デジタル入出力ボード/USB製品用
Windowsドライバ『API-DIO(WDM)』

従来ドライバ『API-DIO(98/PC)』
からの移行ガイド

株式会社コンテック

www.contec.com

1. はじめに

本書は従来ドライバ(API-DIO(98/PC))から新ドライバ(API-DIO(WDM))への移行をスムーズに行うための資料です。

以降本書では、各ドライバを簡略化して下記で表記いたします。
・従来ドライバ(API-DIO(98/PC)) → 98/PCドライバ
・新ドライバ(API-DIO(WDM)) → WDMドライバ

■ WDMドライバについて
98/PCドライバに対して「より使いやすく便利に」を目指したのがWDMドライバです。
お客様に当社製品をお使いいただくにあたってはWDMドライバの使用をお勧めします。
WDMドライバでは、新規OSや新規デバイスへの対応は行いますが、98/PCドライバでは対応できない場合があります。

✓ 対応OS （2019年2月現在）
WDMドライバ

98/PCドライバ

✓ 対応デバイス製品（2019年2月現在）
WDMドライバ
PCI Expressボード、PCIボード、PCカード(CardBus)、
USBデバイス、Ethernetデバイス、Wirelessデバイス

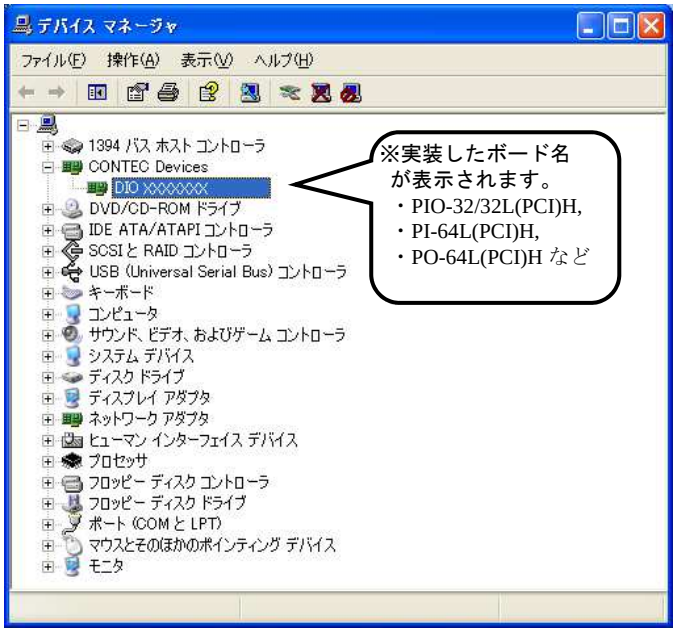
98/PCドライバ
PCI Expressボード、PCIボード、PCカード(PCMCIA、CardBus)、
ISAボード

なお、同じパソコン上で98/PCドライバとWDMドライバを同時に使用することはできません。
使用するドライバソフトを変更するには、それまで使用していたドライバソフトをアンインストールする必要があります。

2. 98/PCドライバとWDMドライバの相違点

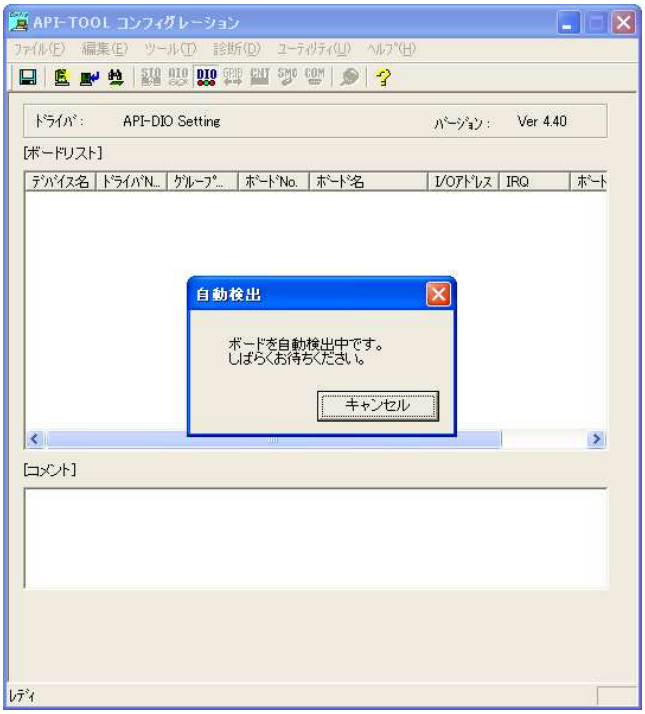
■ デバイスマネージャでの扱いの違い
98/PCドライバでは「多機能アダプタ」に分類されていましたが、WDMドライバでは
「CONTEC Devices」という新規カテゴリに分類されます。

《98/PCドライバの場合》

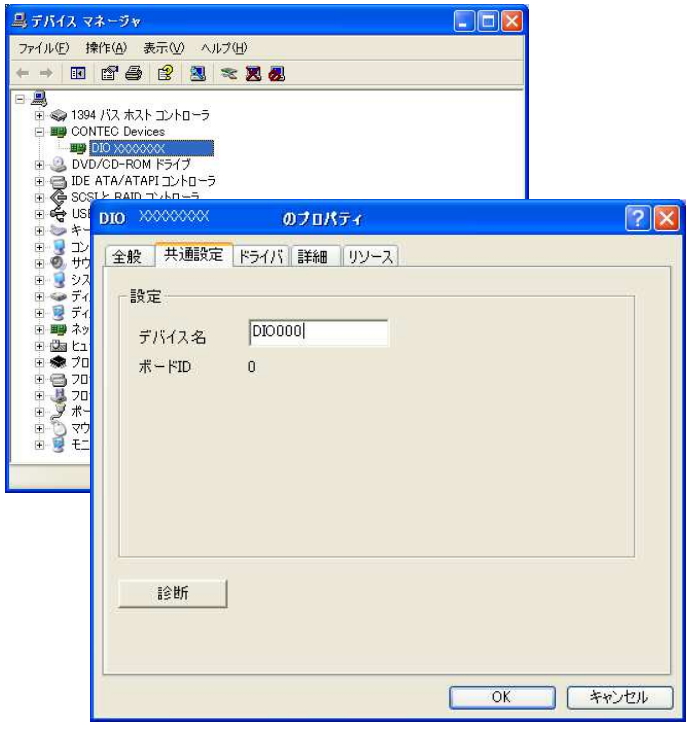


《WDMドライバの場合》

■ ソフトウェアの初期設定の違い
98/PCドライバでは、「API-TOOLコンフィグレーション」にて使用するデバイスを登録し、
グループごとにデバイス名を設定していました。WDMドライバでは、デバイスマネージャから
デバイスを選択し、「プロパティ」の「共通設定」タブでデバイスごとにデバイス名を設定します。



《98/PCドライバの場合》



《WDMドライバの場合》

■ API関数の違い (Visual Basic の例で紹介しますが、他の言語でも同様です)

【初期化関数】
98/PCドライバではグループに複数デバイスを登録することが可能でしたが、考え方が
複雑になってしまう側面があり、WDMドライバではデバイスごとに管理する形をとっています。

項目	API-DIO(98/PC)ドライバ	API-DIO(WDM)ドライバ
関数名	DioOpen DioOpenEx	DioInit
関数の種類	グループ番号を指定する方法と、デバイス名を指定する方法の２種類があります。	1種類のみで、デバイス名を指定する方法となります。
指定できるデバイス数	API-TOOLで1つのグループに複数のデバイスを登録することで、最大4つのデバイスを1度に扱うことが可能です。	1度に扱えるのは1つのデバイスまでです。
初期化関数後のデバイス指定方法	初期化関数実行後の引数に入ってくる情報を、デバイスハンドルとして使用します。	初期化関数実行後の引数に入ってくる情報を、デバイスIDとして使用します。

(例)98/PCドライバ

```
DrvNo = 0                       ‘ ドライバ番号に入出力固定ボードを指定
GrpNo = 1                       ‘ グループ番号を指定
Ret = DioOpen ( hDrv, DrvNo, GrpNo )
‘ もしくは
DeviceName = “DIO000”       ‘ デバイス名を指定
Ret = DioOpenEx ( DeviceName, hDrv )
```

(例)WDMドライバ

```
DeviceName = “DIO000”       ‘ デバイス名を指定
Ret = DioInit ( DeviceName , Id )
```

【終了関数】
関数名および引数に違いがありますが、機能的には相違ありません。

項目	API-DIO(98/PC)ドライバ	API-DIO(WDM)ドライバ
関数名	DioClose	DioExit
引数	デバイスハンドルで指定します。	デバイスIDで指定します。

(例)98/PCドライバ

```
Ret = DioClose ( hDrv )
```

(例)WDMドライバ

```
Ret = DioExit ( Id )
```

【デジタル入出力関数】（簡易入出力関数）
引数に違いがありますが、関数名および機能的には相違ありません。

項目	API-DIO(98/PC)ドライバ	API-DIO(WDM)ドライバ
関数名	DioInpByte DioOutByte DioEchoBackByte DioInpBit DioOutBit DioEchoBackBit	DioInpByte DioOutByte DioEchoBackByte DioInpBit DioOutBit DioEchoBackBit
引数	デバイスハンドルで指定します。	デバイスIDで指定します。

(例)98/PCドライバ

```
PortNo = 0                                  ‘ ポートNo.0を設定
Ret = DioInpByte ( hDrv , PortNo , Data )   ‘ ポートNo.0から入力
```

(例)WDMドライバ

```
PortNo = 0                                  ‘ ポートNo.0を設定
Ret = DioInpByte ( Id , PortNo , Data )     ‘ ポートNo.0から入力
```

【デジタル入出力関数】（複数バイト(ビット)入出力関数）
関数名および引数に違いがありますが、機能的には相違ありません。

項目	API-DIO(98/PC)ドライバ	API-DIO(WDM)ドライバ
関数名	DioInp DioOut DioEchoBack DioBitInp DioBitOut DioBitEchoBack	DioInpMultiByte DioOutMultiByte DioEchoBackMultiByte DioInpMultiBit DioOutMultiBit DioEchoBackMultiBit
引数	デバイスハンドルで指定します。 入出力ポートなどの情報を構造体に まとめて指定します。	デバイスIDで指定します。 入出力ポートなどの情報を各引数で 指定します。

(例)98/PCドライバ

```
    lpDInp.PortNum = 2              ‘ ポート数2を設定
    PortNo(0) = 0                   ‘ ポートNo.0を設定
    PortNo(1) = 1                   ‘ ポートNo.1を設定

    ‘引数(構造体)に変数アドレスを渡す
    lpDInp.InpPortNo = LpWord(PortNo(0))
    lpDInp.Buf = LpByte(Data(0))

    ‘デジタル入力関数実行
    Ret = DioInp ( hDrv, lpDInp ) ‘ ポートNo.0,1から入力
```

(例)WDMドライバ

```
    PortNum = 2                     ‘ ポート数2を設定
    PortNo(0) = 0                   ‘ ポートNo.0を設定
    PortNo(1) = 1                   ‘ ポートNo.1を設定

    Ret = DioInpMultiByte ( Id , PortNo , PortNum , Data ) ‘ ポートNo.0,1から入力
```

【デジタル入出力関数】（バックグラウンド入出力関数）
WDMドライバでは実装されていません。

項目	API-DIO(98/PC)ドライバ	API-DIO(WDM)ドライバ
関数名	DioInpBack DioOutBack DioBitInpBack DioBitOutBack DioSts	×（該当する関数はありません）

WDMドライバでバックグラウンド入出力を行いたいときは、Windowsのシステムタイマや
当社製タイマドライバ(API-TIMER(WDM))をお使いください。

【割り込み機能関数】
WDMドライバでは割り込み機能をシンプルに集約しています。

項目	API-DIO(98/PC)ドライバ	API-DIO(WDM)ドライバ
関数名	DioEvent DioIntEnable DioIntSence DioEventEx DioIntEnableEx	DioNotifyInterrupt
引数・機能	デバイスハンドルで指定します。 割り込み要因設定と禁止/許可を別関数で用意しています。 割り込み発生イベントはウィンドウハンドルの指定で受け取りが可能です。	デバイスIDで指定します。 割り込み要因設定と禁止/許可をDioNotifyInterrupt関数に集約。 割り込み発生イベントはウィンドウハンドルが指定できるほか、コールバック関数で受け取りが可能です。

【デジタルフィルタ関数】
関数名および引数に違いがありますが、機能的には相違ありません。

項目	API-DIO(98/PC)ドライバ	API-DIO(WDM)ドライバ
関数名	DioSFilter	DioSetDigitalFilter
引数	デバイスハンドルで指定します。	デバイスIDで指定します。 論理ビットの引数を省略しています。

【8255搭載ボード専用関数】
関数名に違いがありますが、機能的には相違ありません。

項目	API-DIO(98/PC)ドライバ	API-DIO(WDM)ドライバ
関数名	Dio8255Mode DioGet8255Mode	DioSet8255Mode DioGet8255Mode
引数	デバイスハンドルで指定します。	デバイスIDで指定します。

【プロセスコントロール機能関数】
WDMドライバでは実装されていません。

項目	API-DIO(98/PC)ドライバ	API-DIO(WDM)ドライバ
関数名	DioPtnSet DioPtnStart DioPtnSts	×（該当する関数はありません）

【トリガ監視機能関数】
WDMドライバではトリガ監視機能をシンプルに集約しています。

項目	API-DIO(98/PC)ドライバ	API-DIO(WDM)ドライバ
関数名	DioTrgSet DioTrgStart DioTrgSts	DioNotifyTrg
引数・機能	デバイスハンドルで指定します。 トリガ監視ビット設定と開始/停止を 別関数で用意しています。 トリガ発生イベントはウィンドウハンドルの 指定で受け取りが可能です。	デバイスIDで指定します。 トリガ監視ビット設定と開始/停止を DioNotifyTrg関数に集約。 トリガ発生イベントはウィンドウハンドルが 指定できるほか、コールバック関数で受け取り が可能です。

【コード変換機能関数】
WDMドライバでは実装されていません。

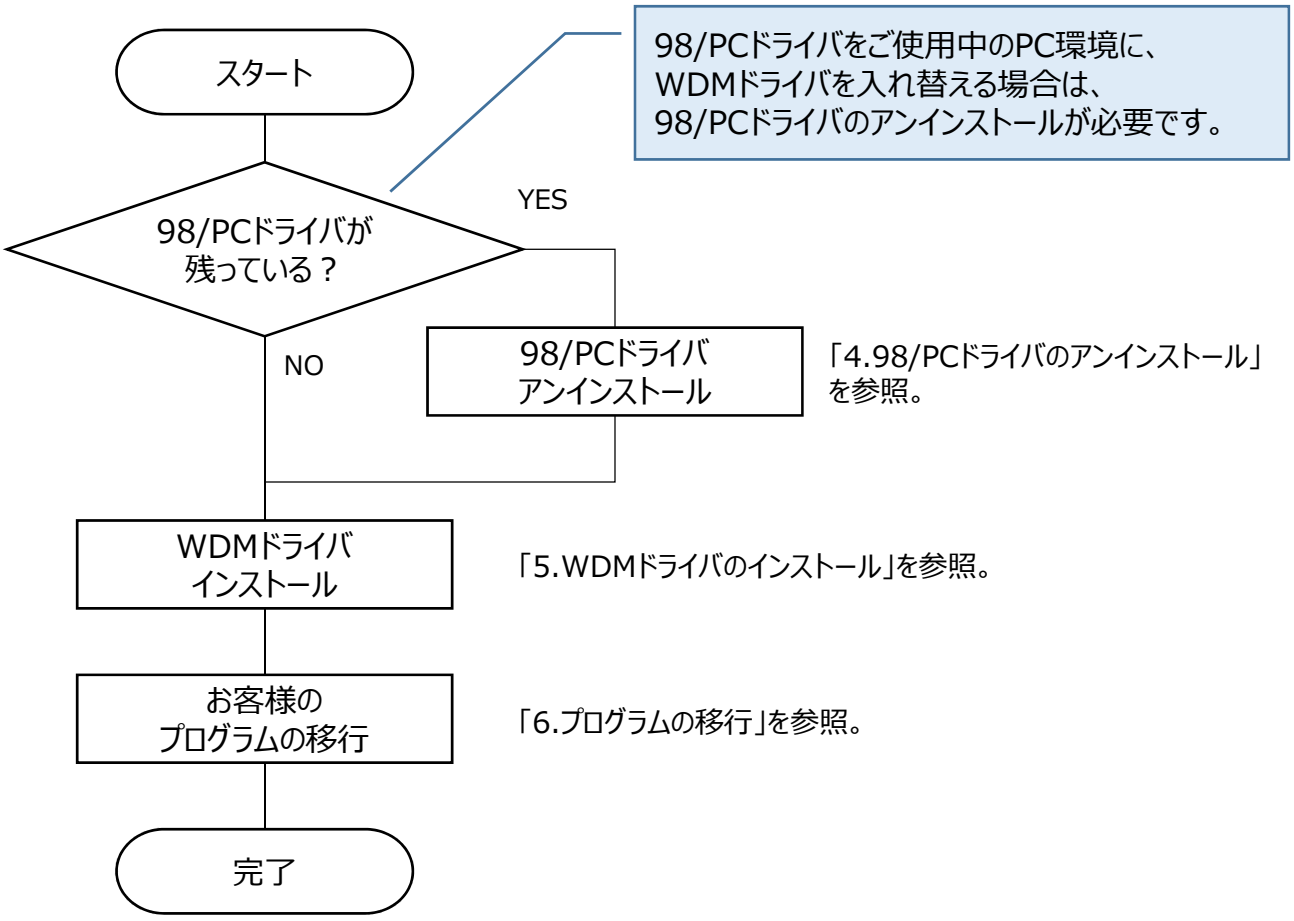
項目	API-DIO(98/PC)ドライバ	API-DIO(WDM)ドライバ
関数名	DioInpBCD DioOutBCD DioNInpBCD DioNOutBCD	×（該当する関数はありません）

【その他関数】
WDMドライバでは実装されていません。

項目	API-DIO(98/PC)ドライバ	API-DIO(WDM)ドライバ
関数名	DioWait DioWaitEx DioUseMutex LpByte LpWord LpDWord	×（該当する関数はありません）

3. ドライバ移行のステップ

■ ドライバ移行のステップ



4. 98/PCドライバのアンインストール

1.デバイスマネージャから登録されているデバイスを削除します。
「多機能アダプタ」(またはマルチファンクションアダプタ)登録のハードウェアを削除します。
※右クリックでポップアップメニューから[削除]を選択、またはDELキーで削除できます。

2.INFファイルを削除します。
INFファイルはWindowsフォルダの以下の場所にコピーが作成されます。
このファイルがあると、一旦デバイスを削除しても、自動的に再度そのドライバが
インストールされてしまいますので、削除します。

¥Windows¥Inf、または¥Winnt¥Inf フォルダ中にある以下のファイルから、
該当するファイルを削除します。
oem*.inf、oem*.pnf （*は任意の数字）

WindowsがINFファイルの名前を変更してコピーしまうため、ファイルの内容を確認して
削除してください。すべてのoem*.*ファイルを削除した場合は、再度ハードウェアの
インストール時にINFファイルが要求されます。(当社のハードウェアだけとは限りません)
ただし、ファイルがoem1.*しかない場合は、それが該当ファイルになるので削除してください。

INFファイルの確認方法
1).メモ帳(NOTEPAD.EXE)等でoem*.infファイルを開く
2).[version]セクションに以下の文字列があるか確認
Class=MultiFunction
provider=%CONTEC%
3).[strings]セクションに使用するハードウェアが記載されていることを確認
4).以上が確認できたらファイルを削除します。

3.Windowsをシャットダウンし、パソコンの電源を切ります。

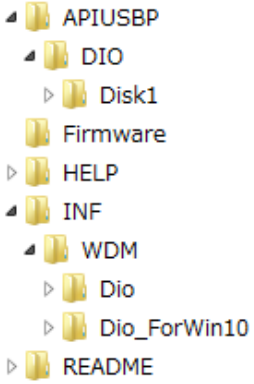
4.パソコンの電源が切れてから、PCI Expressボードなどのハードウェアをパソコンから外します。

5. WDMドライバのインストール

製品によってはドライバソフトを格納したCD/DVD等のメディアが添付していますが、
当社ホームページから最新バージョンをダウンロードして使用されることをお勧めします。
ご使用製品のページから「サポート・ダウンロード」→「ドライバ」にある
“Windows版高機能デジタル入出力ドライバ API-DIO(WDM) 開発環境(フルセット)”
をダウンロードしてください。

下記ではホームページからダウンロードしたファイル(USBデバイス製品用ドライバソフトウェア)をインストールする内容で記載いたします。

1.ダウンロードした圧縮ファイルを展開してください。



← 圧縮ファイル展開後のイメージ

2.開発環境をインストールしてください。
開発環境にはサンプルプログラムや、関数リファレンスなどが書かれたヘルプファイルがあります。
APIUSBP¥DIO¥Disk1¥setup.exe を実行してください。

画面の指示に沿って進めてください。

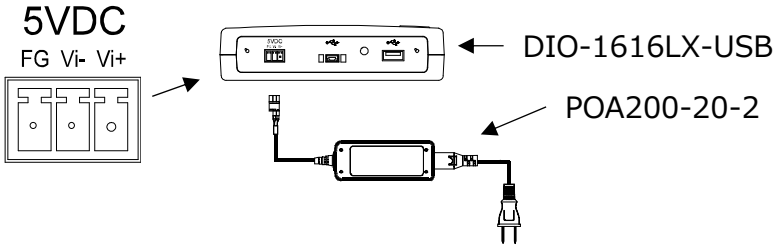
3.デバイスドライバをインストールしてください。
Windows10：INF¥WDM¥Dio_ForWin10¥Setup.exe を実行してください。
その他OS：INF¥WDM¥Dio¥Setup.exe を実行してください。

画面の指示に沿って進めてください。

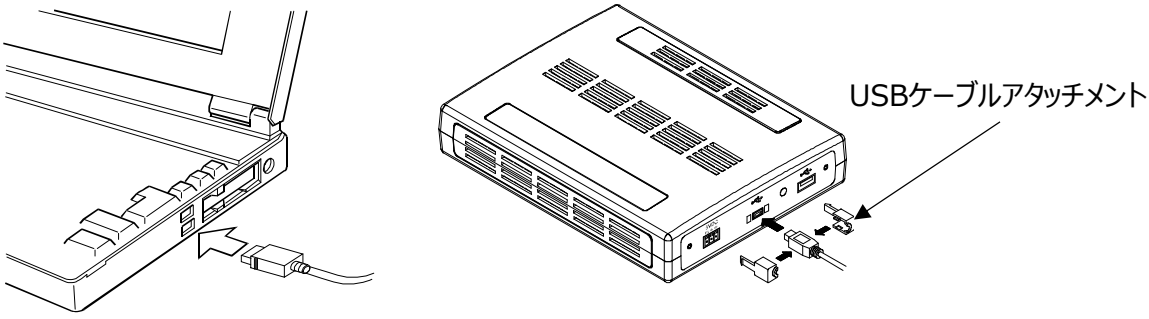
4.OSを起動したままデバイスをプラグインします。
デバイスのプラグインにより、デバイスのインストールは自動的に行われます。

下記はDIO-1616LX-USBを例に説明します。
お使いの製品の解説書をご参照ください。

◆セルフパワー用の5VDC電源との接続
DIO-1616LX-USBをパソコン本体のUSBコネクタに接続する場合はバスパワーで動作可能なので5VDC電源は必要ありません。
ただし当社USBデバイスに搭載しているハブ機能(USB Type A端子)を使用する場合は、5VDC電源を接続して(セルフパワーで)使用する必要があります。
この場合、+5VDC入力端子を使用して5VDC電源と接続します。
オプションのACアダプタ[POA200-20-2]を使用する場合は、入力端子にそのまま接続してください。

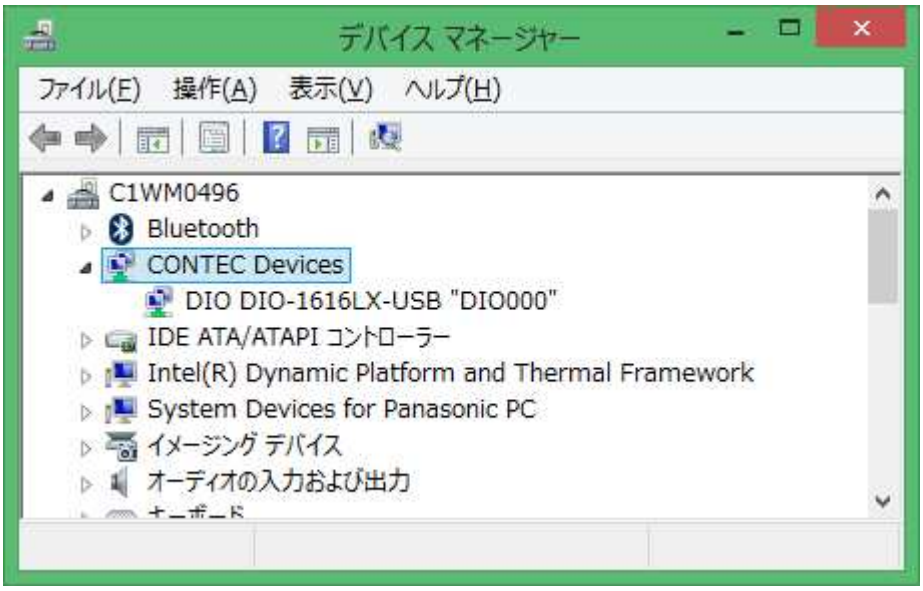


◆製品の接続
DIO-1616LX-USBをパソコンのUSBポートに接続してください。
当社USBデバイスに搭載しているUSBハブを介しての接続も可能です。
USBケーブルアタッチメントを取り付けると本体からUSBケーブルが抜けにくくなります。

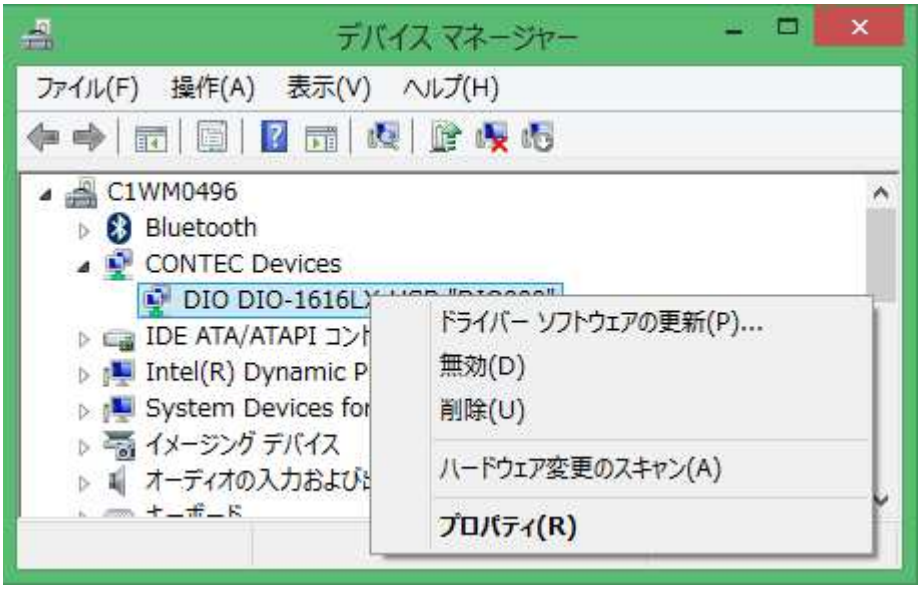


5.デバイス名の確認と変更を行います。
各関数にアクセスするためには、“DIO000”などのデバイス名を使用します。
このデバイス名の確認や変更はデバイスマネージャから行います。
デバイス名の確認はいずれの権限でも可能ですが、デバイス名を変更するには
管理者権限が必要になります。

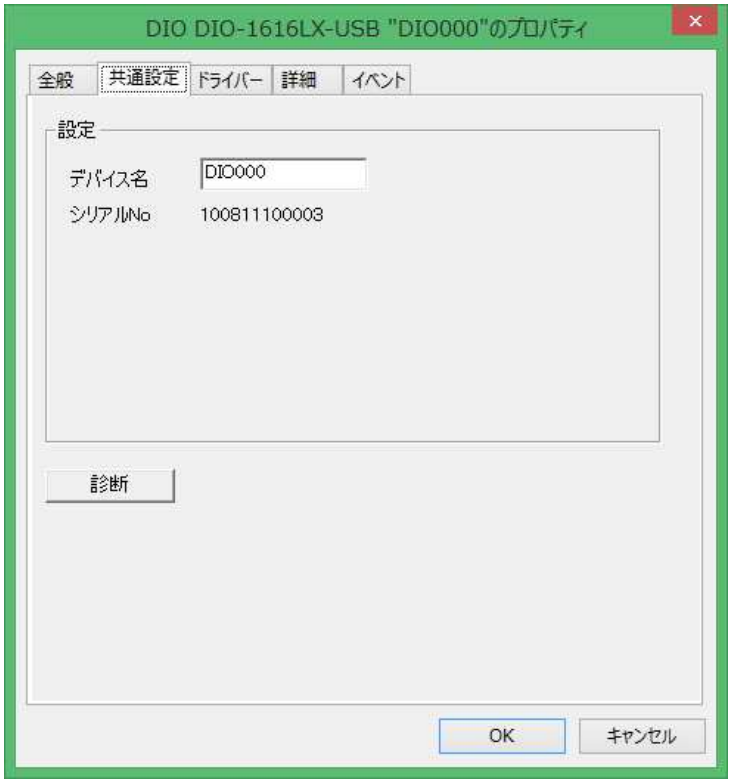
◆デバイス名の確認方法
デバイスマネージャを起動します。
ドライバを使用するハードウェアは全て、CONTEC Devicesツリーの下に登録されています。
デバイスツリーを開くと、デバイスに割り当てられているデバイス名が表示されます。



◆デバイス名の変更方法
デバイスマネージャを起動します。
デバイスツリーを開き、設定するハードウェアを選択して右クリックしてください。
ポップアップメニューから[プロパティ]をクリックします。

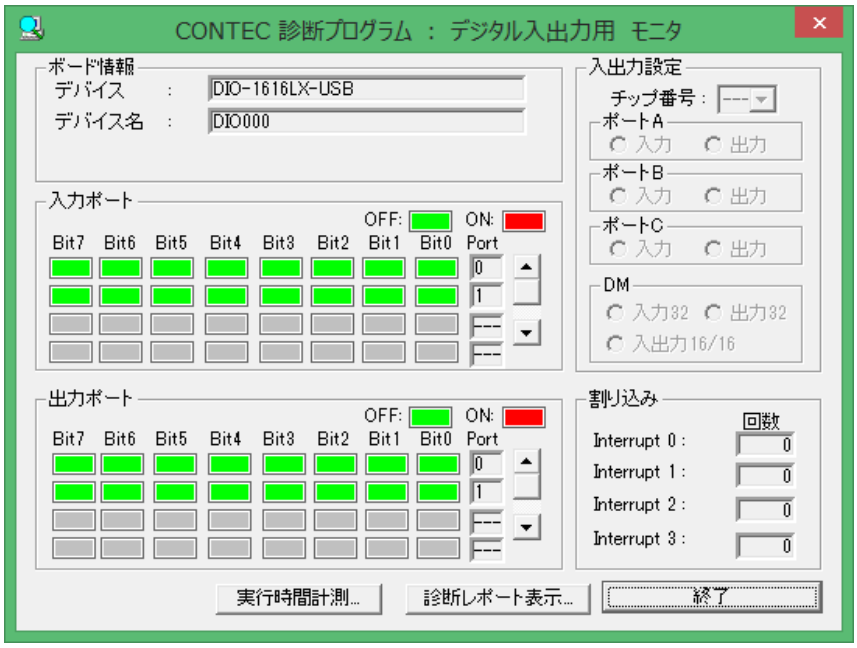


デバイスのプロパティページが表示されます。デバイス名の設定は共通設定タブで行います。
デバイス名を入力して[OK]をクリックしてください。



複数のデバイスを使用する場合、デバイス名は重複しないように設定する必要があります。
デバイス名に使用できる文字列は、半角で最大256文字までです。

◆デバイスの簡易動作確認
診断プログラムを使用して、ハードウェアの簡易的な動作確認を行うことができます。
診断プログラムを起動するには、プロパティページで[診断]ボタンをクリックします。



6. プログラムの移行

98/PCドライバとWDMドライバの関数対比表を下記に記します。

API-DIO(98/PC)ドライバ		API-DIO(WDMドライバ
関数名	機能	関数名
《共通部》		
DioOpen	初期化処理	DioInit
DioOpenEx	初期化処理(デバイス名)	DioInit
DioClose	終了処理	DioExit
DioWait	指定時間の待ち時間処理	×（該当関数無し）　*1
DioWaitEx	指定時間の待ち時間処理（ms単位）	×（該当関数無し）　*1
DioSFilter	指定グループのフィルタ処理	DioSetDIgitalFilter
DioUseMutex	マルチプロセス時の排他処理	×（該当関数無し）　*2
《8255搭載ボード専用関数》		
Dio8255Mode	プログラマブルボードの8255モード設定	DioSet8255Mode
DioGet8255Mode	プログラマブルボードの8255モード設定取得	DioGet8255Mode
《PIO-32D(PM)専用関数》		
DioSetIoDirection	PIO-32D(PM)(Dual)の入出力設定	×（該当関数無し）　*3
DioGetIoDirection	PIO-32D(PM)(Dual)の入出力設定取得	×（該当関数無し）　*3
《基本機能関数》 デジタル入出力関数　（簡易入出力関数）		
DioInpByte	指定ポートのデジタル入力処理	DioInpByte
DioOutByte	指定ポートのデジタル出力処理	DioOutByte
DioEchoBackByte	指定ポートのリードバックデータ入力処理	DioEchoBackByte
DioInpBit	指定ビットのデジタル入力処理	DioInpBit
DioOutBit	指定ビットのデジタル出力処理	DioOutBit
DioEchoBackBit	指定ビットのリードバックデータ入力処理	DioEchoBackBit
《基本機能関数》 デジタル入出力関数　（複数バイト(ビット)入出力関数）		
DioInp	指定ポートのデジタル入力処理（複数バイト）	DioInpMultiByte
DioOut	指定ポートのデジタル出力処理（複数バイト）	DioOutMultiByte
DioEchoBack	指定ポートのリードバックデータ入力処理（複数バイト）	DioEchoBackMultiByte
DioBitInp	指定ビットのデジタル入力処理（複数ビット）	DioInpMultiBit
DioBitOut	指定ビットのデジタル出力処理（複数ビット）	DioOutMultiBit
DioBitEchoBack	指定ビットのリードバックデータ入力処理（複数ビット）	DioEchoBackMultiBit

*1　API-TIMER(WDM)を併用することで実現可能です。
*2　WDMドライバは標準でマルチプロセスに対応しているため必要ありません。
*3　WDMドライバはPIO-32D(PM)に対応していません。

(前ページからの続き)

API-DIO(98/PC)ドライバ		API-DIO(WDMドライバ
関数名	機能	関数名
《基本機能関数》デジタル入出力関数　(バックグラウンド入出力関数)		
DioInpBack	バックグラウンドでの指定ポートのデジタル入力処理	×（該当関数無し）
DioOutBack	バックグラウンドでの指定ポートのデジタル出力処理	×（該当関数無し）
DioBitInpBack	バックグラウンドでの指定ビットのデジタル入力処理	×（該当関数無し）
DioBitOutBack	バックグラウンドでの指定ビットのデジタル出力処理	×（該当関数無し）
DioSts	関数のバックグラウンドでの実行状態の取得	×（該当関数無し）
DioStop	バックグラウンドでのデジタル入出力処理の停止	×（該当関数無し）
《基本機能関数》割り込み機能関数		
DioEvent	割り込みイベントの設定	DioNotifyInterrupt
DioIntEnable	割り込みイベント信号入力の禁止/許可の設定	DioNotifyInterrupt
DioIntSence	割り込みコントロールポートのステータスの取得	×（該当関数無し）　*4
DioEventEx	割り込みイベントの設定（拡張版）	DioNotifyInterrupt
DioIntEnableEx	割り込みイベント信号入力の禁止/許可の設定（拡張版）	DioNotifyInterrupt
《高機能関数》プロセスコントロール機能関数		
DioPtnSet	プロセスパターンの設定	×（該当関数無し）
DioPtnStart	プロセスコントロール開始	×（該当関数無し）
DioPtnSts	プロセス開始時のステータスの取得	×（該当関数無し）
《高機能関数》トリガ監視機能関数		
DioTrgSet	トリガ監視ビットの設定	DioNotifyTrg　*5
DioTrgStart	トリガ開始	DioNotifyTrg　*5
DioTrgSts	トリガイベント発生時のステータスの取得	DioNotifyTrg　*5
《高機能関数》コード変換機能関数		
DioInpBCD	指定ポートのデータをBCDコードに変換して入力	×（該当関数無し）
DioOutBCD	指定ポートのデータをBCDコードに変換して出力	×（該当関数無し）
DioNInpBCD	指定ポートのデータをBCDコードに変換して入力 (負論理)	×（該当関数無し）
DioNOutBCD	指定ポートのデータをBCDコードに変換して出力(負論理)	×（該当関数無し）
《その他》		
LpByte	符号なしバイト変数のアドレス取得(Visual Basic)	×（該当関数無し）　*6
LpWord	符号なしワード変数のアドレス取得(Visual Basic)	×（該当関数無し）　*6
LpDWord	符号なしダブルワード変数のアドレス取得(Visual Basic)	×（該当関数無し）　*6

*4　WDMドライバでは関数の仕様上、こちらに相当する関数は必要ありません。
*5　WDMドライバのトリガ監視にはトリガ発生時に他のビットデータを入力する機能はありません。
*6　WDMドライバでは関数の仕様上、VBでもこちらに相当する関数を使用する必要はありません。

6.1 ヘッダファイル、ライブラリファイルの準備

API関数ライブラリを使用するためには、作成したプロジェクトに、専用のライブラリや
ヘッダファイル等をリンクする必要があります。これらは言語によって方法が異なります。
なお、プロジェクトの作成方法は、ご使用の開発言語のオンラインヘルプ等をご参照ください。

◆Visual Basic .NET、6.0、5.0の場合
Visual Basicでプログラムを作成するには、プロジェクトファイルを作成し、
APIを呼び出すための宣言の入った標準モジュールをプロジェクトに追加して使用します。
標準モュールは、サンプルプログラムの場所にあります。

標準モジュール名　：CDIO.VB（Visual Basic .NET）
　　：CDIO.BAS（Visual Basic 6.0、5.0）

◆Visual C#.NETの場合
Visual C#でプログラムを作成するには、プロジェクトファイルを作成し、
APIを呼び出すための宣言の入ったクラスファイルをプロジェクトに追加して使用します。
クラスファイルは、サンプルプログラムの場所にあります。

クラスファイル名：CDIOCS.CS

◆Visual C++ .NET(MFC)、6.0、5.0の場合
VisualC++でプログラムを作成するには、プロジェクトファイルを作成し、コンパイルおよび
リンクによって、実行ファイルを作成します。
コンパイルを行うときは、アプリケーションプログラムに、C言語用ヘッダファイルをインクルードします。
また、リンクを行うときには、インポートライブラリをプロジェクトに追加して、リンクしてください。
ヘッダファイルとLIBファイルは、サンプルプログラムの場所にあります。

ヘッダファイル　：CDIO.H
ライブラリ　　：CDIO.LIB

6.2 WDMドライバの基本的なプログラミング

関数の使い方は開発環境インストール時にインストールされるヘルプファイルの
[関数リファレンス] に記載されていますのでご参照ください。

ここでは、基本的なプログラミングの流れを説明します。
(Visual Basic.NETの例で紹介しますが、他の言語でも同様です)

◆ドライバの初期化/終了とハンドル
API関数ライブラリでは、デバイスにアクセスするために、デバイスドライバを使用しています。
デバイスドライバを使用するには初期化を、使い終わったら終了処理を行います。
初期化を行うと対象のデバイスをあらわすID(ハンドル)が返りますので、以降の条件設定や
入出力などの関数では、このIDでデバイスを指定します。
複数のデバイスを使用する場合には、使用するデバイスごとに初期化を行います。

Visual Basic.NETでデジタル入出力ボードを使用する例

```
'Idはデバイスドライバのハンドルを取得します。
'これは他の関数を実行するときに必要ですので、グローバル変数として宣言します。

Dim Id As Short                 'デバイスID（グローバル変数で宣言）
Dim Ret As Integer              '戻り値
Dim InpBitData As Byte          'デジタル入力値を格納

'初期化
Ret = DioInit ( "DIO000", Id )  ‘デバイス名”DIO000”のハンドル値がIdに返ります

'データ入力
Ret = DioInpBit ( Id, 0, InpBitData )  'DioInitで取得したIdを指定します

'終了処理
Ret = DioExit ( Id )            ‘終了処理を行うデバイスのIdを指定します
```

※ここでは説明のために、DioInit → DioInpBit → DioExit と記述していますが、
実際のアプリケーションで何度も入力や出力を行う場合、入出力の関数のみを繰り返し、
DioInit と DioExit は、使用開始時と終了時(通常はアプリケーションを終了するとき)に
それぞれ1度だけ実行します。

◆複数のデバイスを使用する場合
複数のデバイスを使用する場合は、それぞれのデバイスに対して初期化を実行します。
その為、ハンドルを格納するグローバル変数を複数用意する必要があります。

```
Dim Id1 As Short                          'デバイスID1
Dim Id2 As Short                          'デバイスID2
Dim Ret As Integer                        '戻り値
Dim InpBitData1 As Byte                   'デジタル入力値1を格納
Dim InpBitData2 As Byte                   'デジタル入力値2を格納

'初期化
Ret = DioInit ( “DIO000”, Id1 )           ‘デバイス１を初期化
Ret = DioInit ( “DIO001”, Id2 )           ‘デバイス２を初期化

'データ入力
Ret = DioInpBit ( Id1, 0, InpBitData1 )   'デバイス１の0ビット入力
Ret = DioInpBit ( Id2, 2, InpBitData2 )   'デバイス２の2ビット入力

'終了処理
Ret =DioExit ( Id1 )
Ret =DioExit ( Id2 )
```

◆イベント通知
割り込みやトリガ検出などのイベントは、ウィンドウメッセージとしてアプリケーションに通知されます。

6.3 98/PCからWDMドライバへの置換例

ここでは、98/PCからWDMドライバへのプログラムの置換について具体例をもって説明します。

※Visual Basicの例で紹介しますが、他の言語でも同様です
※98/PCドライバ側をVB6、WDMドライバ側をVisual Basic.NETとして記載しています
　他の開発言語の場合はヘルプにて書式や変数の型などを確認の上、ご参考ください

◆VB6とVisual Basic.NETでのデータ型の違いについて
　以下のように、型名とサイズが変更されていますので注意が必要です。

サイズ	値の範囲	VB6	VB.NET
16bit 整数型	-32,768 ～ 32,767	Integer	Short
32bit 整数型	-2,147,483,648 ～ 2,147,483,647	Long	Integer
64bit 整数型	-9,223,372,036,854,775,808 ～ 9,223,372,036,854,775,807	-	Long

◆初期化関数の置換(DioOpen関数をご使用だった場合)

98/PCドライバ

```
'初期化関数 変数宣言、引数設定

Dim hDrv As Long
Dim DrvNo As Integer
Dim GrpNo As Integer
Dim Ret As Long

DrvNo = 0          'ドライバ番号
GrpNo = 1          'グループ番号

'初期化関数実行

Ret = DioOpen ( hDrv, DrvNo, GrpNo )
```

WDMドライバ

```
Dim Id As Short
Dim DeviceName As String
Dim Ret As Integer

DeviceName = "DIO000"    'デバイス名

Ret = DioInit ( DeviceName, Id )
```

◆初期化関数の置換(DioOpenEx関数をご使用だった場合)

98/PCドライバ

```
'初期化関数 変数宣言、引数設定

Dim hDrv As Long
Dim DeviceName As String
Dim Ret As Long

DeviceName = "DIO000"    'デバイス名

'初期化関数実行

Ret = DioOpenEx ( DeviceName, hDrv )
```

WDMドライバ

```
Dim Id As Short
Dim DeviceName As String
Dim Ret As Integer

DeviceName = "DIO000"    'デバイス名

Ret = DioInit ( DeviceName, Id )
```

◆デジタル入力関数の置換(簡易入力関数)

98/PCドライバ

```
'デジタル入力関数 変数宣言、引数設定
Dim hDrv As Long
Dim PortNo As Integer
Dim Data As Byte
Dim Ret As Long

PortNo = 0              '入力論理ポート番号

'デジタル入力関数実行
Ret = DioInpByte(hDrv, PortNo, Data)
```

WDMドライバ

```
Dim Id As Short
Dim PortNo As Short
Dim Data As Byte
Dim Ret As Integer

PortNo = 0              '入力論理ポート番号


Ret = DioInpByte(Id, PortNo, Data)
```

◆デジタル入力関数の置換(複数バイト入力関数)

98/PCドライバ

```
'デジタル入力関数 変数宣言、引数設定
Dim hDrv As Long
Dim lpDInp As DINP
Dim PortNo(1) As Integer
Dim Data(1) As Byte
Dim Ret As Long

lpDInp.PortNum = 2    'ポート数
PortNo(0) = 0         '入力論理ポート番号
PortNo(1) = 1         '入力論理ポート番号

'変数のアドレス取得
lpDInp.InpPortNo =LpWord(PortNo(0))
lpDInp.Buf = LpByte(Data(0))

'デジタル入力関数実行
Ret = DioInp(hDrv, lpDInp)
```

WDMドライバ

```
Dim Id As Short
Dim PortNo(1) As Short
Dim PortNum As Short
Dim Data(1) As Byte
Dim Ret As Integer

PortNum = 2           'ポート数
PortNo(0) = 0         '入力論理ポート番号
PortNo(1) = 1         '入力論理ポート番号




Ret = DioInpMultiByte(Id, PortNo, PortNum, Data)
```

◆デジタル出力関数の置換(簡易出力関数)

98/PCドライバ

```
'デジタル出力関数 変数宣言、引数設定
Dim hDrv As Long
Dim PortNo As Integer
Dim Data As Byte
Dim Ret As Long

PortNo = 0          '出力論理ポート番号
Data = &H55         '出力データ

'デジタル出力関数実行
Ret = DioOutByte(hDrv, PortNo, Data)
```

WDMドライバ

```
Dim Id As Short
Dim PortNo As Short
Dim Data As Byte
Dim Ret As Integer

PortNo = 0          '出力論理ポート番号
Data = &H55         '出力データ


Ret = DioOutByte(Id, PortNo, Data)
```

◆デジタル出力関数の置換(複数バイト出力関数)

98/PCドライバ

```
'デジタル出力関数 変数宣言、引数設定
Dim hDrv As Long
Dim lpDOut As DOUT
Dim PortNo(1) As Integer
Dim Data(1) As Byte
Dim Ret As Long

lpDOut.PortNum = 2    'ポート数
PortNo(0) = 0         '出力論理ポート番号
PortNo(1) = 1         '出力論理ポート番号
Data(0) = &H55        '出力データ
Data(1) = &HAA        '出力データ

'変数のアドレス取得
lpDOut.OutPortNo = LpWord(PortNo(0))
lpDOut.Buf = LpByte(Data(0))

'デジタル出力関数実行
Ret = DioOut(hDrv, lpDOut)
```

WDMドライバ

```
Dim Id As Short
Dim PortNo(1) As Short
Dim PortNum As Short
Dim Data(1) As Byte
Dim Ret As Integer

PortNum = 2           'ポート数
PortNo(0) = 0         '出力論理ポート番号
PortNo(1) = 1         '出力論理ポート番号
Data(0) = &H55        '出力データ
Data(1) = &HAA        '出力データ




Ret = DioOutMultiByte(Id, PortNo, PortNum, Data)
```

◆終了関数の置換

98/PCドライバ	WDMドライバ
‘終了関数 変数宣言、引数設定	
Dim Ret As Long	Dim Ret As Integer
‘終了関数実行	
Ret = DioClose (hDrv)	Ret = DioExit (Id)

6.4 WDMドライバの便利な機能

WDMドライバの便利な機能をご紹介します。必要に応じてお客様のプログラムにご利用ください。

◆デバイス一覧を取得する関数を実装

パソコンに実装しているデバイス一覧が取得できる関数を実装しています。
下記のようなシチュエーションで便利にお使いいただけます。
・デバイスは1つのみ使用し、インストール時のデバイス名に左右されずに使用する場合。
・使用するデバイスを選択させるような、汎用的なアプリケーションを作成する場合。

関数名：
DioQueryDeviceName

機能：
使用可能なデバイスの一覧を取得します。

書式：
Ret = DioQueryDeviceName (Index , DeviceName , Device)

引数：
Index
最初に0を指定し、以降+1ずつ値を指定します。
DeviceName
デバイス名を返します。(例："DIO000")
Device
デバイス名称を返します。(例："DIO-8/8(USB)")

戻り値：

値	意味
0	正常終了。
10051	利用可能なデバイスが見つかりません。 デバイスが、デバイスマネージャに登録されているかを確認してください。
20032	情報取得でデバイスの情報が見つかりませんでした。

使用例：
Index=0で情報を取得します。
VB.NETの場合

```
Dim Ret As Integer
Dim DeviceName As String = Space(256)
Dim Device As String = Space(256)
Ret = DioQueryDeviceName ( 0 , DeviceName , Device )
```

改定履歴

年 月	改定内容
2018年4月	初版
2019年2月	改定 体裁調整、誤記修正

デジタル入出力ボード/USB製品用
Windowsドライバ『API-DIO(WDM)』
従来ドライバ『API-DIO(98/PC)』からの移行ガイド

発行日　2019 年 2 月 21 日

発行責任　株式会社　コンテック

© 2019 CONTEC CO.,LTD.Printed in Japan

本書の一部または全部を、無断で複写、複製、転載することを禁じます。