetDeviceCount() デバイス枚数の取得

機　能	現在パソコンに装着されている DIO464 の枚数を取得します.

開発環境	書　式
VC++	DWORD WINAPI dio464_GetDeviceCount(DWORD*　count);
VC#	[DllImport("*libname* ")] public static extern uint dio464_GetDeviceCount(ref uint count);

引　数	説　明
count	取得した DIO464 枚数

VC++ 記述例	DWORD*　count;　　//DIO464 の枚数 DWORD　　ret;　　//関数の戻り値 ret = dio464_GetDeviceCount(&count);

備　考	

7.6.2 dio464_ GetDeviceInfo() デバイス情報の取得

機　能	現在パソコンに装着されている指定枚数 DIO464 のデバイス情報を取得します. この結果，デバイス情報構造体の配列にデバイス情報が格納されます. このデバイス情報は，デバイスオープン時に利用します.

開発環境	書　式
VC++	DWORD WINAPI dio464_GetDeviceInfo(DWORD* count, HUSBDEVINF* HusbInf);
VC#	[DllImport("*libname* ")] public static extern uint dio464_GetDeviceInfo(ref uint count, ref HUSBDEVINF HusbInf);

引　数	説　明
count	取得した DIO464 枚数
HusbInf	取得するデバイス情報

VC++ 記述例	DWORD　　ret;　　//関数の戻り値 DWORD　　count = 2;　　//使用枚数は 2 HUSBDEVINF　HusbInf[2];　　//2 枚の DIO464 のデバイス情報がセットされるべきエリア ret = dio464_GetDeviceInfo(&count, &HusbInf[0]);

備　考	

7.6.3 dio464_ OpenDevice() デバイスのオープン

機　能	渡したデバイス情報を持つ DIO464 をオープンし，他と識別するためのデバイスハンドルを取得します． 以降このデバイスハンドルは，この DIO464 にアクセスするためのハンドルとなります． また，この時出力ポートはすべて”0”になります．

開発環境	書　式
VC++	DWORD WINAPI dio464_OpenDevice(DWORD * hDev, HUSBDEVINF* HusbInf);
VC#	[DllImport("*libname* ")] public static extern uint dio464_OpenDevice(ref uint hDev, ref HUSBDEVINF HusbInf);

引　数	説　明
hDev	取得するデバイスハンドル
HusbInf	オープンするデバイスのデバイス情報

VC++ 記述例	DWORD ret;　　//関数の戻り値 DWORD hDev;　　//デバイスハンドルの格納先 HUSBDEVINF hInf;　　//オープンする DIO のデバイス情報格納先 ret = dio464_OpenDevice(&hDev, &hInf);

備　考	

7.6.4 dio464_ CloseDevice() デバイスのクローズ

機　能	渡したデバイスハンドルを持つ DIO464 をクローズします． 以降このデバイスハンドルは，無効となり，この DIO464 にアクセスはできません． また，出力ポートの状態は最終出力状態が保持されます．

開発環境	書　式
VC++	DWORD WINAPI dio464_CloseDevice(DWORD hDev);
VC#	[DllImport("*libname* ")] public static extern uint dio464_CloseDevice(uint hDev);

引　数	説　明
hDev	クローズするデバイスのデバイスハンドル

VC++ 記述例	DWORD　ret;　　//関数の戻り値 ret = dio464_CloseDevice(hDev);

備　考	

7.6.5　dio464_rInB()　入力ポートからの 1 バイトデータ読出

7.6.6　dio464_rOutB()　出力ポートからの 1 バイトデータ読出

7.6.7　dio464_wOutB()　出力ポートへの 1 バイトデータ書込

機　能	デバイスハンドルで指定された DIO464 の指定された入力ポートまたは出力ポートから 入力ポートからの1バイトデータ読出・・1 バイトデータを読み出し指定したエリアに格納します. 出力ポートからの1バイトデータ読出・・1 バイトデータを読み出し指定したエリアに格納します. 出力ポートへの1バイトデータ書込　・・1 バイトデータを書き込みます. 出力ポートの読み出しにより最終出力状態の確認ができます.

開発環境	書　式
VC++	DWORD WINAPI dio464_rInB (DWORD hDev, WORD port, WORD*　wrdt); DWORD WINAPI dio464_rOutB(DWORD hDev, WORD port, WORD*　wrdt); DWORD WINAPI dio464_wOutB(DWORD hDev, WORD port, WORD　wwdt);
VC#	[DllImport("*libname* ")] public static extern uint dio464_rInB(uint hDev, ushort port, ref ushort wrdt); [DllImport("hudio464.dll")] public static extern uint dio464_rOutB (uint hDev, ushort port, ref ushort wrdt); [DllImport("hudio464.dll")] public static extern uint dio464_wOutB(uint hDev, ushort port, ushort wwdt);

引　数	説　明
hDev	対象デバイスのデバイスハンドル
port	ポート指定(0:ポート 1, 1:ポート 2, 2:ポート 3, 3:ポート 4)
wrdt	読み出しデータを格納する2バイトエリアのアドレス（上位バイトは 0）
wwdt	2バイト書き込みデータ（上位バイトの内容は無効）

VC++ 記述例	DWORD　ret;　　//関数の戻り値 WORD　　dt;　　//データ格納エリア ret = dio464_rInB(hDev, 0, &dt);　//入力ポート 1 の読み出し

備　考	1バイト入出力関数で返すワードデータ上位の内容は次のようになります 読出データは　　上位バイト:0x00 　　下位バイト:読出データ 書込データは　　上位バイト:無視 　　下位バイト:書込データ

7.6.8　dio464_rInW()　入力ポートからの 2 バイトデータ読出

7.6.9　dio464_rOutW()　出力ポートからの 2 バイトデータ読出

7.6.10 dio464_wOutW()　出力ポートへの 2 バイトデータ書込

機　能	デバイスハンドルで指定された DIO464 の指定された入力ポートまたは出力ポートから 入力ポートからの2バイトデータ読出・・2 バイトデータを読み出し指定したエリアに格納します. 出力ポートからの2バイトデータ読出・・2 バイトデータを読み出し指定したエリアに格納します. 出力ポートへの2バイトデータ書込　・・2 バイトデータを書き込みます. 出力ポートの読み出しにより最終出力状態の確認ができます.

開発環境	書　式
VC++	DWORD WINAPI dio464_rInW (DWORD hDev, WORD port, WORD*　wrdt); DWORD WINAPI dio464_rOutW(DWORD hDev, WORD port, WORD*　wrdt); DWORD WINAPI dio464_wOutW(DWORD hDev, WORD port, WORD　 wwdt);
VC#	[DllImport("*libname* ")] public static extern uint dio464_rInW (uint hDev, ushort port, ref ushort wwdt); [DllImport("*libname* ")] public static extern uint dio464_rOutW (uint hDev, ushort port, ref ushort wwdt); [DllImport("*libname* ")] public static extern uint dio464_wOutW (uint hDev, ushort port, ushort wwdt);

引　数	説　明
hDev	対象デバイスのデバイスハンドル
port	ポート指定(0:ポート 1, 1:ポート 2, 2:ポート 3, 3:ポート 4)
wrdt	読み出しデータを格納する2バイトエリアのアドレス
wwdt	2ﾊﾞｲﾄ書き込みデータ

VC++ 記述例	DWORD　ret;　　//関数の戻り値 WORD　 dt;　　//データ格納エリア ret = dio464_rInW(hDev, 0, &dt);　//入力ポート 1,2 の読み出し

備　考	入出力関数でﾎﾟｰﾄ4を指定した場合，上位ﾊﾞｲﾄは入力の場合は 0 を返し，出力の場合は無視されます.

7.6.11 dio464_rInDW() 入力ポートからの 4 バイトデータ読出

7.6.12 dio464_rOutDW() 出力ポートからの 4 バイトデータ読出

7.6.13 dio464_wOutDW() 出力ポートへの 4 バイトデータ書込

機　能	デバイスハンドルで指定された DIO464 の入力ポートまたは出力ポートから 入力ポートからの 4 バイトデータ読出・・4 バイトデータを読み出し指定したエリアに格納します. 出力ポートからの 4 バイトデータ読出・・4 バイトデータを読み出し指定したエリアに格納します. 出力ポートへの 4 バイトデータ書込　・・4 バイトデータを書き込みます. 出力ポートの読み出しにより最終出力状態の確認ができます.

開発環境	書　式
VC++	DWORD WINAPI dio464_rInDW (DWORD hDev, DWORD*　drdt); DWORD WINAPI dio464_rOutDW(DWORD hDev, DWORD*　drdt); DWORD WINAPI dio464_wOutDW(DWORD hDev, DWORD　dwdt);
VC#	[DllImport("*libname*")] public static extern uint dio464_rInDW(uint hDev, ref uint drdt); [DllImport("*libname*")] public static extern uint dio464_rOutDW(uint hDev, ref uint drdt); [DllImport("*libname*")] public static extern uint dio464_wOutDW(uint hDev, uint dwdt);

引　数	説　明
hDev	対象デバイスのデバイスハンドル
drdt	読み出しデータを格納する4バイトエリアのアドレス
dwdt	4ﾊﾞｲﾄ書き込みデータ

VC++ 記述例	DWORD　ret;　　//関数の戻り値 DWORD　dt;　　//データ格納エリア ret = dio464_rInDW(hDev, &dt);　//入力ポート読み出し

備　考	1バイトの入出力関数の場合 読出データは　　上位バイト:0x00 　　下位バイト:読出データ 書込データは　　上位バイト:無視 　　下位バイト:書込データ

7.6.14 dio464_InOutDW() DIO 出力ポート 4 バイト書込／入力ポート 4 バイト読出

機　能	デバイスハンドルで指定された DIO464 の出力ポートへ 4 バイトデータを書込ます. また出力直前の入力ポートから4バイトデータを読み出し，指定したエリアに格納します.

開発環境	書　式
VC++	DWORD WINAPI dio464_InOutDW (DWORD hDev, DWORD* drdt, DWORD dwdt);
VC#	[DllImport("*libname*")] public static extern uint dio464_InOutDW(uint hDev, ref uint drdt, uint dwdt);

引　数	説　明
hDev	対象デバイスのデバイスハンドル
drdt	読み出しデータを格納する4バイトエリアのアドレス
dwdt	4ﾊﾞｲﾄ書き込みデータ

VC++ 記述例	DWORD　ret;　　//関数の戻り値 DWORD　rdt;　　//データ格納エリア ret = dio464_InOutDW(hDev, &rdt, 0x55555555); //55555555h のデータ出力と入力ポート読出

備　考	

7.6.15 dio464_SetFilter() 入力ポートのフィルタの設定（HUSB-DIO464v2 互換）

機　能	デバイスハンドルで指定された DIO464 の指定された入力ポートのフィルタを設定します この関数は DIO464V2 とソフトウェア互換を保つために用意されています. 設定可能なフィルタ時間は従来製品と異なりますのでご注意ください

開発環境	書　式
VC++	DWORD WINAPI dio464_SetFilter (DWORD hDev, WORD port, DWORD wtm);
VC#	[DllImport("*libname*")] public static extern uint dio464_SetFilter(uint hDev, ushort port, uint wtm);

引　数	説　明
hDev	対象デバイスのデバイスハンドル
port	ポート指定 (0x1：ポート 1, 0x2：ポート 2, 0x4：ポート 3, 0x8：ポート 4)
wtm	フィルタ時間 (0 ～640 ※実際に設定される値は備考を参照)

VC++
記述例
	DWORD ret; //関数の戻り値
	ret = dio464_ SetFilter(hDev, 0x1, 100); //ポート 1 に 10msec のフィルタ設定

備　考

ポート指定が複数の場合は OR したデータを与えます.

従来製品 HUSB-DIO464V2 の wtm 設定範囲は 0 から 1 単位の設定ですが、本製品で使用した場合，指定したフィルタ時間に対する実際に設定される値は以下の設定になります.

Wtm の設定値	実際に設定される値	フィルタ時間（サンプリング間隔）
0	0	フィルタなし
1	1	0.1ms
2	2	0.2ms
3~4	4	0.4ms
5~8	8	0.8ms
9~10	10	1ms
11~40	40	4ms
41~80	80	8ms
81~160	160	16ms
161~320	320	32ms
321 以上	640	64ms

7.6.16 dio464_SetFilterEx() 入力ポートのフィルタの設定

機　能	デバイスハンドルで指定された DIO464 の指定された入力ビットのフィルタを設定します（4 ビット単位） この関数は HUSB-DIO464U/HETN-DIO864T/HWIF-DIO864W 専用の入力ポートフィルタの 設定関数です.

開発環境	書　式
VC++	DWORD WINAPI dio464_SetFilterEx (DWORD hDev, WORD port, WORD wtmno);
VC#	[DllImport("*libname*")] public static extern uint dio464_SetFilterEx (uint hDev, ushort port, ushort wtmno);

引　数	説　明
hDev	対象デバイスのデバイスハンドル
port	設定ビット指定 (0x01：Bit4~1, 0x02：Bit8~5, 0x04：Bit12~9, 0x08：Bit16~13, 0x10：Bit20~17, 0x20：Bit24~21, 0x40：Bit28~25, 0x80：Bit32~29)
wtmno	フィルタ選択 (0:フィルタなし, 1：5us, 2：10us, 3：20vs, 4：40μs, 5：80μs, 6：100μs, 7：200μs, 8：400μs, 9：800μs, 10：1ms, 11：4ms, 12：8ms, 13：16ms, 14：32ms, 15：64ms)

VC++
記述例
	DWORD ret; //関数の戻り値
	ret = dio464_ SetFilterEx(hDev, 0x3, 10); //ポート 1(Bit7-0)に 1msec のフィルタ設定

備　考	フィルタ設定をするビットの指定は 4 ビット単位のブロックで行うことができ, 該当する複数のブロックがある場合は OR したデータを与えます.

7.6.17 dio464_GetBoardCode() デバイス固有コードの取得

機　能	デバイスハンドルで指定された DIO464 のデバイス固有コードを取得します

開発環境	書　式
VC++	DWORD WINAPI dio464_GetBoardCode (DWORD hDev, WORD* bcode);
VC#	[DllImport("hudio464.dll")] public static extern uint dio464_GetBoardCode (uint hDev, ref ushort bcode);

引　数	説　明
hDev	対象デバイスのデバイスハンドル
bcode	デバイス固有コードの格納先

VC++
記述例
	DWORD ret; //関数の戻り値
	WORD bcode;
	ret = dio464_GetBoardCode (hDev, &bcode); //

備　考	HUSB-DIO464U, HETN-DIO864T, HWIF-DIO864W の何れかで 0x464b を返します. その他デバイスの場合はエラーを返します.

7.6.18 dio464_GetBoardType()　デバイスタイプの取得

機　能	デバイスハンドルで指定された DIO464 のボードタイプを取得します

開発環境	書　式
VC++	DWORD WINAPI dio464_ dio464_GetBoardType (DWORD hDev,WORD* btype);
VC#	[DllImport("hudio464.dll")] public static extern uint dio464_GetBoardType (uint hDev, ref ushort btype);

引　数	説　明
hDev	対象デバイスのデバイスハンドル
btype	デバイスタイプの格納先

VC++ 記述例	DWORD　ret;　　//関数の戻り値 WORD btype ret = d dio464_GetBoardType (hDev, &btype);　//

備　考	HUSB-DIO464U の場合は 3，HETN-DIO864T の場合は 4，HWIF-DIO864W の場合は 5 を返します． 何れのデバイスでもない場合はエラーを返します．

7.6.19 dio464_rVersion ()　デバイスバージョンの取得

機　能	デバイスハンドルで指定された DIO464 のハードウェアおよびソフトウェアバージョンを取得します

開発環境	書　式
VC++	DWORD WINAPI dio464_rVersion (DWORD hDev, WORD* ver);
VC#	[DllImport("hudio464.dll")] public static extern uint dio464_rVersion (uint hDev, ref ushort ver);

引　数	説　明
hDev	対象デバイスのデバイスハンドル
Ver[2]	バージョンの格納先（[0]:ハードウェアバージョン，[1]:ソフトウェアバージョン）

VC++ 記述例	DWORD　ret;　　//関数の戻り値 WORD ver[2]; ret = dio464_rVersion (hDev, ver);　//

備　考	バージョン 1.2.3.4 の場合は 0x1234 を返します．

7.6.20 dio464_UsbSend()　USB 送信

機　能	デバイスハンドルで指定された DIO464 のへ USB 経由でデータを送信します

開発環境	書　式
VC++	DWORD WINAPI dio464_ UsbSend (DWORD hDev, BYTE* data, WORD* leng);
VC#	[DllImport("*libname*")] public static extern uint dio464_ UsbSend (uint hDev, ref byte data, ref ushort leng);

引　数	説　明
hDev	対象デバイスのデバイスハンドル
data	送信データの格納先
leng	送信するデータ数

VC++ 記述例	DWORD　ret;　　//関数の戻り値 BYTE sdata[6] = {0x90, 0x00, 0x10, 0x32, 0x54, 0x76}; WORD sleng = 6; ret = dio464_UsbSend (hDev, sdata, &sleng);　//DO 書込コマンドでデータ 0x76543210 を送信

備　考	この関数は「9.USB コマンド情報」に記載されているコマンドを使用する場合に限り、dio464_UsbRecv()と共に使用されるものです. その他用途では使用できませんのでご注意ください. .

7.6.21 dio464_UsbRecv()　USB 受信

機　能	デバイスハンドルで指定された DIO464 から USB 経由でデータを受信します

開発環境	書　式
VC++	DWORD WINAPI dio464_ UsbRecv (DWORD hDev, BYTE* data, WORD* leng);
VC#	[DllImport("*libname*")] public static extern uint dio464_ UsbRecv (uint hDev, ref byte data, ref ushort leng);

引　数	説　明
hDev	対象デバイスのデバイスハンドル
data	受信データの格納先
leng	受信したデータ数

VC++ 記述例	DWORD　ret;　　//関数の戻り値 BYTE sdata[2] = {0x93, 0x00}; BYTE rdata[4]; WORD sleng = 2; WORD rleng; = 4; ret = dio464_UsbSend (hDev, sdata, &sleng);　//DI 読出コマンド送信 if(ret != 0) return; ret = dio464_ UsbRecv (hDev,　rdata, &rleng);　//DI 読出コマンドの応答データを受信

備　考	デバイスに対する USB 受信は，デバイスに対する有効な読出コマンドの発行と必ず対で使用する必要があります. 不正な受信サイズや受信操作でアクセスするとﾀｲﾑｱｳﾄ等でエラーとなります.

8. サンプルプログラム

本製品では，ドライバ関数の使用方法を解説する目的でサンプルプログラムを添付しています．
サンプルプログラムには各言語のプロジェクトファイルとソースコードファイルが添付されています．
さらに動作確認用として，C/C++のサンプルプログラムには実行ファイルが添付されています．
サンプルプログラムの使用にあたっては，ユーザーが開発でご使用する Visual Studio に，添付されているプロジェクトファイルをインポートしてご使用ください．
なおサンプルプログラムは次の2種類の言語で作成したものを添付しています．これらはほぼ同一の画面表示と操作になっています．
以降の説明では，(Visual C/C++ 用コードを用いてます．

(1) Visual C/C++ 用コード　　【 sud46400.vcproj 】
(2) Visual C# 用コード　　【 sud46404.csproj 】

≪ ご注意 ≫

(1) サンプルプログラムはハードディスク等にコピーしてご使用ください．
(2) 各サンプルプログラムを使用される場合，ユーザーにおいてプロジェクトファイルをインポート，構成を構成マネージャで設定したうえでビルドを行い，実行ファイルを生成して下さい．
(3) 添付されているサンプルプログラム以外の言語のコードが必要な場合，添付されているサンプルプログラムを参考にコードの置き換えまたは変換を手動にて行って作成して下さい．
(4) DIO464 を 2 枚以上で使用する場合，ボード ID は重複しないようにして下さい．

8.1 サンプルプログラムの起動と操作

サンプルプログラムの実行ファイル（sud46400.exe または sud46404.exe）を起動すると，次の画面が表示されます．

(1) デバイスの選択

サンプルプログラムでは，デバイス上の動作，操作，開始，終了を次の手順で行います．

① デバイス情報取得

・・・ケーブルを接続しデバイス電源を投入してから“デバイス情報取得”ボタンを押下します

② デバイスの選択（2 枚以上の場合）

・・・“Board ID”コンボボックスからオープンするデバイスを選択します．

デバイスが1台も認識できない場合は“NOT FOUND”エラーが表示され、リストは表示されません．

③ デバイスオープン

・・・デバイスオープン後 DIO 出力及び入力状態の表示が可能になります．

④ デバイスクローズ

・・・“デバイスクローズ”ボタンを使用します．

クローズ後は DO 出力操作及び DI 入力状態表示は不可になります．

（全出力ポートには'0'が書き込まれ出力は全 OFF になります）

⑤ サンプルプログラムの終了（デバイスクローズ後）

・・・画面右上の×で行います．

(2) デバイス上の操作と表示

デバイスオープンを行うと次の画面となり，各状態の表示およびデバイス操作が可能になります．

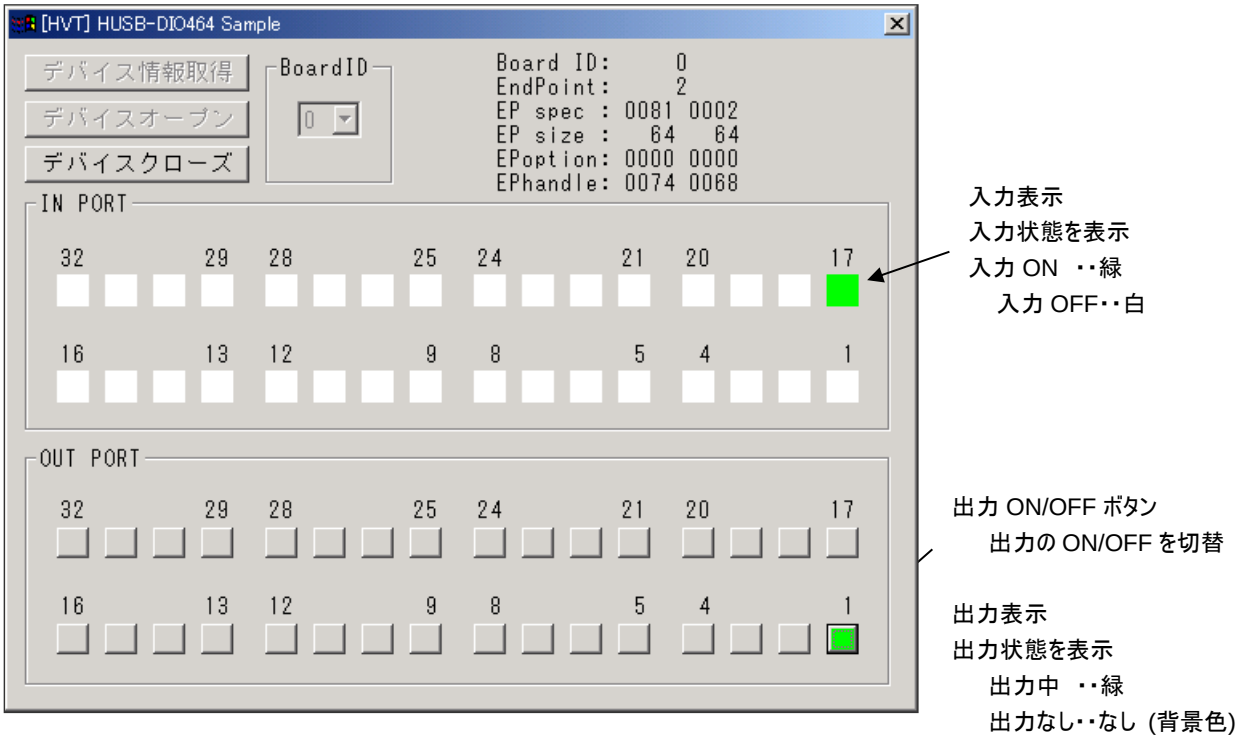

図 8-1.1 サンプルプログラム表示画面

9. USB コマンド資料

USB 送信関数および USB 受信関数を使って下記コマンドの送信および応答受信を行うことにより，既存関数にない機能を使用することが可能です．これらコマンドを使用する場合は，送受信フォーマットに間違いがないことを十分に確認したうえでご使用ください．間違ったコマンドを送った場合は以降のデバイス操作が正常に行えなくなることがあります．さらに応答データの受信データ数を間違えた場合 PC がハングアップする場合があります．

なお USB 送受信関数を使った場合のこれらコマンドは HUSB-DIO464U/ HETN-DIO864T/ HWIF-DIO864WIF 専用のコマンドフォーマットです．***HUSB-DIO464v2 とはコマンドフォーマットに互換性が無く使用できません***のでご注意ください．

9.1 USB コマンド一覧

No.	機能名称	ｺﾏﾝﾄﾞｺｰﾄﾞ	ﾃﾞｰﾀｻｲｽﾞ 上段:ｺﾏﾝﾄ送信 下段:応答受信	詳細
	デバイスアクセスコマンド			
1	DO ポート書込指令	90h	6 0	32ﾋﾞｯﾄの出力データを書き込みます．
2	DO ポート読出指令	91h	2 4	DOポートに書きこんだ32ビットの出力データを読み出します．
3	DO ポートビット書込指令	92h	10 0	32ビットの出力データをビット単位で指定して書き込みます．
4	DI ポート読出指令	93h	2 4	入力データの種類を指定して32ビットの入力データを読み出します
5	DO ポートビット書込／DIO ポート読出指令	95h	10 8	32ビットの出力データをビット単位で指定したデータの書き込みと同時に，入力データの種類を指定して32ビットの入力データを読み出しおよび書きこんだ出力データを読み戻します
6	DI ラッチデータクリア	98h	6 0	入力データの種類を指定してラッチされている入力データをビット単位でクリアします．
7	DO 同期出力データクリア	9Ah	2 0	全同期出力データをクリアします．
	動作設定コマンド			
8	レベルラッチ入力有効設定書込指令	A0h	6 0	レベルラッチの種類を指定して割り込みの有効／無効をビット単位で設定します．
9	レベルラッチ入力有効設定読出指令	A1h	2 4	レベルラッチ入力イネーブル設定を読み出します．
10	入力フィルター設定 書込指令	A2h	6 0	入力データ32ビットを4ビット単位でフィルター値を設定します．
11	入力フィルター設定 読出指令	A3h	2 4	入力フィルター設定を読み出します．
12	同期入力信号設定 書込指令	A4h	2 0	同期入力データ取り込みに使用する同期信号を設定します．
13	同期入力信号設定 読出指令	A5h	2 2	同期入力信号設定を読み出します．
14	出力データ選択設定 書込指令	A6h	6 0	出力データに出力するデータ種類をビット単位で指定します．
15	出力データ選択設定 読出指令	A7h	2 4	出力データ選択設定を読み出します．
16	同期出力トリガー信号 設定書込指令	A8h	2 0	同期出力に使用するトリガー信号を設定します．
17	同期出力トリガー信号 設定読出指令	A9h	2 2	同期出力に使用するトリガー信号設定を読み出します．

表 9-1.1 USB コマンド一覧

9.2 デバイスアクセスコマンドフォーマット

9.2.1 DO ポート書込指令

- コマンドデータ（送信）［6 Byte］

+0	+1	+2	+3	+4	+5
コマンド (90h)	0	DO ポート書き込み値			
		Bit7～0	Bit15～8	Bit23～16	Bit31～24

- 応答データ（受信）　なし

9.2.2 DO ポート読出指令

- コマンドデータ（送信）［2 Byte］

+0	+1
コマンド (91h)	0

- 応答データ（受信）［4 Byte ］

+0	+1	+2	+3
DO ポート読み出し値			
Bit7～0	Bit15～8	Bit23～16	Bit31～24

9.2.3 DO ポートビット書込指令

- コマンドデータ（送信）［10 Byte］

+0	+1	+2	+3	+4	+5
コマンド (92h)	0	DO 書き込み値マスクビット（0：アンマスク / 1：マスク）			
		OUT7～0	OUT15～8	OUT23～16	OUT31～24

+6	+7	+8	+9
DO ポート書き込み値			
Bit7～0	Bit15～8	Bit23～16	Bit31～24

- 応答データ（受信）　なし

9.2.4 DI ポート読出指令

● コマンドデータ（送信）［2 Byte］

+0	+1
コマンド (93h)	読出対象 入力データ

読出対象入力データ番号
0･･･入力ソースデータ（フィルタを通る前の入力データ）
1･･･フィルタリングデータ
2･･･H レベルラッチデータ
3･･･L レベルラッチデータ
4･･･立ち上がりエッジ検出データ
5･･･立ち下がりエッジ検出データ
6･･･両エッジ検出データ
7･･･同期入力データ（同期入力条件でラッチされた入力データ）

● 応答データ（受信）［4 Byte ］

+0	+1	+2	+3
DIポート読み出し値			
Bit7～0	Bit15～8	Bit23～16	Bit31～24

9.2.5 DO ポートビット書込／DIO ポート読出指令

● コマンドデータ（送信）［10 Byte］

+0	+1	+2	+3	+4	+5
コマンド (95h)	読出対象 入力データ	DO 書き込み値マスクビット（0：アンマスク / 1：マスク）			
		OUT7～0	OUT15～8	OUT23～16	OUT31～24

+6	+7	+8	+9
DO ポート書き込み値			
Bit7～0	Bit15～8	Bit23～16	Bit31～24

読出対象入力データ番号
0･･･入力ソースデータ（フィルタを通る前の入力データ）
1･･･フィルタリングデータ
2･･･H レベルラッチデータ
3･･･L レベルラッチデータ
4･･･立ち上がりエッジ検出データ
5･･･立ち下がりエッジ検出データ
6･･･両エッジ検出データ
7･･･同期入力データ（同期入力条件でラッチされたデータ）

● 応答データ（受信）［8 Byte ］

+0	+1	+2	+3	+4	+5	+6	+7
DIポート読み出し値				DO ポート読み出し値			
Bit7～0	Bit15～8	Bit23～16	Bit31～24	Bit7～0	Bit15～8	Bit23～16	Bit31～24

9.2.6　DI ラッチデータクリア指令

- コマンドデータ（送信）［6 Byte］

+0	+1	+2	+3	+4	+5
コマンド (98h)	クリア対象 ラッチデータ	クリアマスクビット（0：アンマスク（クリア） / 1：マスク）			
		IN7～0	IN15～8	IN23～16	IN31～24

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0	0	同期 入力	両エッジ	立ち下がり エッジ	立ち上がり エッジ	L レベル	H レベル

※同期入力データのラッチクリアはマスクビット無効

- 応答データ（受信）　なし

9.2.7　DO 同期出力データクリア指令

- コマンドデータ（送信）［2 Byte］

+0	+1
コマンド (9Ah)	クリアデータ

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0	0	0	0	0	0	0	出力クリア

- 応答データ（受信）　なし

9.3 動作設定コマンドフォーマット

9.3.1 レベルラッチ入力有効設定書込指令

- コマンドデータ（送信）［6 Byte］

+0	+1
コマンド (A0h)	設定対象 入力データ

→ 有効設定対象入力データ番号
2･･･H レベルラッチデータ
3･･･L レベルラッチデータ

+2	+3	+4	+5
レベルラッチ入力有効設定値（1：有効）			
Bit7～0	Bit15～8	Bit23～16	Bit31～24

※初期値はH/L共に全ビット無効

- 応答データ（受信）　なし

9.3.2 レベルラッチ入力有効設定読出指令

- コマンドデータ（送信）［2 Byte］

+0	+1
コマンド (A1h)	読出対象 入力データ

→ 有効設定読出対象入力データ番号
2･･･H レベルラッチデータ
3･･･L レベルラッチデータ

- 応答データ（受信）［4 Byte］し

+0	+1	+2	+3
レベルラッチ入力有効設定読み出し値			
Bit7～0	Bit15～8	Bit23～16	Bit31～24

9.3.3　入力フィルター設定書込指令

- コマンドデータ（送信）［6 Byte］

+0	+1	+2	+3	+4	+5
コマンド (A2h)	設定 イネーブル	フィルター設定値			
		Bit7～0	Bit15～8	Bit23～16	Bit31～24

設定イネーブル（4 ビット単位で設定可能）
IN4 -1 ・・・0x01　　IN20-17・・・0x10
IN8 -5 ・・・0x02　　IN24-21・・・0x20
IN12-9 ・・・0x04　　IN28-25・・・0x40
IN16-13・・・0x08　　IN32-29・・・0x80

フィルター設定値(4 ビット)
0・・・フィルターなし　　8・・・400μs
1・・・5μs　　9・・・800μs
2・・・10μs　　10・・・1ms
3・・・20μs　　11・・・4ms
4・・・40μs　　12・・・8ms
5・・・80μs　　13・・・16ms
6・・・100μs　　14・・・32ms
7・・・200μs　　15・・・64ms

※

Bit1	Bit1	Bit1	Bit1	Bit1	Bit1	Bit7	Bit8	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
IN6-13 フィルター値				IN12-9 フィルター値				IN8-5 ィルター値				IN4-1 フィルター値			
Bit3	Bit3	Bit2	Bit2	Bit2	Bit2	Bit2	Bit2	Bit2	Bit2	Bit2	Bit2	Bit1	bit1	Bit1	Bit1
IN32-29 フィルター値				IN28-25 フィルター値				IN24-21 ィルター値				IN20-17 フィルター値			

ビット1ms

- 応答データ（受信）　なし

9.3.4　入力フィルター設定読出指令

- コマンドデータ（送信）［2 Byte］

+0	+1
コマンド (A3h)	0

- 応答データ（受信）［4 Byte］し

+0	+1	+2	+3
フィルター設定読み出し値			
Bit7～0	Bit15～8	Bit23～16	Bit31～ 24

9.3.5　同期入力信号設定書込指令

- コマンドデータ（送信）［2 Byte］

+0	+1
コマンド (A4h)	同期入力 信号

同期入力信号設定値(5 ビット)
00h･･･IN1 ～1Fh･･･IN32

- 応答データ（受信）　なし

9.3.6　同期入力信号設定読出指令

- コマンドデータ（送信）［2 Byte］

+0	+1
コマンド (A5h)	0

- 応答データ（受信）［2 Byte］し

+0	+1
同期入力 信号設定 読み出し値	予約 (0)

9.3.7　出力データ選択設定書込指令

- コマンドデータ（送信）［6 Byte］

+0	+1
コマンド (A6h)	0

+2	+3	+4	+5
出力データ選択設定値（0：通常出力 1：同期出力）			
Bit7～0	Bit15～8	Bit23～16	Bit31～24

- 応答データ（受信）　なし

9.3.8　出力データ選択設定読出指令

- コマンドデータ（送信）［2 Byte］

+0	+1
コマンド (A7h)	0

- 応答データ（受信）［4 Byte］し

+0	+1	+2	+3
出力データ選択設定読み出し値			
Bit7～0	Bit15～8	Bit23～16	Bit31～24

9.3.9　同期出力トリガー信号設定書込指令

- ホストコマンド（送信）［2Byte］

+0	+1
コマンド (A8h)	同期出力 信号

同期出力信号に」同期する入力信号設定値
00h･･･IN1　～1Fh･･･IN32

- 応答データ（受信）　なし

9.3.10 同期出力トリガー信号設定読出指令

- コマンドデータ（送信）［2 Byte］

+0	+1
コマンド (A9h)	0

- 応答データ（受信）［2 Byte］し

+0	+1
同期出力 信号設定 読み出し値	予約 (0)

10.更新履歴

日付	版	更新内容
2019/04/24	1.00	新規作成

表 10-1.1 更新履歴