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

HETN-DIO864T
HWIF-DIO864W

ユーザーズマニュアル

〈ソフトウェア・Ethernet/WiFi 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 + 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 25 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．

⚠ 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>
(3) ユーザーズマニュアル　<ソフトウェア・Ethernet/WiFi 編 Windows 版> ※本書

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) → 本マニュアル説明対象外
HUSB-DIO464U(D) ・・・ USB＋DIN 取付台 → 本マニュアル説明対象外
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うち，Ethernet または WiFi インターフェースを持つ製品に関してのソフトウェアマニュアル（Ethernet/WiFi 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，USB 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・USB 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 DIO864 と呼称する場合があります．

2. ソフトウェア概要

本製品を Ethernet/WiFi インターフェースによって使用するには，常駐プログラムおよび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プログラム，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

本製品では以下のプログラムおよび DLL，ライブラリが必要です．さらに DLL に含まれないを使ってより高度な Ethernet/WiFi 接続機能を持ったライブラリ関数があります．
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．

(1) 常駐プログラム

常駐プログラムは Ethernet/WiFi インターフェースを使用する場合においてホストとデバイス間の通信制御を行うプログラムです．
常駐プログラムは弊社製 Ethernet/WiFi 全製品に共通して使用するプログラムです．
このプログラムはアプリケーションと同じパソコン内（ローカル PC）で実行することも，同一ネットワーク内にある他のパソコン上（リモート PC）で実行することも可能です．またマルチスレッド，マルチプロセス，複数 PC からのアクセスが可能です．
（デバイスに対するコマンド実行の排他処理および排他関連エラーの処理はユーザー様にて行っていただく必要があります）
本マニュアルではこのプログラムを“常駐プログラム”と呼称します．
このプログラムは以下のインストーラによってネットワーク上の PC1 台にインストールします．同一ネットワーク上の複数 PC にインストールするとデバイスの取り合いが起きますので必ず 1 台の PC のみにインストールが必要です．
◇Windows 7 以降 - 64bit 版用・・CD フォルダ Ethernet¥rpg¥x64¥内 setup.exe
◇Windows 7 以降 - 32bit 版用・・CD フォルダ Ethernet ¥ rpg¥x86¥内 setup.exe
インストーラにより hsocket.exe および hsktuif.dll がインストールされます．

(2) 常駐プログラム I/F 用 DLL

本デバイスで使用する DLL は，常駐プログラムを介してデバイスをコントロールするための DLL です．
この DLL は USB インターフェース用のインストーラではインストールされませんのでご注意ください．
◇Windows 7 以降 - 64bit 版用・・CD フォルダ Ethernet¥dll¥x64 内 setup.exe
◇Windows 7 以降 - 32bit 版用・・CD フォルダ Ethernet ¥dll¥x86 内 setup.exe
インストーラにより hedio864.dll がインストールされます．

(3) ライブラリ関数

常駐プログラムに接続するためのプログラム起動 PC の IP アドレス自動検出等の検索を自動的に行う関数がライブラリ関数としてソースファイルで提供しています．ユーザはライブラリ関数を含んだファイルを一緒にビルドすることで使用します．
◇CD フォルダ Ethernet¥include1¥vc¥Ed864l1a.c および Ed864l1a.h または Ethernet¥include1¥vcs¥Ed864l1a.cs

4. 添付ソフトウェアの構成

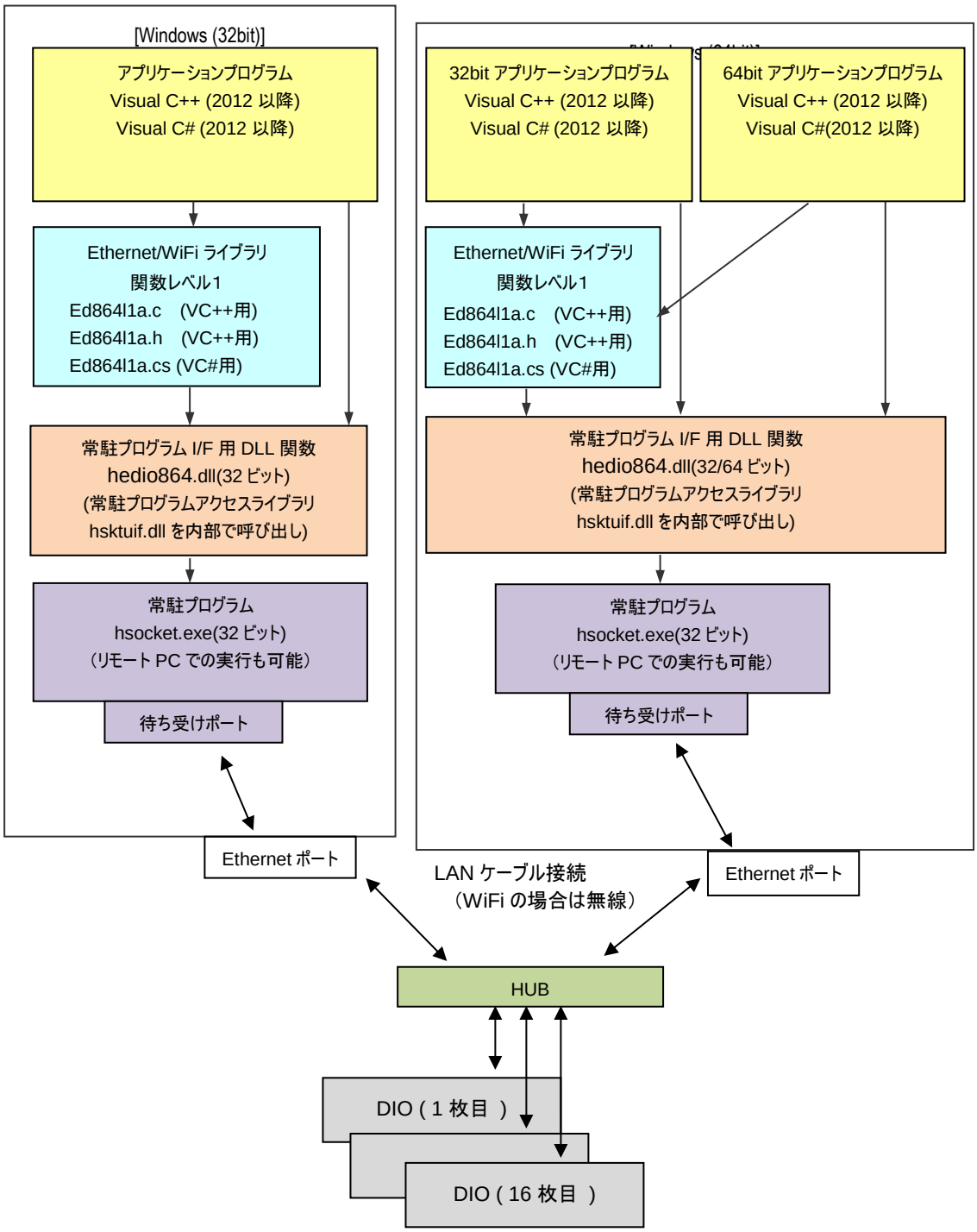


図 4.1-1 ソフトウェア構成

5. インストールとアンインストール

DIO864 を Ethernet/WiFi で使用するには，常駐プログラムおよび常駐プログラム I/F 用 DLL のインストールが必要です．
インストールは本製品に添付されている CD から行います．本製品の旧バージョンソフトウェアが既にインストールされている場合は，インストール前に旧バージョンを全てアンインストールする必要があります．
さらに Ethernet/WiFi を使った他製品のソフトウェアが既にインストールされている場合も，既にインストールされている関連ソフトウェアを一度全てアンインストールし，本製品と共に最新のソフトウェアを再インストールする必要があります．
常駐プログラムについては，弊社 Ethernet/WiFi 全製品で共通に使用するプログラムですので，製品に関係なく最新の常駐プログラムを1回インストールします．

5.1 常駐プログラムのインストール

DIO864 は Ethernet または WiFi 使用時は常駐プログラムを経由して PC と Ethernet/WiFi 通信を行います．そのため本製品を使用するには常駐プログラムが必要です．常駐プログラムは．弊社 Ethernet/WiFi 全製品の通信の交通整理を行う働きを持ち．そのため最新のバージョンが常にインストールされている必要があります．既に他製品のインストール等で最新の常駐プログラムがインストールされている場合，常駐プログラムのインストールは必要ありません．

Windows7 以降へのインストール

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 “¥Ethernet¥rpg¥Setup.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 正常に終了したらインストールは終了です．
次の常駐プログラム I/F 用 DLL を行います．

5.2 常駐プログラムI／F用 DLL のインストール

常駐プログラム I/F 用 DLL は，常駐プログラムを介して DIO864 をコントロールする関数群です．
この DLL は常駐プログラムと共に動作する DIO864 専用のソフトウェアです．

Windows7 以降へのインストール

① 旧バージョンが既にインストールされている場合はアンインストールします．
② OS が 32bit 版の場合，CD ドライブ“¥Ethernet¥dll¥x86”を開きます．
OS が 64bit 版の場合，CD ドライブ“¥ Ethernet ¥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 正常に終了したら終了です．

5.3 常駐プログラム・DLL のアンインストール

DIO864 に添付された CD からインストールしたプログラムおよび 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 常駐プログラムを終了し，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．
この時，PC 上で弊社 Ethernet あるいは WiFi を使用した他製品を使用中の場合，常駐プログラムはそれら製品でも使用するため，アンインストールは行わないでください．（DLL のみ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） ※
常駐プログラム	Hivertec Windows Socket Program Package for HETN and HWIF Seriese Board
DIO864 用 常駐プログラム I/F DLL	Hivertec Library Package for HETN-DIO864T/HWIF-DIO864W

※常駐プログラムおよび 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 5.3-1 “アプリと機能”上でのアプリケーション登録名

図 5.3-1 “アプリと機能”画面（Windows10）

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は，常駐プログラムを終了してからデバイス DIO864 等の電源を OFF して行ってください．常駐プログラムが起動したままアンインストールを行った場合，正常にアンインストールされない場合があります．

5.4 常駐プログラムの設定

インストール直後の常駐プログラムには以下のデフォルト値が設定されています.
通常はこのままの値でご使用ください.
変更する場合は各項目の意味と設定範囲をご理解の上で操作してください.
なお設定画面は Windows 画面右下のアイコンのコンテンツメニューから“設定”を選択して開きます. ここでの変更は変更を行った PC 上で常駐プログラムを RUN させる場合に有効です. 他 PC 上で RUN させる場合は当該 PC 上で設定を変更する必要があります.

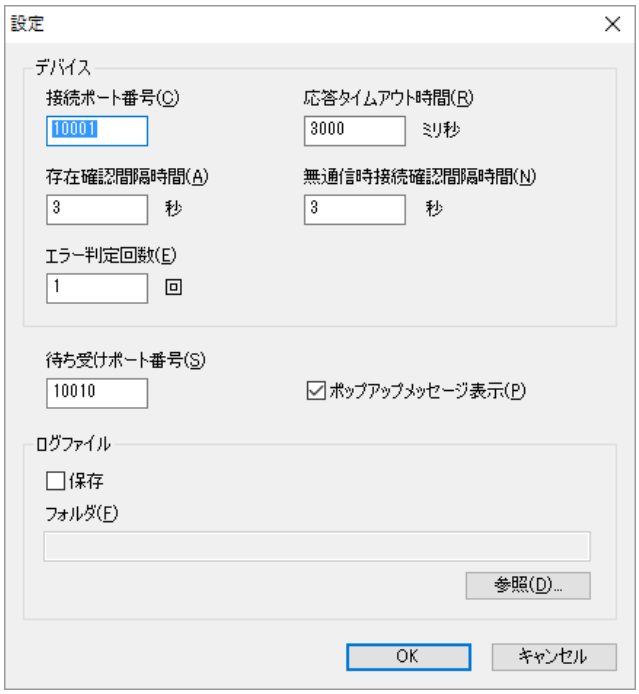


図 5.4-1 常駐プログラム設定画面

●接続ポート番号
・・・CPD に接続する時のポート番号を設定します. この値は CPD で固定になっているので変更しないでください
●応答タイムアウト
・・・常駐プログラムからは定期的にアライブパケットが送信されます. また CPD に対してパケットを転送した後は, その応答パケットを待っています. これらが設定した時間以内に返ってこない場合はエラーとしてドライバ関数実行時にエラーを返します.
●存在確認間隔時間
・・・アライブパケットを送る間隔を設定します. アライブパケットを CPD に送って, その応答が無い場合は CPD が存在していないと判断します. この値は 5 秒以上に設定しないでください.
●無通信時接続確認間隔時間
・・・存在確認間隔時間と重複するため使用していませんが, 存在確認間隔時間と同じ設定にしてください.
●エラー判定回数
・・・アライブパケットに応答しなかった場合, ここで設定した回数以上連続したら“切断”と判定します.
●ログファイル
・・・常駐プログラムに関するアクション(接続や切断, 送受信パケット交換など)の履歴を記録する場合に有効にします.
この機能は主にデバッグ用に設けられたものなので通常は使用しません. さらにこの履歴は自動的には削除されませんのでご使用する場合は定期的に削除するなどの管理が必要になりますのでご注意ください.

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

DIO864 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．

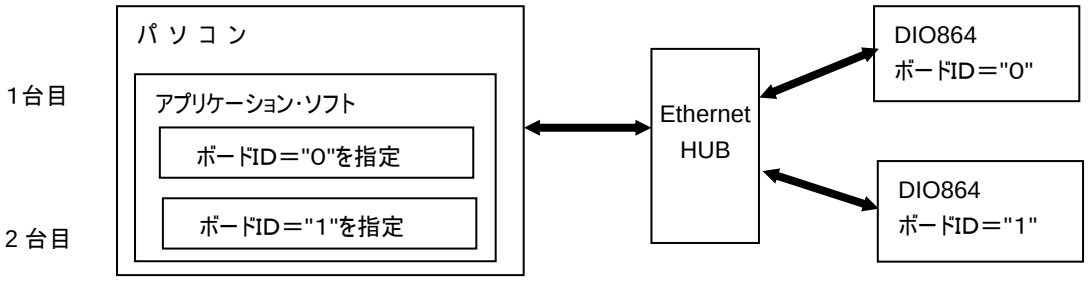


図 6.1-1 PC と複数枚デバイスの接続例

(1) HETN-DIO864T/HWIF-DIO864W の指定
LAN ケーブルまたは WiFi で接続された2台以上の DIO864 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 DIO864 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「¥Ethernet¥sample¥Vc¥Release¥ sed86400.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 は重複しないよう設定してください．
※ DIO864 を Ethernet または WiFi で使用する場合，複数プロセスからのアクセスが可能です．
ただしデバイス間の排他処理は必要です．

6.2 デバイスの着脱と認識について

DIO864 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	DIO864 の認識	デバイス情報取得時に行われます
2	起動時以降に接続されたデバイス	認識可能ですがデバイス情報の再取得が必要です．
3	起動時以降に切り離されたデバイス	デバイスオープン後にそのまま切り離されると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．クローズ状態での切り離しは問題ありません．
4	アプリケーション実行中に任意の Ethernet/WiFi 接続	接続可能ですが，デバイス情報取得後に着脱したデバイスがある場合はデバイス情報の再取得が必要です．
5	DIO864（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 10 秒以上経過してから使用（通信）します． また，デバイス電源 OFF またはケーブル切り離しを行った後の再接続では 3 秒以上待ってから接続してください．

表 6.2-1 Ethernet/WiFi 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 DIO864 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，デバイス情報を使用して

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 _HETNDEVINF {      // ユーザインターフェース用デバイス情報
        short DevTyp;             // デバイスタイプ
        short BrdID;              // ボード ID
        BYTE    IpAdr[4];         // IP アドレス
        BYTE    MacAdr[6];        // MAC アドレス
        BYTE    sp1[2];
} HETNDEVINF, *PHETNDEVINF;
```

[Visual C#]

```
// デバイス情報
[StructLayout(LayoutKind.Sequential)]
public struct HETNDEVINF
{
        Public  short   DevTyp;     // デバイスタイプ
        Public  short   BrdID;      // ボードID
        Public  byte    IpAdr1;     // IPアドレス
        Public  byte    IpAdr2;
        Public  byte    IpAdr3;
        Public  byte    IpAdr4;
        Public  byte    MacAdr1;    // MACアドレス
        Public  byte    MacAdr2;
        Public  byte    MacAdr3;
        Public  byte    MacAdr4;
        Public  byte    MacAdr5;
        Public  byte    MacAdr6;
        Public  byte    sp1;
        Public  byte    sp2;
}
```

6.3.2　デバイスドライバ I/F 関数／ライブラリ関数の使用

(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

◇32 ビット OS 用デバイスドライバ I/F 関数用インポートファイル ・・・ hedio864.lib

◇64 ビット用デバイスドライバ I/F 関数インポート用ファイル ・・・ hedio864x64.lib

◇ライブラリ関数ファイル ・・・ Ed864l1a.c

Ethernet/WiFi アクセス用ライブラリ関数（C ソースコードにて提供）

■インクルードファイル

◇デバイスドライバ I/F 関数用ヘッダファイル ・・・ hedio864.h

◇ライブラリ関数用ヘッダファイル ・・・ Ed864l1a.h

(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.

◇デバイスドライバ I/F 関数用ソースコードファイル ・・・ **hedio864.cs**

◇ライブラリ関数用ソースコードファイル ・・・ Ed864l1a**.cs**

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／ライブラリ関数

7.1 関数一覧

No ※	関数名称	機　能	備考
デバイスドライバーI/F 関数			
D1	ed864_GetDeviceCount	デバイス枚数の取得	
D2	ed864_GetDeviceInfo	デバイス情報の取得	
D3	ed864_OpenDevice	デバイスオープン	
D4	ed864_CloseDevice	デバイスクローズ	
D5	ed864_rInB	入力ポートからの 1 バイトデータ読出	
D6	ed864_rOutB	出力ポートからの 1 バイトデータ読出	
D7	ed864_wOutB	出力ポートへの　1 バイトデータ書込	
D8	ed864_rInW	入力ポートからの 2 バイトデータ読出	
D9	ed864_rOutW	出力ポートからの 2 バイトデータ読出	
D10	ed864_wOutW	出力ポートへの　2 バイトデータ書込	
D11	ed864_rInDW	入力ポートからの 4 バイトデータ読出	
D12	ed864_rOutDW	出力ポートからの 4 バイトデータ読出	
D13	ed864_wOutDW	出力ポートへの　4バイトデータ書込	
D14	ed864_InOutDW	出力ポート 4 バイト書込／入力ポート 4 バイト読出	
D15	ed864_SetFilter	入力ポートのﾌｨﾙﾀの設定	
D16	dio864_GetBoardCode	デバイス固有コードの取得	
D17	dio864_rVersion	デバイスバージョンの取得	
D18	ed864_SetProgLink	常駐プログラムリンク設定	
D19	ed864_GetProgVersion	常駐プログラムバージョン取得	
D20	ed864_EtnSearch	常駐プログラム検索	
D21	ed864_EtnSend	Ethernet/WiFi 送信	
D22	ed864_EtnRecv	Ethernet/WiFi 受信	
D23	ed864_GetOpenDevCount	デバイスオープン済み接続数の取得	
ライブラリ関数			
L1	hed864_ProgLinkSet	常駐プログラムリンク設定	
L2	hed864_GetProgVersion	常駐プログラムソフトウェアバージョン読出	
L3	hed864__GetCmdError	コマンド実行エラーの取得	

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周辺機器，構成等により若干の違いがありますのでご参考用としてご覧ください．

No	関数名称	Ethernet HETN-DIO864T 通信時間(ミリ秒)	WiFi HETN-DIO864T 通信時間(ミリ秒)	参考値 USB HUSB-DIO464U: 通信時間(ミリ秒)
D1	ed864_GetDeviceCount	----	----	----
D2	ed864_GetDeviceInfo	----	----	----
D3	ed864_OpenDevice	----	----	----
D4	ed864_CloseDevice	----	----	----
D5	ed864_rInB	20.0	15.0	3.0
D6	ed864_rOutB	19.5	15.0	3.0
D7	ed864_wOutB	20.0	14.5	0.2
D8	ed864_rInW	19.5	14.0	3.0
D9	ed864_rOutW	20.0	14.5	3.0
D10	ed864_wOutW	20.5	15.0	0.2
D11	ed864_rInDW	19.5	15.0	3.0
D12	ed864_rOutDW	20.0	14.5	3.0
D13	ed864_wOutDW	19.5	14.5	0.2
D14	ed864_InOutDW	21.5	15.0	0.5
D15	ed864_SetFilter	19.5	14.5	0.5
D16	dio864_GetBoardCode	----	----	----
D17	dio864_rVersion	19.5	15.0	3.0
D18	ed864_SetProgLink	----	----	
D17	ed864_GetProgVersion	5.0	4.5	
D19	ed864_EtnSearch	----	----	
D20	ed864_EtnSend	5.0	5.0	0.2
D21	ed864_EtnSend + ed864_EtnRecv	21.0	15.0	0.4
D22	ed864_GetOpenDevCount	----	----	
L1	hed864_ProgLinkSet	----	----	----
L2	hed864_GetProgVersion	----	----	----
L3	hed864_GetCmdError	----	----	----

表 7.2-1 関数通信時間

注意 1 ：項目 D16,D18 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 D20 は設定されたﾀｲﾑｱｳﾄ時間まで検索を続けるため処理時間はﾀｲﾑｱｳﾄ時間と同じになります．

注意 3 ：項目 D21，D22 はデータ長によって時間が変わります．（項目 D21 は 10Byte のデータを送信した時の時間）

注意 4 ：項目 D22 は単独での計測ができないため項目 D21 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 D14 の機能を本関数で実行）

注意 5 ：項目 D21,D22 の参考値である USB インターフェースの処理時間は関数 dio464_EtnSend／dio464_EtnRecv を使ってのものです

7.3 デバイスアクセスの準備手順

デバイスを使う場合は，まず最初に“常駐プログラム”が起動していることが必要です．起動している状態で常駐プログラムを検索し，常駐プログラムへのアクセスに必要な各情報を取得，常駐プログラムへのリンク設定を行います．
これで Ethernet/ WiFi を使用してのデバイスへのアクセスが可能になるので，DIO に対してのオープンや初期化を行います．
以下に手順を示します．

(1) 常駐プログラムの検索とリンク設定

常駐プログラムがネットワーク上のどのパソコン上で実行されているか検索して，その IP アドレスやポート番号などアクセスに必要な情報を取得します．取得した情報を使って常駐プログラムにリンク設定を行ないます．

◆ ed864_EtnSearch()・・・起動している常駐プログラムをネットワーク上から検索して，その情報を取得します．

◆ ed864_SetProgLink()・・・アプリケーションが常駐プログラムにアクセスするために必要なリンク設定を行います．

上記 2 つの関数を使用して自動的あるいは指定したアドレスの常駐プログラムに接続する応用関数がライブラリに用意されており，この関数を使うことで複雑な処理を 1 関数で記述できます．

◆◆ hed864_ProgLinkSet ()

(2) Ethernet/WiFi 全デバイスのデバイス情報取得

常駐プログラムがネットワーク上から検出した Ethernet/WiFi 接続している DIO のデバイス情報を取得します．この関数によって取得したデバイス個数や情報は最新の状態に更新され，アプリケーションを実行するパソコン内部に保持されます．

◆ ed864_GetDeviceCount()・・・Ethernet ポートあるいは WiFi に接続されている DIO の個数取得

◆ ed864_GetDeviceInfo() ・・・Ethernet ポートあるいは WiFi に接続されている DIO のデバイス情報を取得

(3) DIO 毎にデバイスオープン

使用する DIO の“ボードタイプ”，“ボード ID”をデバイスオープン関数に渡します．DIO がオープンされると常駐プログラムを介してパソコンと DIO はソケットで接続され，以降はパソコンから DIO の操作ができるようになります．
オープンに失敗した場合は同一 DIO に対してデバイスクローズが必要です．

◆ ed864_OpenDevice() ・・・DIO のオープン処理

(4) デバイスの信号処理方法の初期設定

各レジスタの初期化を行う前に，次の関数でデバイス使用条件（ボードの信号処理方法）の設定を行います．

◆ ed864_wPortB() ・・・各 DIO デバイス使用条件（ボードの信号処理方法）の設定

(5) 各 DIO・各軸の初期化

上記設定以降に，使用する全 DIO の各軸の初期化を行います．
「DIO ボードシリーズ ユーザーズマニュアル〈運用編〉」を参照し，各レジスタを設定して下さい．
これにより，通常動作としての各軸パルス出力動作等が可能となります．

(6) 全デバイスのデバイスクローズ

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は，オープンした DIO の「クローズ処理」を行って下さい．

◆ ed864_CloseDevice() ・・・DIO のクローズ処理

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同一デバイスにアクセスするような場合は同期(排他処理)が必要です．
DIO では常駐プログラムがデバイス毎に生成されたソケット接続を管理し，ホスト PC と各デバイス間のデータ通信の交通整理を行っています．したがって通常ユーザはデバイス間の排他を意識せずご利用することができます．
ただしデバイスドライバ I/F 関数あるいはライブラリ関数を使って，複数スレッドから同一デバイスに対して制御を行う場合は
使用する関数の種類が異なる場合でも排他処理が必要です．

例えば二つのスレッドで同一デバイスを制御している場合，一つのスレッドが DI 読み出し、もう一方のスレッドが DO 書き込みを行っているような場合、排他されていないと DI 読み出しと DO 書き込みの要求や応答通信が混在してしまい正常にデータが伝わらなくなる場合が発生します．
この様な状況を避けるため，それぞれのスレッド内でミューテックスやクリティカルセクションを使用して関数実行を排他制御します．
複数デバイスの同時制御に際しての同一関数使用については排他制御は必要ありません．

<イメージ>

```
Thread1()
{
   wType = 0x13;
   wBid = 5;

   Lock();
   ed864_rInDW(wType, wBid, &dt );          //HWIF864W,BID=5 の DI 読出
   UnLock();
}

Thread2()
{
   wType = 0x13;
   wBid = 5;

   Lock();
   ed864_wOutDW(wType, wBid , 0x11223344 );  // HWIF864W,BID=5 の DO 書込
   UnLock();
}
```

(2) リトライ処理について

本製品に添付されているデバイスドライバ I/F 関数，ライブラリ関数は正常に実行されなかった場合エラーを返します．
通常これらのエラー発生時はアプリケーション側で必要に応じたリカバリ処理を行ないますが，Ethernet/WiFi タイプは PC の能力や負荷,使用環境によって通信状態が変化します．特に WiFi タイプ製品では電波の状態が設置状態や環境,障害物の有無により瞬間的な通信の切断が起こりやすく，その時に関数実行が失敗する場合があります．
このため，関数実行で受信関係のエラーが発生した場合はリトライ処理を含めた処理を行うことを推奨いたします．

<イメージ>

```
while(err_cnt < 2) {      //リトライ回数 2 回まで実施
      ans = ed864_rInDW( hDev1, &dt ); (wType, wBid, wAxis, byCmd, dwData);
      if((ans != 0x40000) && (ans != ILLEGAL_RESPONSE)) {
            // 関数戻り値が” 受信タイムアウト(0x40000)”と” 受信データ不正（サイズ不一致）(0x400000)”
      //以外はリトライしないで終了
            break;
      }
      err_cnt++;
}
return ans;
```

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	記号表記	戻り値 16 進数表記 VC++ VC#	 VB VB.NET	異常内容と確認項目
1	INVALID_SOCHD	0x00010000	&H10000	無効なソケットハンドルを指定
2	NETDEV_NOTFOUND	0x00020000	&H20000	指定したデバイスが見つからない
3	RECEIVE_TIMEOUT	0x00040000	&H40000	データ受信でタイムアウト発生
4	BUFFER_OVERFLOW	0x00080000	&H80000	デバイス内部のデータ受信用バッファがオーバーフローした ◎システムが不安定になっている可能性がありますので，弊社サポートまでお問い合わせください
5	ILLEGAL_NETPARAM	0x00100000	&H100000	関数の引数の値が異常 ◇速度倍率設定値の範囲は 2～4095. ◇その他引数の設定値を確認
6	ILLEGAL_FORMAT	0x00200000	&H200000	コマンドの書式が以上 ◇コマンド列のデータ内容を確認
7	ILLEGAL_RESPONSE	0x00400000	&H400000	データ読み出しで予期しないデータを受信 ◎システムが不安定になっている可能性がありますので，弊社サポートまでお問い合わせください
8	ILLEGAL_NETDEVICE	0x00800000	&H800000	デバイスの種別とボード ID が重複しています． ◎このエラーはデバイス情報取得時またはデバイスオープン時に発生します． 各デバイスの設定を確認してください
9	ILLEGAL_NETACCESS	0x01000000	&H1000000	他の処理が送受信中（排他エラー） ◇割り込み処理やマルチスレッド，マルチプロセスなどで作成する場合は 適切な処理を施してください． ◇ほかの処理が使用を終えるまで待ちます

表 7.5-1 関数の戻り値

7.6 デバイスドライバ I/F 関数詳細

7.6.1 ed864_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 DIO864 の枚数を取得します.

開発環境	書　式
VC++	DWORD WINAPI　ed864_GetDeviceCount(DWORD*　count);
VC#	[DllImport("*libname*")] public static extern uint　ed864_GetDeviceCount(ref uint count);

引　数	説　明
count	取得した DIO864 枚数

VC++ 記述例	DWORD*　count;　　//DIO864 の枚数 DWORD　　ret;　　//関数の戻り値 ret = ed864_GetDeviceCount(&count);

備　考	

7.6.2 ed864_ 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 DIO864 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　ed864_GetDeviceInfo(DWORD* count, HETNDEVINF* HetnInf);
VC#	[DllImport("*libname* ")] public static extern uint　ed864_GetDeviceInfo(ref uint count, ref HETNDEVINF HetnInf);

引　数	説　明
count	取得した DIO864 枚数
HetnInf	取得するデバイス情報の格納先配列

VC++ 記述例	DWORD　　ret;　　//関数の戻り値 DWORD　　count = 2;　　//使用枚数は 2 HETNDEVINF　HetnInf[2];　　//2 枚の DIO864 のデバイス情報がセットされるべきエリア ret = ed864_GetDeviceInfo(&count, &HetnInf[0]);

備　考	

7.6.3　ed864_ OpenDevice() デバイスのオープン

機　能	渡したデバイス情報を持つ DIO864 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 DIO864 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　ed864_OpenDevice(DWORD * hDev, HETNDEVINF* hInf);
VC#	[DllImport("*libname* ")] public static extern uint　ed864_OpenDevice(ref uint hDev, ref HETNDEVINF hInf);

引　数	説　明
sType	ボードタイプ (0x12:HETN-DIO864T / 0x13:HWIF-DIO864W
sBid	ボード ID (0～15)
hInf	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 HETNDEVINF hInf;　　//オープンする DIO のデバイス情報格納先 ret = ed864_OpenDevice(0x12, 14, &hInf);　//HETN-DIO864T，BID=14 をｵｰﾌﾟﾝ

備　考	**デバイスのオープンに失敗した場合は，失敗した DIO に対してデバイスクローズを行って下さい．**

7.6.4　ed864_ CloseDeviceInfo() 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 DIO864 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 DIO864 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　ed864_CloseDevice(DWORD　hDev);
VC#	[DllImport("*libname* ")] public static extern uint　ed864_CloseDevice(uint hDev);

引　数	説　明
sType	ボードタイプ (0x12:HETN-DIO864T / 0x13:HWIF-DIO864W
sBid	ボード ID (0～15)

VC++ 記述例	DWORD　ret;　　//関数の戻り値 ret = ed864_CloseDevice(0x12, 14,);　//HETN-DIO864T，BID=14 をクローズ

備　考	

7.6.5　ed864_rInB()　入力ポートからの 1 バイトデータ読出

7.6.6　ed864_rOutB()　出力ポートからの 1 バイトデータ読出

7.6.7　ed864_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 DIO864 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　ed864_rInB (DWORD hDev, WORD port, WORD*　wrdt); DWORD WINAPI　ed864_rOutB(DWORD hDev, WORD port, WORD*　wrdt); DWORD WINAPI e d864_wOutB(DWORD hDev, WORD port, WORD　wwdt);
VC#	[DllImport("*libname*")] public static extern uint ed864_rInB(uint hDev, ushort port, ref ushort wrdt); [DllImport("*libname*")] public static extern uint ed864_rOutB (uint hDev, ushort port, ref ushort wrdt); [DllImport("*libname* ")] public static extern uint ed864_wOutB(uint hDev, ushort port, ushort wwdt);

引　数	説　明
sType	ボードタイプ (0x12:HETN-DIO864T / 0x13:HWIF-DIO864W
sBid	ボード ID (0～15)
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 = ed864_rInB(0x12, 14, 0, &dt);　// HETN-DIO864T, BID=14 の入力ポート 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　ed864_rInW()　入力ポートからの 2 バイトデータ読出

7.6.9　ed864_rOutW()　出力ポートからの 2 バイトデータ読出

7.6.10 ed864_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 DIO864 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　ed864_rInW (short sType, short sBid, WORD port, WORD*　wrdt); DWORD WINAPI　ed864_rOutW(short sType, short sBid, WORD port, WORD*　wrdt); DWORD WINAPI　ed864_wOutW(short sType, short sBid,, WORD port, WORD　wwdt);
VC#	[DllImport("*libname*")] public static extern uint ed864_rInW (short sType, short sBid,, ushort port, ref ushort wwdt); [DllImport("*libname*")] public static extern uint ed864_rOutW (short sType, short sBid, ushort port, ref ushort wwdt); [DllImport("*libname* ")] public static extern uint ed864_wOutW (short sType, short sBid, ushort port, ushort wwdt);

引　数	説　明
sType	ボードタイプ (0x12:HETN-DIO864T / 0x13:HWIF-DIO864W
sBid	ボード ID (0～15)
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 = ed864_rInW(0x12, 14, 0, &dt);　　//HETN-DIO864T, BID=14 の入力ポート 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 ed864_rInDW() 入力ポートからの 4 バイトデータ読出

7.6.12 ed864_rOutDW() 出力ポートからの 4 バイトデータ読出

7.6.13 ed864_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 DIO864 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　ed864_rInDW (short sType, short sBid, DWORD*　drdt); DWORD WINAPI　ed864_rOutDW(short sType, short sBid,, DWORD*　drdt); DWORD WINAPI　ed864_wOutDW(short sType, short sBid,, DWORD　dwdt);
VC#	[DllImport("hedio864.dll")] public static extern uint ed864_rInDW (short sType, short sBid, ref uint drdt); [DllImport("*libname*")] public static extern uint ed864_rOutDW (short sType, short sBid, ref uint drdt); [DllImport("*libname*")] public static extern uint ed864_wOutDW (short sType, short sBid, uint dwdt);

引　数	説　明
sType	ボードタイプ (0x12:HETN-DIO864T / 0x13:HWIF-DIO864W
sBid	ボード ID (0～15)
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 = ed864_rInDW(0x12, 14, &dt);　// HETN-DIO864T, BID=14 の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 ed864_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 DIO864 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　ed864_InOutDW (short sType, short sBid, DWORD* drdt, DWORD dwdt);
VC#	[DllImport("*libname*")] public static extern uint　ed864_InOutDW(short sType, short sBid, ref uint drdt, uint dwdt);

引　数	説　明
sType	ボードタイプ (0x12:HETN-DIO864T / 0x13:HWIF-DIO864W
sBid	ボード ID (0～15)
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バ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 = ed864_InOutDW(0x12, 14, &rdt, 0x55555555);// HETN-DIO864T，BID=14 へ
	//55555555h のデータ出力と入力ポート読み出し

備　考	

7.6.15 ed864_SetFilter()　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 DIO864 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）

開発環境	書　式
VC++	DWORD WINAPI　ed864_SetFilter (short sType, short sBid, WORD port, WORD wtmno);
VC#	[DllImport("*libname*")] public static extern uint　ed864_SetFilter (short sType, short sBid, ushort port, ushort wtmno);

引　数	説　明
sType	ボードタイプ (0x12:HETN-DIO864T / 0x13:HWIF-DIO864W
sBid	ボード ID (0～15)
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 = ed864_SetFilter(0x12, 14, 0x3, 10);　// HETN-DIO864T，BID=14 の //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.

7.6.16 ed864_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 DIO864 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　ed864_GetBoardCode (DWORD hDev, WORD* bcode);
VC#	[DllImport("*libname* ")] public static extern uint　e864_GetBoardCode (uint hDev, ref ushort bcode);

引　数	説　明
sType	ボードタイプ (0x12:HETN-DIO864T / 0x13:HWIF-DIO864W
sBid	ボード ID (0～15)
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 = ed864_GetBoardCode (0x12, 14, &bcode);　// HETN-DIO864T，BID=14 のボードコード取得

備　考	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.

7.6.17 ed864_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 DIO864 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　ed864_rVersion (DWORD hDev, WORD* ver);
VC#	[DllImport("*libname* ")] public static extern uint　ed864_rVersion (uint hDev, ref ushort ver);

引　数	説　明
sType	ボードタイプ (0x12:HETN-DIO864T / 0x13:HWIF-DIO864W
sBid	ボード ID (0～15)
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 = ed864_rVersion (0x12, 14, ver);　//

備　考	バージョン 1.2.3.4 の場合は 0x1234 を返します.

7.6.18 ed864_SetProgLink()　　常駐プログラムリンク設定

機　能	常駐プログラムの動作設定を行います.

開発環境	書　式
VC++	DWORD WINAPI　ed864_SetProgLink(BYTE* pIp, WORD port, WORD tmo);
VC#	[DllImport("*libname* ")] public static extern uint　ed864_ SetProgLink (byte[] pIp, ushort port, ushort tmo);

引　数	説　明
pIp	常駐プログラムが RUN している PC の IP アドレス
Port	常駐プログラム待ち受けポート番号
tmo	送受信タイムアウト設定

VC++ 記述例	DWORD ret;　　//関数の戻り値 Char* ip =”192.168.1.100”;　　//常駐プログラムが RUN する PC の IP アドレス WORD port=10010;　　//待ち受けするポート番号 WORD tmo = 3000　　//タイムアウト 3000ms ret = ed864_ SetProgLink (ip, port、tmo);

7.6.19 ed864_GetProgVersion()　　常駐プログラムのバージョン取得

機　能	常駐プログラムのバージョンを取得します

開発環境	書　式
VC++	DWORD WINAPI　ed864_GetProgVersion(WORD* pver, WORD* lver);
VC#	[DllImport("*libname* ")] public static extern uint　ed864_ GetProgVersion (ref ushort pver, ref ushort lver);

引　数	説　明
pver	常駐プログラムのバージョン
lver	常駐プログラムアクセスライブラリーのバージョン

VC++ 記述例	DWORD ret;　　//関数の戻り値 WORD* pver;　　//常駐プログラムのバージョン番号格納アドレス WORD* lver;　　//常駐プログラムアクセスライブラリーのバージョン番号格納アドレス ret = ed864_ GetProgVersion　(pver, lver); ※各バージョン番号は ver1.234 の場合 0x1234 と格納されます

7.6.20 ed864_EtnSearch()　　常駐プログラムの検索

機　能	起動中の常駐プログラムを検索して情報を取得します．取得できなかった場合（常駐プログラムが起動していない場合を含む）はエラーを返します． 常駐プログラムが起動している PC に複数枚の有効な NIC が存在する場合は，その NIC 分の常駐プログラムが重複検出されます．

開発環境	書　式
VC++	DWORD WINAPI　ed864_EtnSearch(BYTE* info, WORD* max, WORD tmo);
VC#	[DllImport("*libname* ")] public static extern uint　ed864_EtnSearch (ref byte info, ref ushort max, ushort tmo);

引　数	説　明
Info	常駐プログラム情報の格納ｱﾄﾞﾚｽ．情報データは 8Byte
max	ﾊﾞｯﾌｧに格納する情報数の最大値と実際に格納した情報数． ﾊﾞｯﾌｧは情報数最大値×8Byte 以上の大きさが必要
tmo	応答待ち時間（注意：指定した時間待ちます）

VC++ 記述例	DWORD ret;　　//関数の戻り値 BYTE info[32];　　//プログラム情報の格納領域(最大 4 情報分のサイズ) ret = ed864_ GetProgInfo　(info, 4, 3000);　//取得ﾀｲﾑｱｳﾄ（=待ち時間 3 秒），情報格納最大 4
備考	ﾃﾞｰﾀﾌｫｰﾏｯﾄ(1 情報分) +0～+3 IP アドレス（例 192.168.1.2 の場合+0:192 +1:168 +2:1 +3:2 が格納される） +4 接続デバイス数 +5 未使用 +6,+7 ポート番号（+6 が下位側） ※同一 PC 上の複数 NIC の情報を取得した場合，IP アドレス以外の情報は同じ内容になります． また，この場合はどの IP アドレスに接続しても目的のデバイスに接続が可能です

7.6.21 ed864_EtnSend() Ethernet 送信

機　能	デバイスハンドルで指定された DIO864 へ Ethernet または WiFi 経由でデータを送信します

開発環境	書　式
VC++	DWORD WINAPI　ed864_EtnSend (short sType, short sBid, BYTE* data, WORD* leng);
VC#	[DllImport("*libname*")] public static extern uint　ed864_ EtnSend (short sType, short sBid, ref byte data, ref ushort leng);

引　数	説　明
sType	ボードタイプ (0x12:HETN-DIO864T / 0x13:HWIF-DIO864W
sBid	ボード ID (0～15)
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 = ed864_EtnSend (0x12, 14, 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.Ethernet/WiFi 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、ed864_EtnRecv()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. .

7.6.22 ed864_EtnRecv() Ethernet 受信

機　能	デバイスハンドルで指定された DIO864 から Ethernet または WiFi 経由でデータを受信します

開発環境	書　式
VC++	DWORD WINAPI　ed864_ EtnRecv (short sType, short sBid, BYTE* data, WORD* leng);
VC#	[DllImport("*libname*")] public static extern uint　ed864_EtnRecv (short sType, short sBid, ref byte data, ref ushort leng);

引　数	説　明
sType	ボードタイプ (0x12:HETN-DIO864T / 0x13:HWIF-DIO864W
sBid	ボード ID (0～15)
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 = ed864_EtnSend (0x12, 14, sdata, &sleng);　//DI 読出コマンド送信 if(ret != 0) return; ret = ed864_ EtnRecv (0x12, 14, 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 Ethernet/WiFi 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.

7.6.23 ed864_GetOpenDevCount ()　　デバイスオープン済み接続数の取得

機　能	ボードタイプとボード ID で指定されたボードのオープン済みソケット接続数を取得します.

開発環境	書　式
VC++	DWORD WINAPI　ed864_GetOpenDevCount(Short sType, short sBid, short* count);
VC#	[DllImport("*libname* ")] public static extern uint　ed864_GetOpenDevCount (short sType, short sBid, ref short count);

引　数	
sType	ボードタイプ (0x12:HETN-DIO864T / 0x13:HWIF-DIO864W
sBid	ボード ID (0～15)
count	デバイスオープンで既にソケット接続している接続数

VC++ 記述例	DWORD ret;　//関数の戻り値 Short count; ret = ed864_GetOpenDevCount(0x12, 14, &count);　//HETN-DIO864, BID=14 のオープン済み // 接続数の取得
備　考	Ethernet による複数プロセスまたはスレッド, 別 PC からの同一デバイスへの接続は可能ですが、後からデバイスオープンしたプログラムで不用意な処理（軸操作や初期化など）を行うと異常な動作になることが考えられます. また、同一ボードに対しては DLL 内で排他処理が行われるため, 排他関連エラーが発生した場合はこれに対応した処理を施す必要があります. これら点にご注意していただいたうえでご使用ください.

7.7ライブラリ関数詳細

7.7.1 Hed864_ ProgLinkSet()　　常駐プログラムのリンク設定

機　能	常駐プログラムに接続するための情報を設定します．設定は接続先 IP，ポートを検索して自動設定することも，指定した値を設定することも可能です．

開発環境	書　式
VC++	DWORD hed864_ProgLinkSet(short sMode, BYTE *sIp, WORD usPort);
VC#	[DllImport("*libname* ")]public static uint hed864_ProgLinkSet(short sMode, string sIp, ushort usPort);

引　数	説　明
sMode	リンク設定方法（0:自動検出 1:指定アドレス接続 2:自ローカルアドレス接続）
sIp	指定する IP アドレスで sMode によって内容が変わります．sMode が 0 の時は自動検索ですが優先して検索するネットワークアドレスを指定（***.***.***.0 の書式），1 の時は指定したアドレスを設定，2 の時は無効（127.0.0.1 を内部で設定）します
usPiort	指定するポート番号で sMode によって内容が変わります．sMode が 0 の時は無効，1 の時は指定したポート番号を設定，2 の時は usPort に 0 以外が設定されている場合はその値，0 の時は常駐プログラムを検索して，存在する場合にはそのポート番号，存在しない場合は“10010”固定で設定します．

VC++
記述例
	DWORD ret;　　//関数の戻り値
	//IP=192.168.2.1 の PC 上で RUN している常駐プログラムに待受けポート番号 10010 で接続する設定をします
	ret = hed864_ProgLink(1, “192.168.2.1”, 10010);

7.7.2 hed864_GetProgVersion()　　常駐プログラムのソフトウェアバージョン取得

機　能	常駐プログラムおよび常駐プログラムのアクセス関数の各バージョンを取得します

開発環境	書　式
VC++	DWORD hed864_GetProgVersion(WORD* pver, WORD* lver);
VC#	[DllImport("*libname* ")] public static extern uint hed864_ GetProgVersion (ref ushort pver, ref ushort lver);

引　数	説　明
pver	常駐プログラムのバージョン
lver	常駐プログラムアクセスライブラリーのバージョン

VC++ 記述例	DWORD ret;　　//関数の戻り値 WORD* pver;　　//常駐プログラムのバージョン番号格納アドレス WORD* lver;　　//常駐プログラムアクセスライブラリーのバージョン番号格納アドレス ret = hed864_GetSftVersion　(pver, lver); ※各バージョン情報は ver1.234 の場合 0x1234 と格納されます

7.7.3 hed864_GetCmdError()　　コマンド実行エラーの取得

機　能	コマンド実行時に発生したエラー情報の取得を行います．エラーの発生があった場合は自動的にエラーをクリアします． エラーは前回取得してから今回取得するまでに発生したエラー情報の累積になります． 不要なエラー項目はマスクしてエラー判定してください．

開発環境	書　式
VC++	DWORD WINAPI hed864_GetError(Short sType, short sBid, DWORD* err);
VC#	[DllImport("*libname* ")] public static extern uint hed864_ GetError (short sType, short sBid, ref uint err);

引　数	説　明
sType	ボードタイプ (0x12:HETN-DIO864T / 0x13:HWIF-DIO864W
sBid	ボード ID (0～15)
Err	エラーコード（ビット対応）

VC++ 記述例	DWORD ret;　　//関数の戻り値 DWORD err;　　//エラー情報の格納先 ret = hed864_ GetError (4、14, &err);　　// HETN-DIO864T，BID=14 のコマンド実行エラー情報 　　// の取得とエラークリア

備　考

エラー内容・・・0 でエラーなし

●コマンド実行エラー

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
---	---	---	---	データ フォーマット エラー	データ範囲 エラー	コマンド 受信不可	未定義 コマンド
Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8
汎用 RS-232C-2				汎用 RS-232C-1			
送信タイム アウト	送信 未完了	受信タイム アウト	通信エラー	送信タイム アウト	送信 未完了	受信タイム アウト	通信エラー
Bit23	Bit22	Bit21	Bit20	Bit19	Bit18	bit17	Bit16
システムで使用							
Bit31	Bit30	Bit29	Bit28	Bit27	Bit26	Bit25	Bit24
その他 エラー	---	---	---	---	他処理 アクセス中	アクセス 拒否	受信バッフ ァオーバー 風呂

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　　【 sed86400.vcproj 】
(2) Visual C# 用コード　　【 sed86404.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) DIO864 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 常駐プログラムの起動

Ethernet/WiFi インターフェース搭載の DIO864 を Ethernet/WiFi を使って使用する場合は，まず「常駐プログラム」を起動する必要があります．このプログラムは各アプリケーションから DIO864 に送られる Ethernet パケットのスイッチング（=交通整理)をする役割をして複数スレッドまたは複数プロセスからのアクセスを可能にしています．（常駐プログラムは DIO864 以外に弊社 Ethernet/WiFi 製品全てに対して共通で使用します）

常駐プログラムはアプリケーションを実行する PC と同一 PC 上で動かすことも，アプリケーションを実行する同一ネットワーク上の他 PC 上で動かすことも可能です．ただし CPD が接続しているネットワーク上にある１台の PC 上でのみ起動が可能です．

常駐プログラムが管理するネットワーク内にある他 PC 上で起動した場合，DIO864 が，複数起動した常駐プログラム間で取り合いとなり正常な動作が行えない場合がありますのでご注意ください．

（PC 同士がネットワークで接続されていても、別ネットワーク上にそれぞれ制御・管理したい DIO864 がある場合は，それぞれの PC 上で常駐プログラムを起動しても問題ありません）

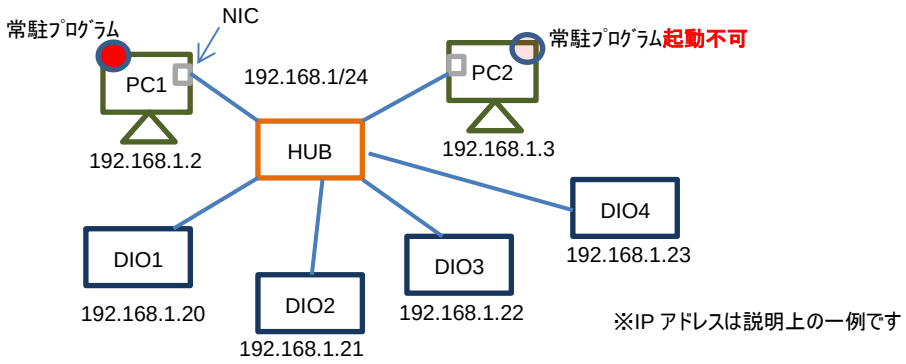


図 8.1-1 接続ケース1（常駐ﾌﾟﾛｸﾞﾗﾑ複数起動が不可能なケース）

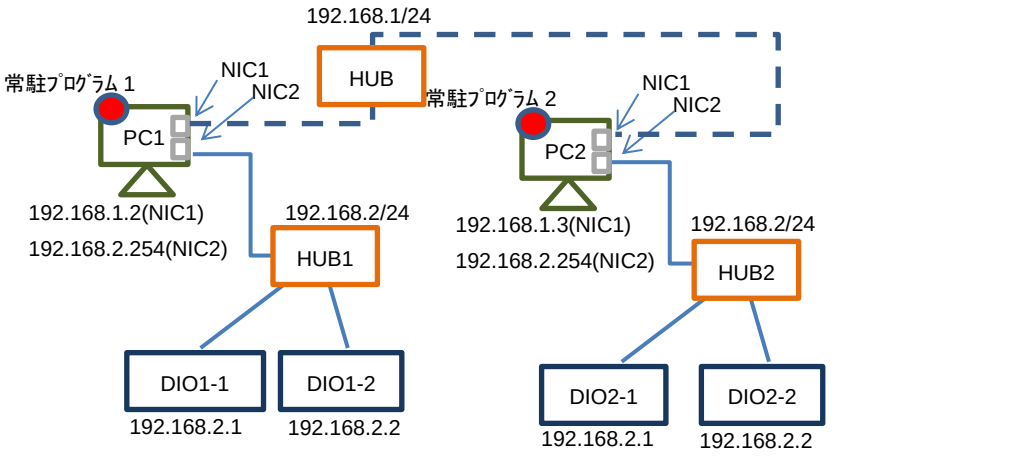
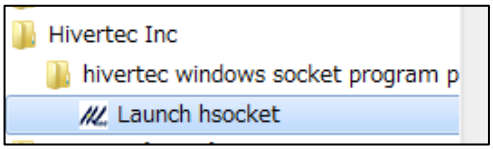


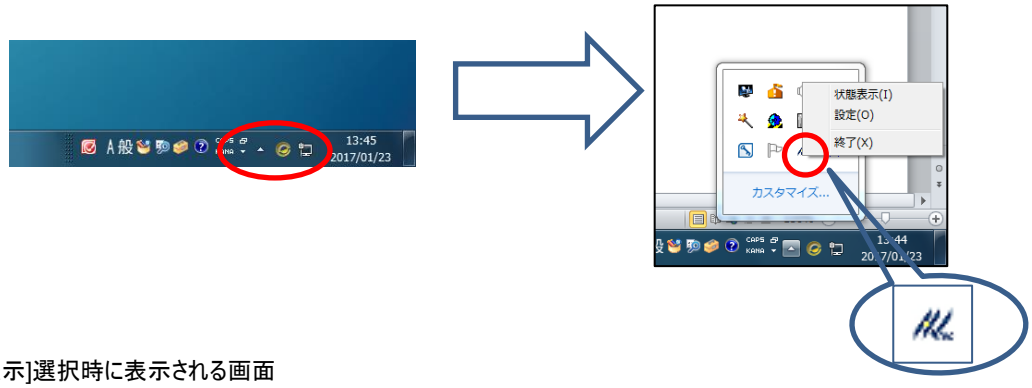
図 8.1-2 接続ケース2（常駐ﾌﾟﾛｸﾞﾗﾑ複数起動が可能なケース）

● 起動手順

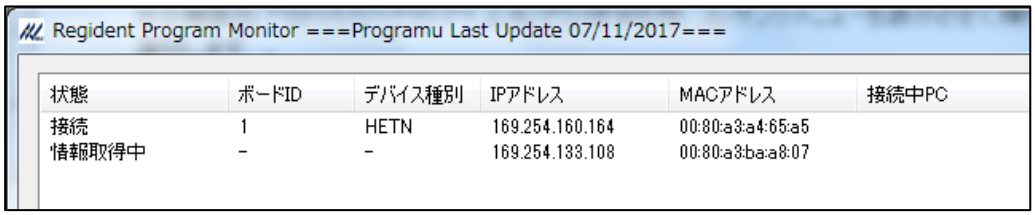
常駐プログラムを Windows スタートメニューの“Hivertec”– “Launch hsocket”を起動します.
常駐プログラムは同一ネットワーク上に接続する1台の PC 上で実行可能です. 1 台に PC 上で複数起動はできません



常駐プログラムは起動すると, Windows 画面右下にアイコンとして表示されます. 設定や終了, 接続状態の確認を行う場合は画面右下のハイバーテックアイコンにマウスを合わせ, コンテンツメニューを表示させて操作(状態表示, 設定, 終了)を選択します.



[表示]選択時に表示される画面



(1) 状態・・・DIO864 を含む弊社 Ethernet/WiFi 製品が検出され, 正常なアライブ応答がある場合は“接続”, アライブ応答が無い場合またはエラーが発生した場合は“情報取得中”, “切断中”を繰り返し表示します. 電源や接続が切れて CPD の検出自体ができない場合は, リストから削除され表示されなくなります.
(2) 接続中 PC・・・アプリケーションからオープン処理をされた時に接続 PC の IP アドレスを表示
(3) その他項目・・・検出したデバイスの情報です. 取得できない情報は「---」を表示します

● 注意事項

◆ 常駐プログラムを起動したまま, 使用しているネットワークアダプター(NIC)を途中で差し替えないでください。接続しているデバイスの検出が正常にできなくなる場合があります. この場合は常駐プログラムを一度終了させてから差し替えを行い, 再度起動してください.

◆ DIO を「DHCP による自動取得」で使用する場合, ネットワーク内に DHCP サーバーが無い場合 AutoIP により自動割り当てされます. この時, NIC 自体が AutoIP による自動割り当てがされないと正常に接続できません. この場合, Windows の AutoIP による自動割り当て設定が有効であることを確認し, 有効でない場合は DIO を固定アドレスで設定するか, または Windows の設定を有効にしてください.
HKEY_LOCAL_MACHINE¥SYSTEM¥CurrentControlSet¥services¥Tcpip¥Parameters
Key 名・・IPAutoconfigurationEnabled

◆ 1つの常駐プログラムが検出可能な DIO がネットワーク上に複数存在する場合, BID の設定はデバイスタイプ毎にユニークになるよう設定してください. 重複すると以下のバルーンが表示されます. BID を変更する場合は CPD の電源を切ってから行ってください

8.2 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（sed86400.exe または sed86404.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，ボード上の動作，操作，開始，終了を次の手順で行います．

① Get Device Info.（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“Get Device Info.”ボタンを押下します

② ボードの選択（2 枚以上の場合）

・・・“Board Type&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．

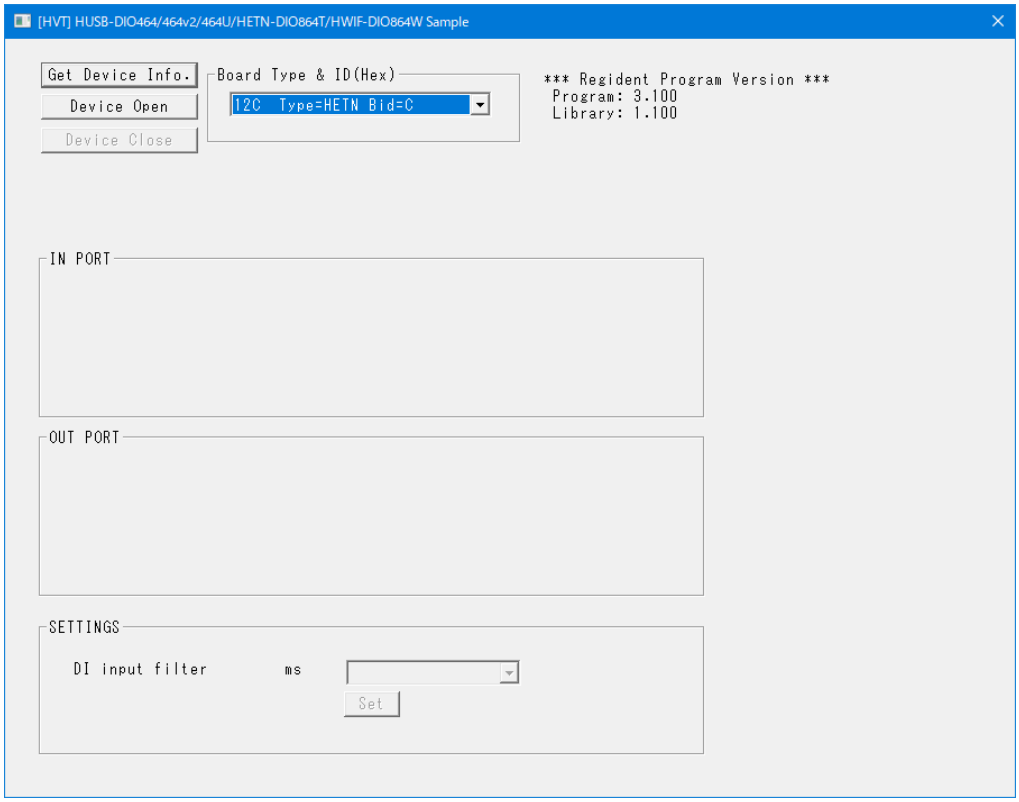


図 8-1.1 サンプルプログラム起動画面

(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．

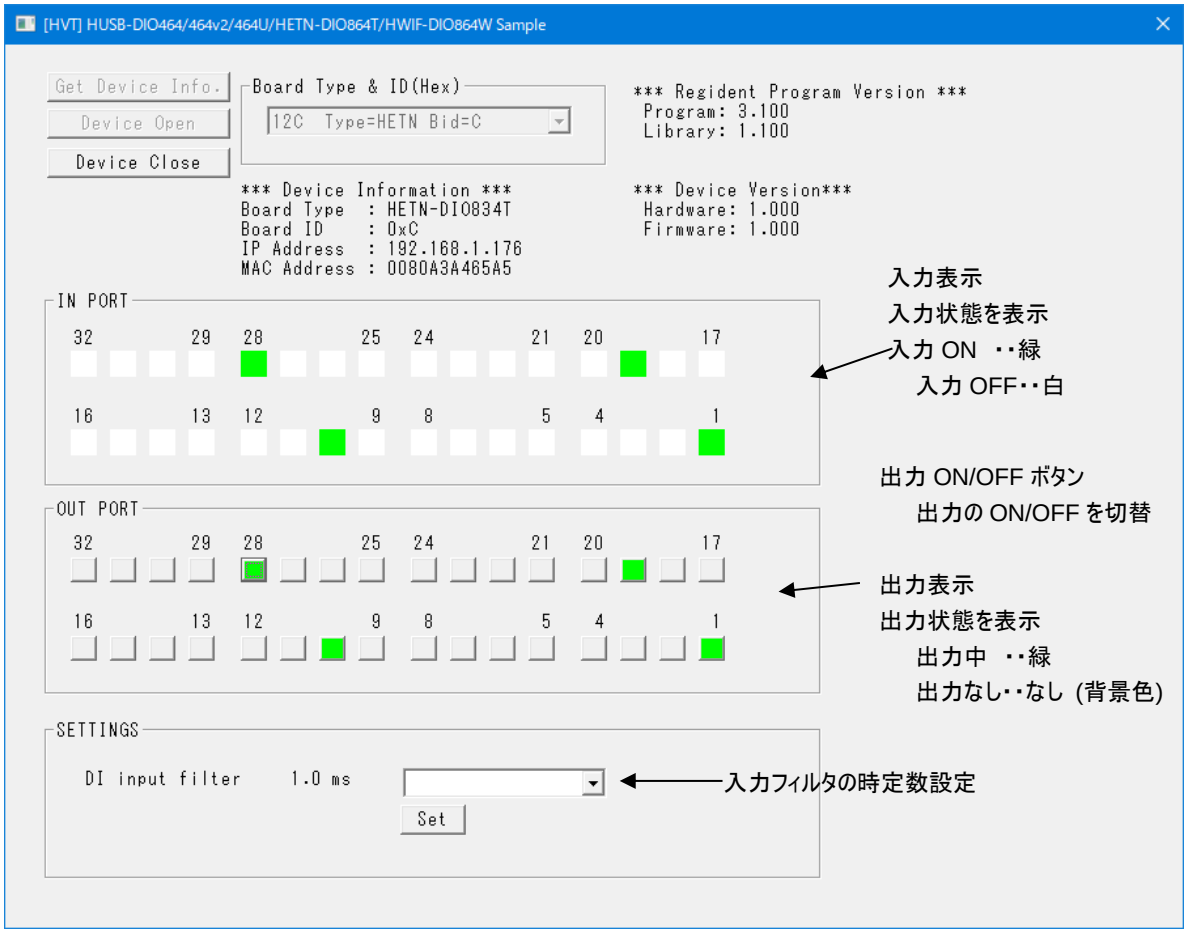


図 8-2.2　サンプルプログラムデバイスオープン後表示画面

9. Ethernet/WiFi コマンド資料

Ethernet 送信関数および Ethernet 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 Ethernet/WiFi 送受信関数を使う場合のこれらコマンドは USB インターフェースを使用した時のコマンドと比べ，送信および受信データにヘッダデータ 8Byte が共に付加されたフォーマットとなりますのでご注意ください．

9.1 Ethernet/WiFi 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	14 8	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	10 12	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	18 8	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	10 12	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	18 16	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	14 8	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	10 8	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	14 8	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	10 12	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	14 8	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	10 12	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	10 8	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	10 10	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	14 8	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	10 12	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	10 8	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	10 10	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 Ethernet/WiFi コマンド一覧

9.2　各コマンド共通ヘッダ部フォーマット

各コマンドの送信データ及び受信データの先頭 8Byte には以下のデータが全てに付加されます.

- 送信データ（コマンド）ヘッダ部

+0	+1	+2	+3	+4	+5	+6	+7
宛先デバイス ボード ID	0	コマンド 本体サイズ(Byte)		ステータス		0	0
		Bit7～0	Bit15～8	Bit7～0	Bit15～8		

+8	+9	+10	+11	+12	～
コマンド本体（2～）					

※コマンド本体サイズはヘッダ部 8Byte を除いたサイズになります.

- 受信データ（応答データ）ヘッダ部

+0	+1	+2	+3	+4	+5	+6	+7
応答デバイス ボード ID	コマンド リダイレクト	応答データ 本体サイズ(Byte)		システム使用		0	0
		Bit7～0	Bit15～8	Bit7～0	Bit15～8		

+8	+9	+10	+11	+12	～
応答データ本体（0～）					

※応答データ本体サイズはヘッダ部 8Byte を除いたサイズになります.

9.3 デバイスアクセスコマンドフォーマット

9.3.1 DO ポート書込指令

- コマンドデータ（送信）［14 Byte］

+8	+9	+10	+11	+12	+13
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）［8 Byte］ 応答データ本体はなし

9.3.2 DO ポート読出指令

- コマンドデータ（送信）［10 Byte］

+8	+9
コマンド (91h)	0

- 応答データ（受信）［12 Byte ］

+8	+9	+10	+11
DO ポート読み出し値			
Bit7～0	Bit15～8	Bit23～16	Bit31～24

9.3.3 DO ポートビット書込指令

- コマンドデータ（送信）［18 Byte］

+8	+9	+10	+11	+12	+13
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

+14	+15	+16	+17
DO ポート書き込み値			
Bit7～0	Bit15～8	Bit23～16	Bit31～24

- 応答データ（受信）［8 Byte］ 応答データ本体はなし

9.3.4 DI ポート読出指令

● コマンドデータ（送信）［10 Byte］

+8	+9
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）［12 Byte ］

+8	+9	+10	+11
DIポート読み出し値			
Bit7～0	Bit15～8	Bit23～16	Bit31～24

9.3.5 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）［18 Byte］

+8	+9	+10	+11	+12	+13
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

+14	+15	+16	+17
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）［16 Byte ］

+8	+9	+10	+11	+12	+13	+14	+15
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

9.3.6　DI ラッチデータクリア指令

- コマンドデータ（送信）［14 Byte］

+8	+9	+10	+11	+12	+13
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）［8 Byte］ 応答データ本体はなし

9.3.7　DO 同期出力データクリア指令

- コマンドデータ（送信）［10 Byte］

+8	+9
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）［8 Byte］ 応答データ本体はなし

9.4 動作設定コマンドフォーマット

9.4.1 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）［14 Byte］

+8	+9
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

+10	+11	+12	+13
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）［8 Byte］ 応答データ本体はなし

9.4.2 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）［10 Byte］

+8	+9
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）［12 Byte］し

+8	+9	+10	+11
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

9.4.3　入力フィルター設定書込指令

- コマンドデータ（送信）［14 Byte］

+8	+9	+10	+11	+12	+13
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）［8 Byte］ 応答データ本体はなし

9.4.4　入力フィルター設定読出指令

- コマンドデータ（送信）［10 Byte］

+8	+9
コマンド (A3h)	0

- 応答データ（受信）［12 Byte］し

+8	+9	+10	+11
フィルター設定読み出し値			
Bit7～0	Bit15～8	Bit23～16	Bit31～ 24

9.4.5　同期入力信号設定書込指令

- コマンドデータ（送信）［10 Byte］

+8	+9
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）［8 Byte］ 応答データ本体はなし

9.4.6　同期入力信号設定読出指令

- コマンドデータ（送信）［10 Byte］

+8	+9
コマンド (A5h)	0

- 応答データ（受信）［10 Byte］し

+8	+9
同期入力 信号設定 読み出し値	予約 (0)

9.4.7　出力データ選択設定書込指令

- コマンドデータ（送信）［14 Byte］

+8	+9
コマンド (A6h)	0

+10	+11	+12	+13
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）［8 Byte］ 応答データ本体はなし

9.4.8　出力データ選択設定読出指令

- コマンドデータ（送信）［10 Byte］

+8	+9
コマンド (A7h)	0

- 応答データ（受信）［12 Byte］し

+8	+9	+10	+11
出力データ選択設定読み出し値			
Bit7～0	Bit15～8	Bit23～16	Bit31～ 24

9.4.9　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）［10Byte］

+8	+9
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）［8 Byte］ 応答データ本体はなし

9.4.10 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）［10 Byte］

+8	+9
コマンド (A9h)	0

- 応答データ（受信）［10 Byte］し

+8	+9
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
2019/04/25	1.00	新規作成

表 10-1.1 更新履歴