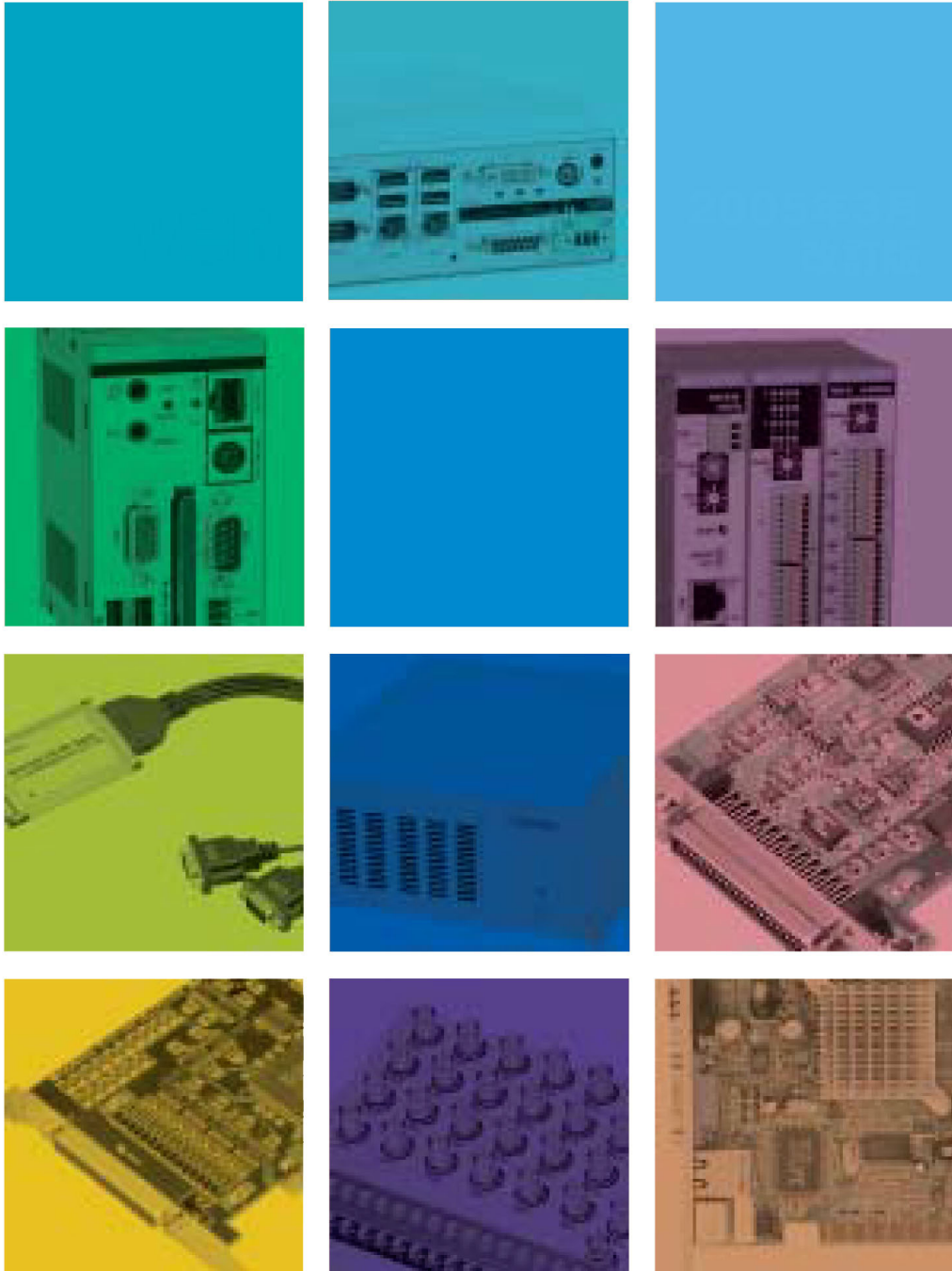


# デジタル入出力 ビギナーズ・ガイドブック

Visual Basic6.0プログラミング編



株式会社コンテック

[www.contec.co.jp](http://www.contec.co.jp)

- ※ Microsoft、Windows、Visual Basic、ActiveXは、米国Microsoft Corporationの商標または登録商標です。
- ※ F&eIT、FLEXLANは株式会社コンテックの登録商標です。
- ※ その他、本書に記載した会社名、商品名は一般的に各社の商標または登録商標です。
- ※ 本書では、®、™マークは省略しています。
- ※ 本書に掲載したプログラムは、すべての動作を保証するものではありません。
- ※ 本書に掲載したプログラムを実行し、お客様または第三者が被った直接的、または間接的ないかなる損害について株式会社コンテックは責任を負わないものとし、一切の賠償などは行わないものとします。
- ※ 本書の著作権は、株式会社コンテックにあります。本書の内容の一部または全部を無断で転載、複写、電子化することはできません。
- ※ 内容および対応製品、製品型式は予告なく改訂・改版することがあります(2006年4月現在)。

# はじめに

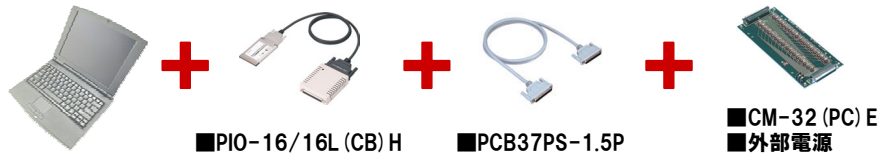
現在、装置制御や自動計測システムを構築する上で、測定データの保存、解析や情報共有などが簡単にできることから、さまざまな分野でパソコンを応用利用しようという考え方が広がっています。しかしながら、はじめてパソコンによる計測・制御システムを手掛けようとする方にとって、学習するための機関や参考書籍が不足しているのが現状です。このような背景からコンテック（以下、弊社）では、Windows におけるハードウェア制御の基礎、計測・制御プログラミングを分かり易く解説する『ビギナーズ・ガイドブック』を発行する事となりました。本書では、デジタル入出力の概要と用語解説、弊社製デジタル入出力インターフェイスを使用したVisual Basicの初歩的な計測・制御プログラム開発手順を解説しています。また、プログラミングに必要なハードウェアの基礎知識に関しても、分かり易く解説しています。

## ■ 本書にて使用しているプログラム実習環境

### ● 第5章 Visual Basic6.0によるデジタル入出力プログラミング

- ① OS : Microsoft Windows XP Professional/Home Edition
- ② 開発言語 : Microsoft Visual Basic 6.0
- ③ インターフェイス : CardBus対応絶縁型デジタル入出力カード 【型式:PIO-16/16L (CB) H】
- ④ ケーブル : 37ピンD-SUB両側コネクタ付シールドケーブル 【型式:PCB37PS-1.5P】
- ⑤ アクセサリ : デジタル入出力信号モニタ (チェックメイト) 【型式:CM-32 (PC) E】
- ⑥ ソフトウェア : Windows版デジタル入出力ドライバソフトウェア (製品添付) 【型式:API-DIO (98/PC)】  
: ソフトウェア・チェックメイト (ドライバ同梱) 【型式:CM\_64】

#### 構成例 ①



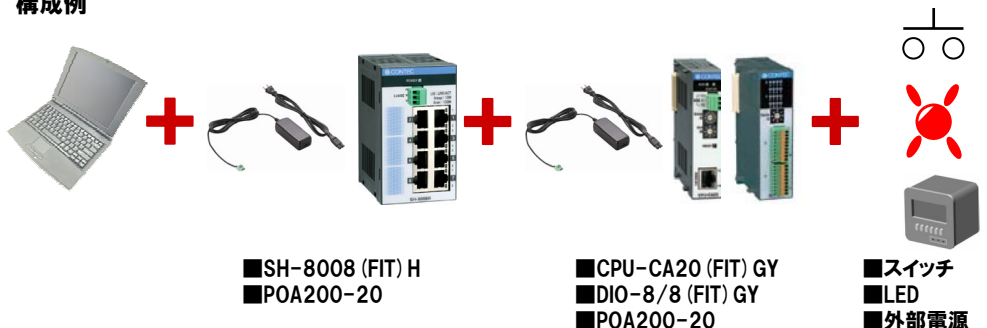
#### 構成例 ② (ハードウェアが用意できない場合)



### ● 第6章 リモートI/O

- ① OS : Microsoft Windows XP Professional/Home Edition
- ② 開発言語 : Microsoft Visual Basic 6.0
- ③ リモートI/O機器 : リモートI/Oコントローラモジュール 【型式:CPU-CA20 (FIT) GY】  
: I/Oデバイスモジュール (デジタル入出力) 【型式:DIO-8/8 (FIT) GY】  
: ACアダプタ電源 ×2台 【型式:POA200-20】  
: スwitchングHUBユニット 【型式:SH-8008 (FIT) H】
- ④ ソフトウェア : I/Oコントローラモジュール用Windowsドライブライブラリ (CPU-CA20 (FIT) GY添付) 【型式:API-CAP (W32)】

#### 構成例



## ■ 本書にて作成するサンプルプログラム一覧

### ● 第5章 Visual Basic6.0によるデジタル入出力プログラミング

- ① タイマコントロールを使用したカウントアッププログラム [プログラムリスト:5-12頁]
- ② ポート単位のデジタル入出力プログラム [プログラムリスト:5-33頁]
- ③ ビット単位のデジタル入出力プログラム [プログラムリスト:5-42頁]
- ④ 割り込み入力プログラム [プログラムリスト:5-57頁]

### ● 第6章 リモートI/O

- ① リモートI/Oによるデジタル入出力プログラム(ビット単位) [プログラムリスト:6-25頁]

本書で作成した上記サンプルプログラムのソースコードは弊社ホームページからダウンロードできます。

<http://www.contec.co.jp/support/technical/tutorial/>

※ 弊社では、ハードウェアの無料貸出を行っております。詳細は、弊社ホームページにて → <http://www.contec.co.jp>

※ 本書では、前項の環境にて解説を行っております。お客様の環境により、ハードウェアの実装手順やドライバのインストール方法が異なる場合があります。

※ 本書にて記載しているプログラムは、前項の環境外での動作保証はいたしませんので、予めご了承ください。

※ デジタル入出力ドライバソフトウェアは、弊社ホームページから無償ダウンロードが可能です。

<http://www.contec.co.jp/apipac/>



# 目次

<b>第1章 パソコンによる計測・制御システム</b> . . . . .	1-2
1-1. センサの主な種類と出力信号例 . . . . .	1-3
1-2. アクチュエータの主な種類と入力信号例 . . . . .	1-3
1-3. システム事例：溶解炉監視制御システム . . . . .	1-3
1-4. 計測・制御用インターフェイスの種類 . . . . .	1-4
デジタル入力	
デジタル出力	
アナログ入力	
アナログ出力	
シリアル通信	
GPIB通信	
カウンタ入力	
モータコントロール	
1-5. パソコン拡張バス・スロットの種類 . . . . .	1-4
PCIバス	
LowProfile PCI	
PCカード(PCMCIA)	
USB(Universal Serial Bus)	
LAN(Ethernet)	
1-6. コンテックのパソコン計測・制御バリエーション . . . . .	1-5
<b>第2章 電気回路の基礎知識</b> . . . . .	2-2
2-1. 電流は電圧の高い方から低い方に流れる . . . . .	2-2
2-2. ダイオードの性質 . . . . .	2-2
2-3. トランジスタの性質 . . . . .	2-3
2-4. フォトカプラの性質 . . . . .	2-3
2-5. 誘導負荷のサージ電圧対策 . . . . .	2-4
<b>第3章 デジタル入出力の基礎知識</b> . . . . .	3-2
3-1. デジタル入出力とは . . . . .	3-2
3-2. デジタル入出力ボード／カードの種類 . . . . .	3-2
3-3. デジタル入出力の適応例 . . . . .	3-3
3-4. デジタル入出力ボードの種類と用途 . . . . .	3-4
3-4-1. フォトカプラ絶縁入出力タイプ . . . . .	3-4
3-4-2. 高速フォトカプラ絶縁入出力タイプ . . . . .	3-4
3-4-3. 電源内蔵フォトカプラ絶縁入出力タイプ . . . . .	3-4
3-4-4. 接点出力タイプ . . . . .	3-4
3-4-5. 非絶縁入出力タイプ . . . . .	3-5
3-4-6. 双方向入出力タイプ . . . . .	3-5
TOPICS：バッファ(ドライブ回路)付きボードとは . . . . .	3-5
TOPICS：i8255とは . . . . .	3-5
3-5. 絶縁タイプと非絶縁タイプ(TTL)との機能比較 . . . . .	3-5
TOPICS：TTL、LVTTLとは . . . . .	3-5

3-6. 出力回路の種類	3-6
3-6-1. トランジスタ出力(無接点出力)	3-6
① フォトカプラ絶縁オープンコレクタ出力(シンクタイプ)	3-6
② フォトカプラ絶縁出力(ソースタイプ)	3-6
③ 非絶縁TTLオープンコレクタ出力(負論理)	3-7
④ 非絶縁TTLレベルドライバ出力(負論理)	3-7
TOPICS: オープンコレクタ出力とは	3-7
⑤ フォトカプラ絶縁TTLドライバ出力(負論理)	3-8
3-6-2. 接点出力(有接点出力)	3-8
3-7. 入力回路の種類	3-9
3-7-1. DC入力	3-9
① フォトカプラ絶縁入力(電流シンク出力対応)	3-9
② フォトカプラ絶縁入力(電流ソース出力対応)	3-9
③ 非絶縁TTLレベル入力(負論理)	3-10
④ フォトカプラ絶縁TTLレベル入力(負論理)	3-10
3-8. 双方向回路の種類	3-11
3-8-1. 双方向回路	3-11
① 非絶縁TTLレベル入出力(正論理)	3-11
② バッファ付き非絶縁TTL/LVTTLレベル入出力(正論理)	3-11
3-9. 用語解説	3-12
3-9-1. 入出力点数	3-12
3-9-2. 操作回路電圧	3-13
3-9-3. コモン構成	3-13
① **点/1コモン	3-13
② 全点共通	3-13
③ 全点独立(独立コモンタイプ)	3-13
3-9-4. 応答速度	3-14
3-9-5. デジタルフィルタ	3-14
3-9-6. 入力抵抗	3-15
3-9-7. 最大定格	3-15
3-9-8. 消費電流	3-15
3-9-9. 電源内蔵型ボード	3-15
3-9-10. バスマスタ転送機能	3-16
3-9-11. I/OポートとI/Oポートアドレス	3-17
3-9-12. 割り込み	3-17

<b>第4章 ハードウェア・ソフトウェアのセットアップ</b> . . . . .	4-2
4-1. セットアップからプログラム開発までの流れ (Windows系OSの場合) . . . . .	4-2
4-2. 手順① : ソフトウェア (ドライバソフトウェア) のインストール . . . . .	4-3
4-3. 手順② : ハードウェアの設定 . . . . .	4-4
4-4. 手順③ : ハードウェアのインストール . . . . .	4-5
4-5. 手順④ : ドライバソフトウェアの初期設定 . . . . .	4-6
4-6. 手順⑤ : ハードウェア・ソフトウェアの動作確認 . . . . .	4-7
4-6-1. 確認方法 . . . . .	4-7
4-7. 手順⑥ : 外部機器との接続 . . . . .	4-8
4-7-1. 外部機器との接続形態 (接続方法) . . . . .	4-8
4-7-2. 仮想のデジタル入出力機器 . . . . .	4-8
4-7-3. インターフェイスコネクタの信号配置図とI/Oポートマップ . . . . .	4-9
4-7-4. 外部機器との配線 . . . . .	4-10
4-7-5. PIO-16/16L (CB)Hの入出力回路 . . . . .	4-11
①PIO-16/16L (CB)Hの入力回路図 . . . . .	4-11
②PIO-16/16L (CB)Hの出力回路図 . . . . .	4-11
4-8. デモボード (仮想ハードウェア) の登録・設定方法 . . . . .	4-12
4-8-1. デモボードの登録方法 . . . . .	4-12
4-8-2. デモボード操作ユーティリティ (CM_64) の実行方法 . . . . .	4-13
4-9. 参考資料 (ドライバソフトウェアの必要性) . . . . .	4-14
4-10. 参考資料 (DLL : Dynamic Link Libraryとは) . . . . .	4-15
4-10-1. Windows環境における実行ファイル (.EXEファイル) . . . . .	4-15
<b>第5章 Visual Basic6.0によるデジタル入出力プログラミング</b> . . . . .	5-2
5-1. Visual Basicでのプログラム開発 . . . . .	5-2
5-1-1. イベント駆動型 (イベントドリブン) 言語とは . . . . .	5-2
5-1-2. Visual Basic6.0用語解説 . . . . .	5-3
5-1-3. Visual Basicのプログラム作成手順 . . . . .	5-3
5-2. タイマコントロールを使用したカウントアッププログラム . . . . .	5-4
5-2-1. プログラムフローチャート . . . . .	5-4
5-2-2. プログラム作成手順① (Visual Basicの起動) . . . . .	5-4
5-2-3. プログラム作成手順② (コントロールの配置) . . . . .	5-6
5-2-4. プログラム作成手順③ (フォームのプロパティ設定) . . . . .	5-7
5-2-5. プログラム作成手順④ (コマンドボタンコントロールのプロパティ設定) . . . . .	5-7
5-2-6. プログラム作成手順⑤ (Labelコントロールのプロパティ設定) . . . . .	5-8
TOPICS : Visual Basicのオブジェクト名の付け方 (プリフィックス) に関して . . . . .	5-8
5-2-7. プログラム作成手順⑥ (タイマコントロールのプロパティ設定) . . . . .	5-9
TOPICS : タイマコントロールに関して . . . . .	5-9
5-2-8. プログラム作成手順⑦ (タイマ開始処理の記述) . . . . .	5-10
TOPICS : オブジェクトの指定方法とプロシージャの指定方法 . . . . .	5-10
TOPICS : Visual Basicのインテリセンス機能 . . . . .	5-10
5-2-9. プログラム作成手順⑧ (タイマ停止処理の記述) . . . . .	5-11
5-2-10. プログラム作成手順⑨ (変数の宣言とカウントアップ&表示処理の記述) . . . . .	5-11
5-2-11. カウントアッププログラムリスト . . . . .	5-12
5-2-12. プログラム作成手順⑩ (プログラムの保存と動作の確認) . . . . .	5-13
TOPICS : Visual Basicのタイマコントロールと弊社提供のタイマコントロールとの違い . . . . .	5-13

5-3. ポート単位のデジタル入出力プログラム	5-14
5-3-1. プログラム概要	5-14
5-3-2. 実行環境	5-14
5-3-3. プログラムフローチャート	5-15
5-3-4. Windows版ドライバソフトウェア『API-PAC(W32)』の基礎知識	5-15
①ドライバソフトウェア『API-PAC(W32)』とは	5-15
②初期設定ユーティリティ『API-TOOL Configuration』とは	5-16
③論理デバイスとは	5-16
④グループとは	5-16
⑤ドライバ番号とは	5-16
⑥API-PAC(W32)の処理体系	5-17
⑦プログラム構成	5-17
5-3-5. プログラム作成手順①(画面の作成：オブジェクトの配置とプロパティ設定)	5-18
5-3-6. プログラム作成手順②(標準モジュールファイルの追加)	5-18
5-3-7. プログラム作成手順③(変数の追加)	5-19
5-3-8. プログラム作成手順④(初期化処理の追加)	5-19
TOPICS：デバイスハンドルとは	5-20
5-3-9. プログラム作成手順⑤(終了処理の追加)	5-21
5-3-10. プログラム作成手順⑥(タイマ開始処理の記述)	5-22
5-3-11. プログラム作成手順⑦(タイマ停止処理の記述)	5-22
5-3-12. プログラム作成手順⑧(デジタル入力処理の追加)	5-23
5-3-13. プログラムの実行	5-24
5-3-14. 参考資料(ビットの操作)	5-27
5-3-15. デジタル出力処理の追加	5-30
5-3-16. プログラムの実行	5-31
5-3-17. ポート単位のデジタル入力プログラムリスト	5-32
5-3-18. ポート単位のデジタル入出力プログラムリスト	5-33
5-4. ビット単位のデジタル入出力プログラム	5-34
5-4-1. プログラム概要	5-34
5-4-2. プログラムフローチャート	5-34
5-4-3. PIO-16/16L(GB)Hのビット割付図	5-34
5-4-4. プログラム作成手順①(画面の作成：オブジェクトの配置とプロパティ設定)	5-35
5-4-5. プログラム作成手順②(標準モジュールファイルの追加)	5-35
5-4-6. プログラム作成手順③(変数の追加)	5-36
5-4-7. プログラム作成手順④(初期化処理の追加)	5-36
5-4-8. プログラム作成手順⑤(終了処理の追加)	5-38
5-4-9. プログラム作成手順⑥(タイマ開始処理の記述)	5-39
5-4-10. プログラム作成手順⑦(タイマ停止処理の記述)	5-39
5-4-11. プログラム作成手順⑧(デジタル入出力処理の追加)	5-40
5-4-12. ビット単位のデジタル入出力関数リファレンス	5-41
5-4-13. ビット単位のデジタル入出力プログラムリスト	5-42
5-4-14. プログラムの実行	5-43

5-5. 割り込み処理(割り込み入力プログラム) . . . . .	5-44
5-5-1. 割り込み処理とは . . . . .	5-44
5-5-2. Windowsにおける割り込み処理 . . . . .	5-44
①メッセージとは . . . . .	5-44
②Visual Basicにおける割り込み入力処理 . . . . .	5-44
5-5-3. 割り込み入力プログラムの概要 . . . . .	5-45
5-5-4. 実行環境 . . . . .	5-45
5-5-5. プログラムフローチャート . . . . .	5-45
5-5-6. プログラム作成手順①(画面の作成とプロパティ設定) . . . . .	5-46
5-5-7. プログラム作成手順②(標準モジュールファイルの追加) . . . . .	5-46
5-5-8. プログラム作成手順③(変数の追加) . . . . .	5-47
5-5-9. プログラム作成手順④(Msgecho.ocxコントロールの追加) . . . . .	5-47
5-5-10. プログラム作成手順⑤(初期化処理の追加) . . . . .	5-48
5-5-11. プログラム作成手順⑥(終了処理の追加) . . . . .	5-49
5-5-12. プログラム作成手順⑦(タイマ開始コードの記述) . . . . .	5-50
5-5-13. プログラム作成手順⑧(タイマ停止コードの記述) . . . . .	5-50
5-5-14. プログラム作成手順⑨(カウントアップ処理コードの記述) . . . . .	5-51
5-5-15. プログラム作成手順⑩(割り込み監視開始処理の追加) . . . . .	5-53
5-5-16. プログラム作成手順⑪(割り込み発生時処理の追加) . . . . .	5-54
TOPICS : DioEventとDioEventExの使い分け . . . . .	5-54
5-5-17. プログラムの実行 . . . . .	5-55
TOPICS : MsgBox関数について . . . . .	5-56
5-5-18. 割り込み入力プログラムリスト . . . . .	5-57
5-6. ドライバソフトウェア提供関数一覧 . . . . .	5-58
TOPICS : BCD(2進化10進数)とは . . . . .	5-59

## 第6章 リモートI/O . . . . . 6-2

6-1. リモートI/Oとは . . . . .	6-2
6-2. イーサネットベースのリモートI/O製品紹介 . . . . .	6-2
①I/Oコントローラモジュール : CPU-CA20(FIT)GY . . . . .	6-3
②I/Oデバイスモジュール一覧 . . . . .	6-3
③I/Oコントローラモジュール用 Windowsドライブライブラリ API-CAP(W32) . . . . .	6-4
④F&EITシリーズ スイッチングHUB SH-8008(FIT)H . . . . .	6-5
⑤無線LANアクセスポイントとステーション FLEXLAN DS540シリーズ . . . . .	6-5
⑥F&EITシリーズの全容 . . . . .	6-5
6-3. リモートI/Oによるデジタル入出力プログラム(ビット単位) . . . . .	6-6
6-3-1. リモートI/Oによるデジタル入出力プログラム概要 . . . . .	6-6
6-3-2. 使用機器一覧 . . . . .	6-6
6-3-3. 機器接続図 . . . . .	6-7
6-3-4. 使用手順 . . . . .	6-8
TOPICS : I/Oアシストサーバユニット : SVR-10A2(FIT)GYとは . . . . .	6-8
6-3-5. ハードウェアの準備(各種IDの設定) . . . . .	6-9
6-3-6. ハードウェアの準備(I/Oデバイスモジュールの接続) . . . . .	6-10
6-3-7. ハードウェアの準備(I/Oデバイスモジュールと外部機器との接続) . . . . .	6-11
6-3-8. ハードウェアの準備(ホストPCとI/Oコントローラユニットとの接続) . . . . .	6-12
6-3-9. F&EIT設定ユーティリティによるセットアップ(ユーティリティの実行) . . . . .	6-12

6-3-10. F&EIT設定ユーティリティによるセットアップ(ネットワークの設定) . . . . .	6-13
6-3-11. F&EIT設定ユーティリティによるセットアップ(デバイス固有の設定) . . . . .	6-13
6-3-12. F&EIT設定ユーティリティによるセットアップ(デバイス名の設定) . . . . .	6-14
6-3-13. F&EIT設定ユーティリティによるセットアップ(設定の保存) . . . . .	6-14
6-3-14. 動作確認(診断モニタの実行) . . . . .	6-15
6-3-15. Visual Basicの起動と画面の作成 . . . . .	6-16
6-3-16. 画面構成 . . . . .	6-17
6-3-17. 各オブジェクトのプロパティ設定 . . . . .	6-17
6-3-18. 標準モジュールファイルの追加 . . . . .	6-19
6-3-19. API-CAP (W32) の処理体系 . . . . .	6-20
6-3-20. 変数の宣言 . . . . .	6-20
6-3-21. 初期化処理の記述 . . . . .	6-21
6-3-22. 終了処理の記述 . . . . .	6-22
6-3-23. 入力処理の記述 . . . . .	6-23
6-3-24. 出力処理の記述 . . . . .	6-24
TOPICS:Val関数(Visual Basic標準関数) . . . . .	6-24
6-4. デジタル入出力カード(PI0-16/16L (GB)H)とのプログラム比較 . . . . .	6-25
6-5. API-CAP (W32) デジタル入出力ドライバ提供関数一覧 . . . . .	6-26
<b>付録(各種開発ツール紹介) . . . . .</b>	<b>付録-1</b>
①計測システム開発用ActiveXコンポーネント集 ACX-PAC (W32) Ver4.0 . . . . .	付録-2
②Windows版ドライブライブラリ API-TOOL for Windows . . . . .	付録-4
③Linux版ドライブライブラリ API-TOOL for Linux . . . . .	付録-5
④LabVIEW対応サポートソフトウェア . . . . .	付録-6
⑤MATLAB対応データ収録用ライブラリ ML-DAQ . . . . .	付録-7

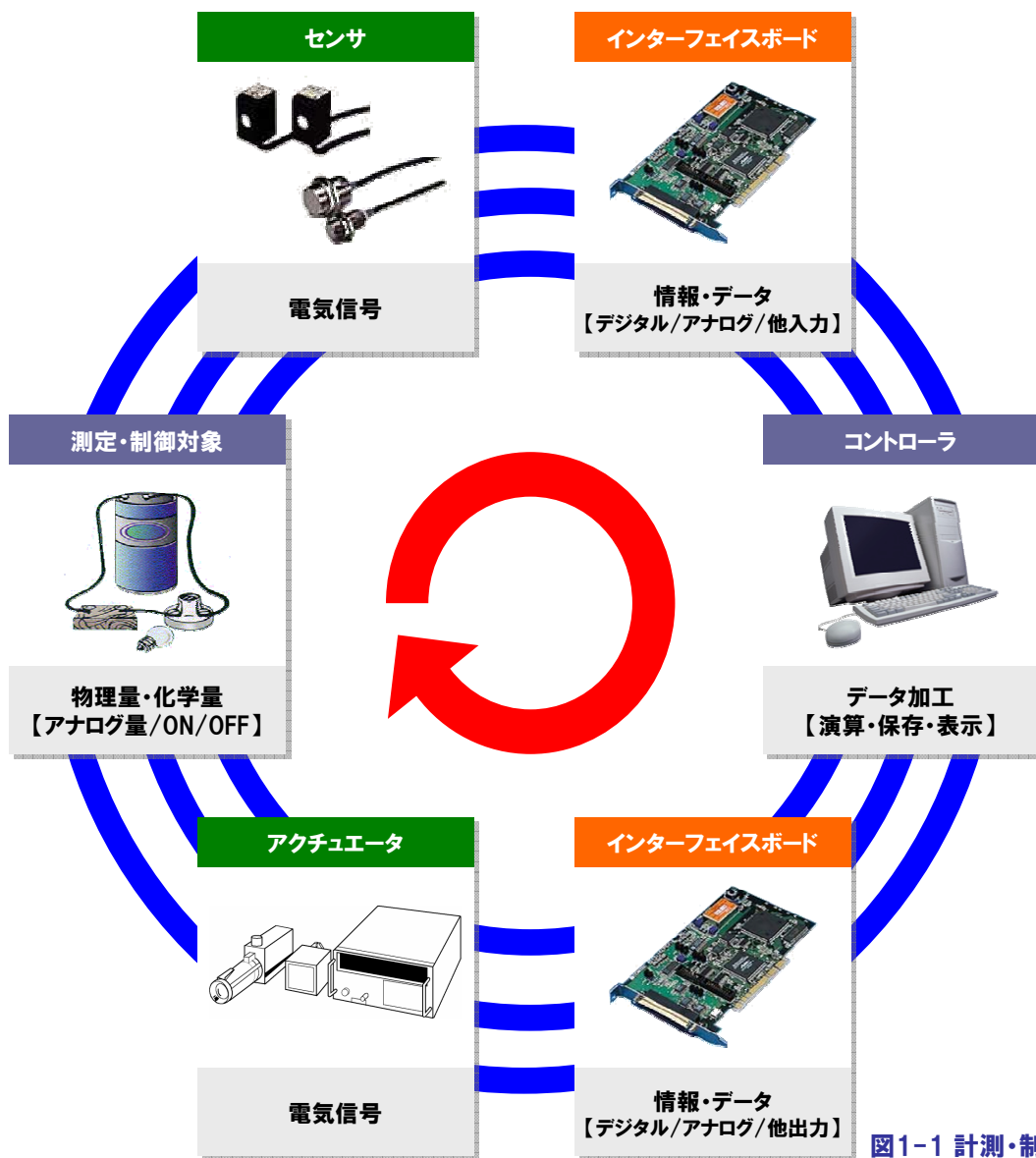
# 第1章

## パソコンによる 計測・制御システム





# 第1章 パソコンによる計測・制御システム



最初に“パソコンによる計測・制御システム”の概要を説明しておきましょう。図1-1は、パソコンによる計測・制御システムの基本的な処理の流れを表したものです。計測・制御システムの第一歩は、測定・制御対象の現在の状態を知る事から始まります。測定・制御対象となるものが温度や圧力、流量などの物理量・化学量である場合は、このままパソコンに取り込み、情報として活用することはできません。こうした物理量・化学量をパソコンに取り込むには『センサ』や『計測器』を使用します。『センサ』とは、測定・制御対象である物理量・化学量を電気信号に変換する装置の総称で、センサが出力する電気信号はアナログ/デジタルなど様々です。これら電気信号をパソコンに入力して情報(データ)化する機器が計測・制御用インターフェイスであり、主にアナログ入力、デジタル入力、カウンタ、 GPIB通信、シリアル通信などがあります。

そして、計測・制御システムの核であるコントローラは、入力した情報をもとに演算、画面表示、解析などを行い『アクチュエータ』へ必要な制御指令を出します。『アクチュエータ』とは、電気信号による制御指令情報を実際の動作に変換する装置の総称で、センサと同様にインターフェイスはアナログ/デジタルなど様々です。これらのインターフェイスに合わせて、アナログ出力、デジタル出力、 GPIB通信、シリアル通信などの計測・制御用インターフェイスを使用し、パソコンで制御します。計測・制御システムの流れは、皆さんの身近なところで見ることができます。例えば、エアコンの動作を考えてみてください。エアコンを単純化したモデルにしてみると、温度センサで現在の室温を測定しており、設定温度より低ければ暖めようとし、高ければ冷やそうとします。設定温度と現在の室温との開きが大きければ風量をあげ、設定温度に近づくとつれて風量を下げる。このようにしてエアコンは、室内を快適な状態に保ってくれるわけですが、これはまさに計測・制御システムの基本的な処理の流れそのものなのです。

## 1-1. センサの主な種類と出力信号例

センサの選定ポイントとしては、何を計測するか、測定可能範囲（目的の範囲を測定できるか）、精度（目的の精度まで測定できるか）、出力信号の電氣的仕様（出力信号の範囲、電圧出力、電流出力など）などがあげられます。

センサの種類	測定可能範囲	精度	出力信号
電子式温度計	0 ~ 80 ℃	±0.3℃	4 ~ 20 mA (DC)
トルク計	0 ~ 200 N・m	±0.15% F.S.	±5VDC
	0 ~ 200 N・m	BCD 5桁	BCDコード 絶縁オープンコレクタ
圧力センサ	0 ~ 500 kgf/cm <sup>2</sup>	±2% F.S	0.5 ~ 4.5VDC
変位センサ	0.6 ~ 2.6 mm	±2% F.S.	0 ~ 2VDC
電力トランデュース	0 ~ 1500 W	±2% F.S.	0 ~ 5VDC

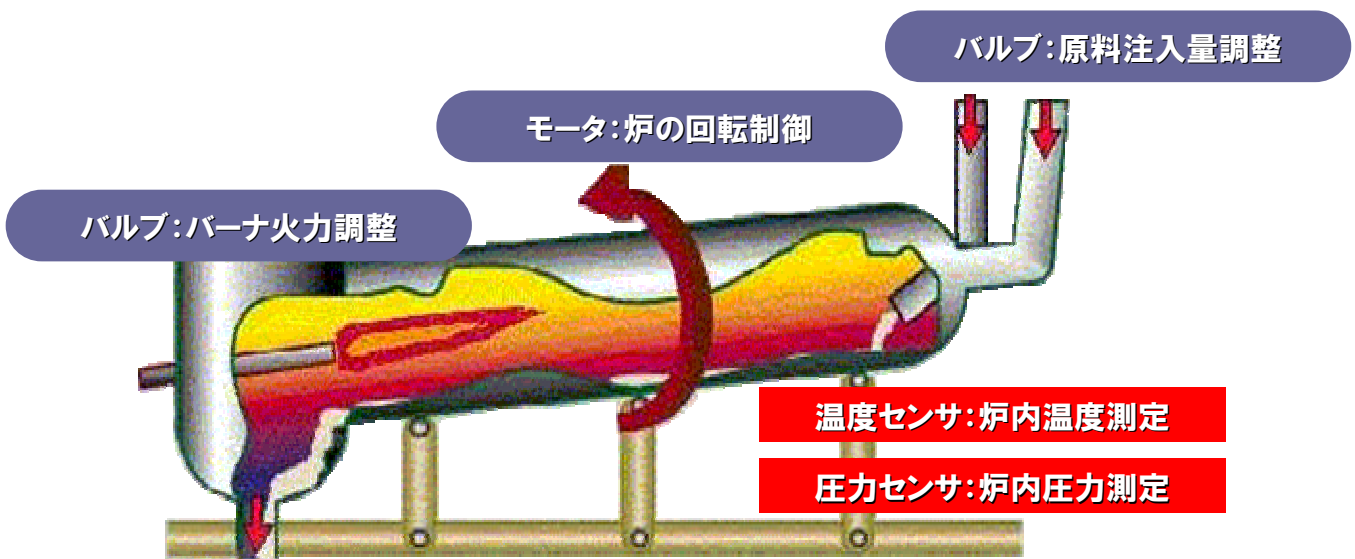
## 1-2. アクチュエータの主な種類と入力信号例

アクチュエータの選定ポイントとしては、何を制御するか、制御可能範囲（目的の範囲を制御できるか）、精度（目的の精度で制御できるか）、入力信号の電氣的仕様（入力信号の範囲、電圧入力、電流入力など）などがあげられます。

アクチュエータの種類	制御可能範囲	精度	入力信号
リニアモーション	0.02 mm/Step	±5 %以下	フォトカプラ絶縁入力 (パルス)
ステッピングモータ	0.72° /Step	±0.05° 以下	フォトカプラ絶縁入力 (パルス)
サーボモータ	4000 r/min	±0.2 %以下	0 ~ 10VDC
バルブ	0 ~ 100 %	±5 %以下	4 ~ 20mA
リレー	ON / OFF	—	12 ~ 24VDC

## 1-3. システム事例: 溶解炉監視制御システム

溶解炉に流れ込む原料の量を調節しながら炉を適度に回転／加熱させ化合物を生産する完全自動化システムです。このようなシステムには、多種多様なセンサ・アクチュエータが不可欠です。



## 1-4. 計測・制御用インターフェイスの種類

前述のセンサや計測器、アクチュエータなどの外部装置とパソコンとのインターフェイスには、次のような種類があり、接続する外部装置にあわせてボードやカード、モジュールをパソコンに実装してデータのやりとりを行います。

### ●デジタル入力

スイッチなどの接点情報 (ON/OFF) の入力、数値 (BCDやバイナリ) データの平行データ入力など。

### ●デジタル出力

負荷 (LED、モータ、バルブなど) 点灯・駆動/消灯・停止制御、数値データの平行データ出力など。

### ●アナログ入力

各種センサを使用した物理量・化学量の測定、アナログ電圧/電流入力全般。

### ●アナログ出力

サーボモータを使った速度制御・シミュレーション信号の出力、電圧/電流アナログ出力全般。

### ●シリアル通信

RS-232C/422A/485規格の通信ポートを持つ各種装置との接続。

### ●GPIB通信

GP-IB (IEEE-488/488.2) 規格準拠の各種装置との接続。

### ●カウンタ入力

パルスエンコーダを使った回転・移動物の位置検出、流量、電力などの積算、生産パルスの積算全般。

### ●モータコントロール

パルスモータ (ステッピングモータなど)、サーボモータを使用した回転・移動物の位置決め制御。

## 1-5. パソコン拡張バス・スロットの種類

現在パソコンには、さまざまな拡張バスが搭載されています。パソコン本体と外部装置との距離や必要なスピード、システム規模・用途に応じて使用する拡張バスを選択します。

### ●PCIバス

インテルが内部ローカルバス (データ伝送路) として提唱し、PCISIGにより外部バスとして標準規格となった、デスクトップ型パソコンの標準バスです。バス幅32ビット、動作周波数33MHz、最大データ転送速度133MB/秒が主流で、最新の規格ではバス幅64ビット、動作周波数66MHz、最大転送速度533MB/秒の仕様もあります。

### ●Low Profile PCI

1999年にPCISIGが策定した小型のPCIボード規格です。省スペースパソコンで問題となる拡張性を解決するため策定されました。現在、省スペース型パソコンの多くに搭載されています。

### ●PCカード (PCMCIA)

PCMCIA (パソコンメモ리카ード協議会) とJEIDA (日本電子工業振興協会) が規格化したパソコン用カード型デバイスの規格です。主にノートパソコンや省スペースパソコンの拡張用スロットとして使用されています。

バス幅16ビット、最大転送速度16MB/秒のPCカードに代わり、バス幅32ビット/最大転送速度132MB/秒の『CardBus』に対応した製品・パソコンが現在主流となっています。

### ●USB (Universal Serial Bus)

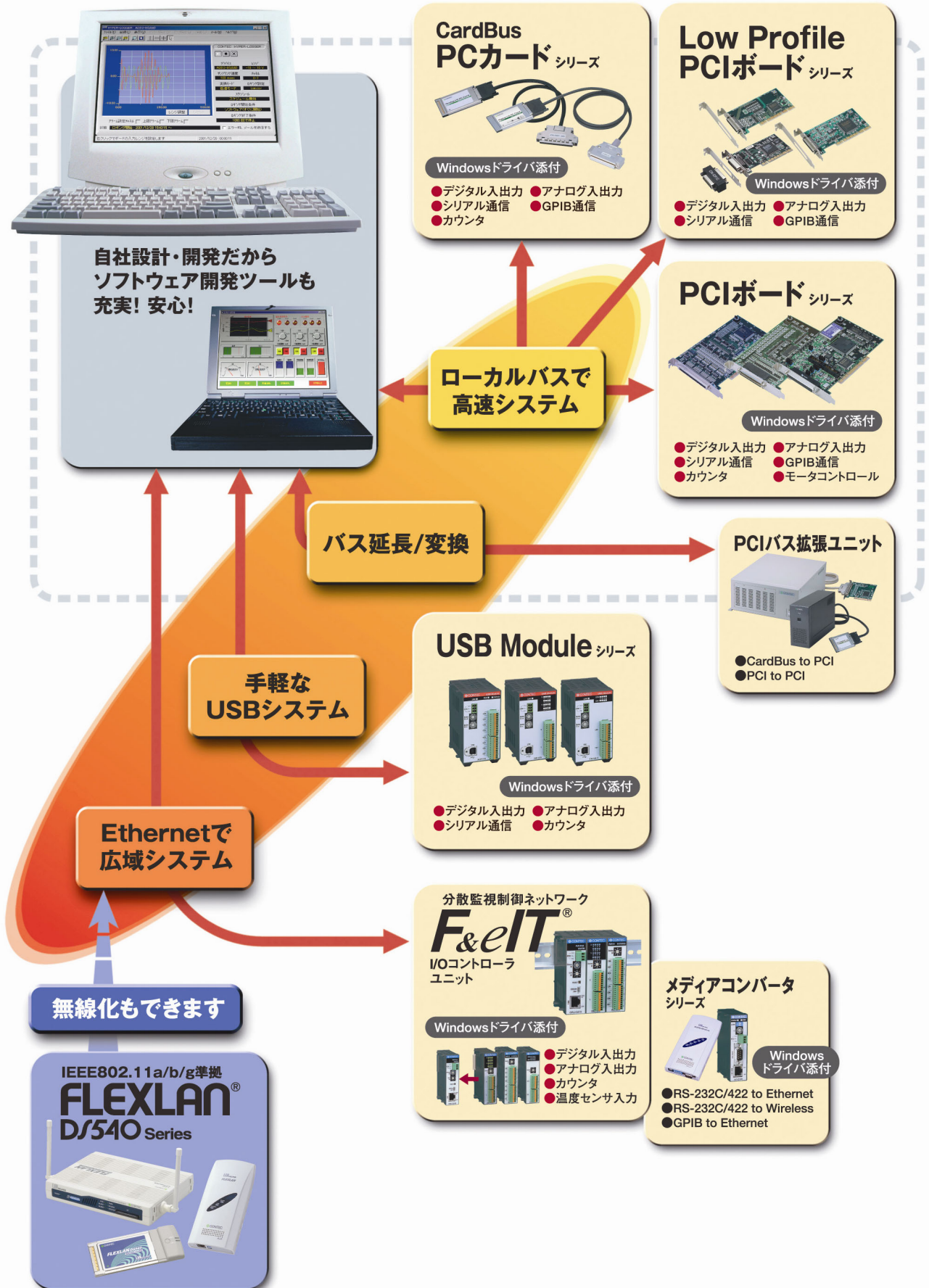
インテル、Microsoft、NECなど7社が中心となり策定した、パソコン用のシリアル・インターフェイスの規格です。USBはUSBハブを介して、パソコンなどのホストに最大127台の機器を接続できるため、拡張バススロット数に制限されない機器の増設が可能です。データ転送速度は、12Mbps (フルスピードモード) と1.5Mbps (ロースピードモード)、最新の「USB2.0」では480Mbps (ハイスピードモード) に対応しています。

### ●LAN (イーサネット)

一般的な拡張バスの定義ではありませんが、広域に点在する設備の集中監視・制御システムなど、既設ネットワークインフラの利用、無線LANとの併用などにより、イーサネット仕様の計測・制御モジュールを使用すれば、イーサネットの汎用性を活かしたローコストでフレキシブルなリモートI/Oシステムが実現します。

# 1-6. コンテックのパソコン計測・制御バリエーション

高速な拡張ボード/PCカードタイプや手軽なUSBタイプ、広域をカバーするEthernetタイプまで、豊富なバリエーションを揃えるコンテックなら、きっとお客様が探している理想的なシステムプランが見つかります。





# 第2章

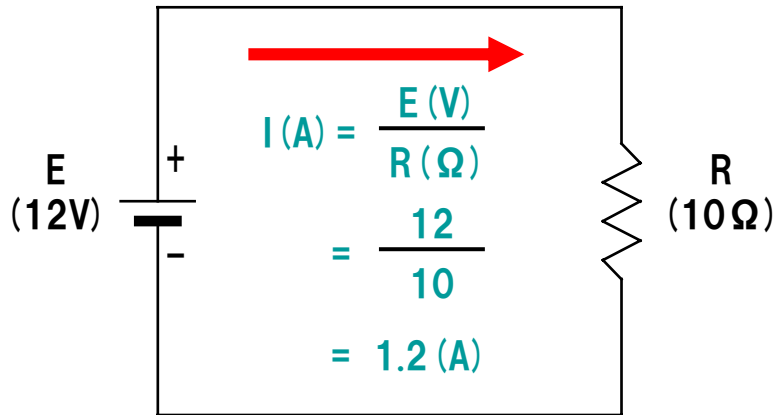
## 電気回路の基礎知識



## 第2章 電気回路の基礎知識

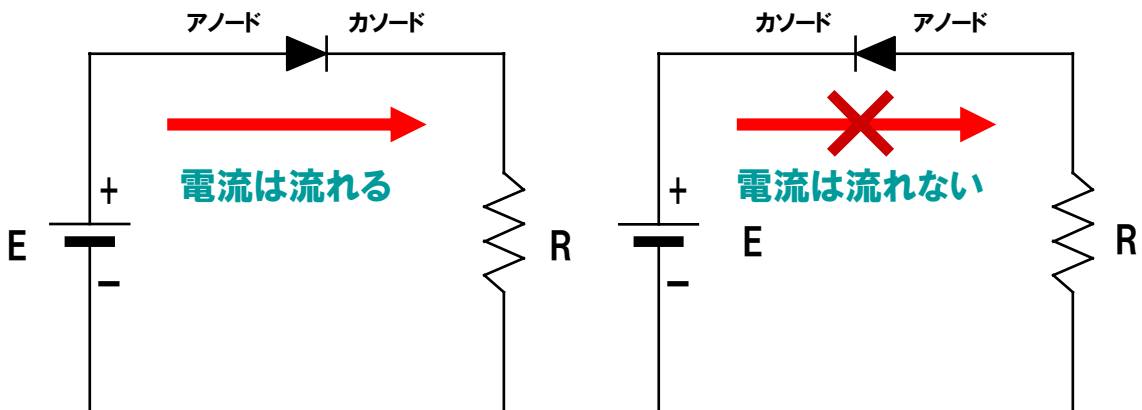
計測・制御システムを構築する場合、スイッチなどの検出装置、パルスモータや表示器などと接続するデジタル回路の設計が不可欠となります。外部機器との接続は、高度な電気的な知識がなければ難しいと言われていますが、これから説明する基礎的な知識で、たいいていの装置と接続することができるようになります。

### 2-1. 電流は電圧の高い方から低い方に流れる



オームの法則:  $I \text{ (A: 電流)} = E \text{ (V: 電圧)} / R \text{ (}\Omega\text{: 抵抗)}$

### 2-2. ダイオードの性質

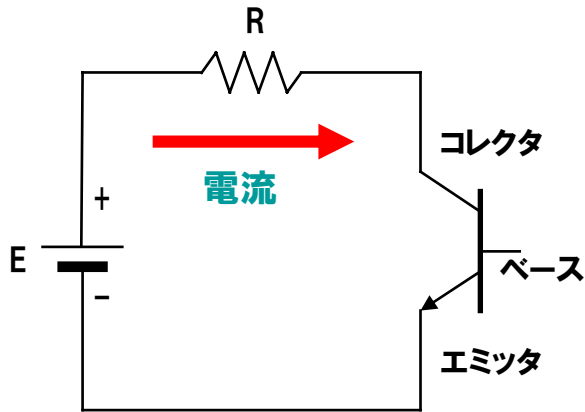


電流は『アノード』から『カソード』の方向にしか流れません。『カソード』側より『アノード』側のほうが電圧が高くないと電流は流れません。

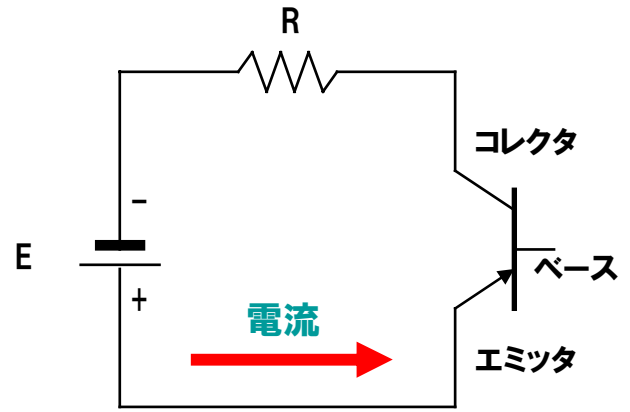
## 2-3. トランジスタの性質

### npnトランジスタ

弊社製デジタル出力基板のオープンコレクタ出力で使用されるトランジスタです。

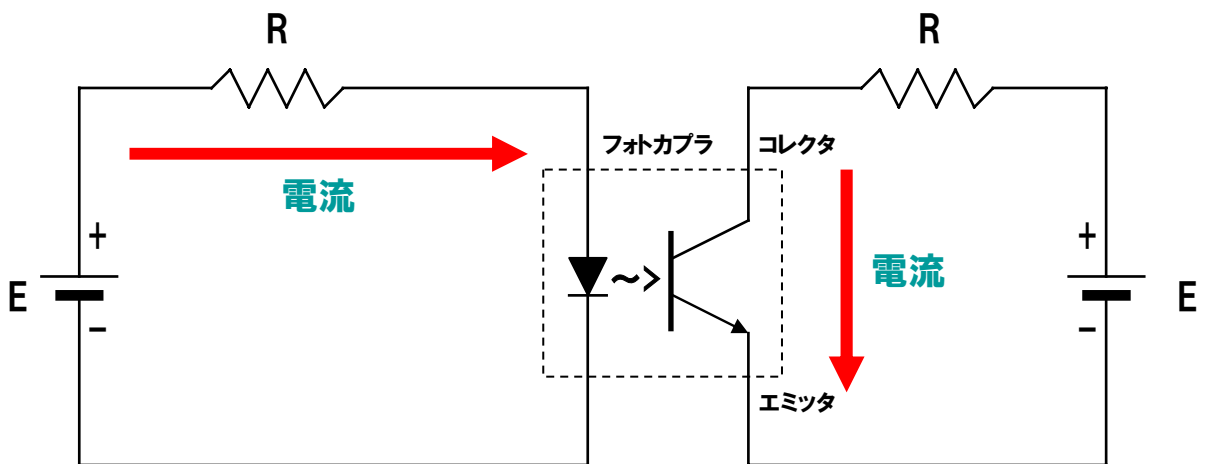


### pnpトランジスタ



『ベース』に電流が流れるとON状態になり『コレクタ』『エミッタ』間に電流が流れます。電流は『エミッタ』の矢印の方向にしか流れません。

## 2-4. フォトカプラの性質



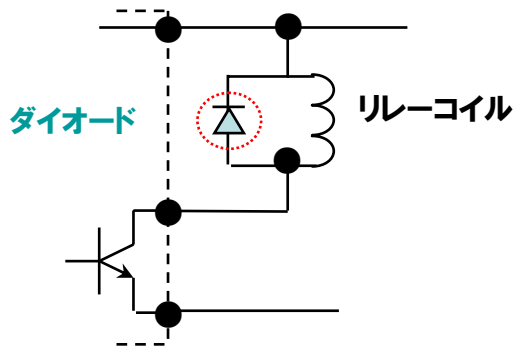
フォトカプラは発光ダイオードとフォトトランジスタを対にした素子です。発光ダイオードは電流が流れると発光します。この光をフォトトランジスタが受けると、フォトトランジスタが『ON』になり電流が流れます。



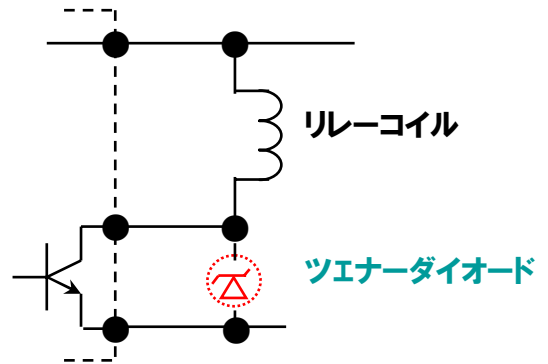
## 2-5. 誘導負荷のサージ電圧対策

リレーコイルを急速遮断すると、急激な高電圧パルスが発生します。また、白熱電球はフィラメントが温まるまでの間、多くの電流（突入電流）が流れます。これらの現象に起因し、出力回路（トランジスタ）の劣化や、破損に至ることがあります。破損に至らなくても、ノイズとしてロジック回路に飛び込み、ボードの誤動作やパソコンが誤動作（暴走）してしまう可能性もあります。リレーコイルのような誘導負荷や白熱電球のようにサージ電圧や突入電流が発生する負荷を接続する場合は、出力段の破損防止やノイズによる誤動作防止のため以下の様な保護対策をとるのが一般的です。

### リレーコイル使用例

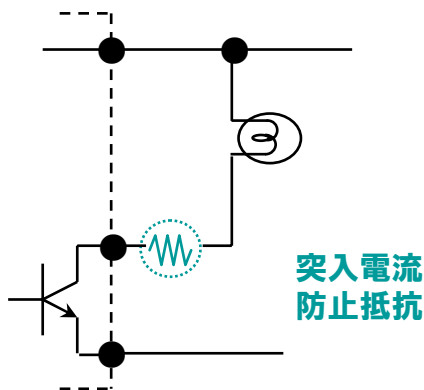


リレーコイルと並列に、サージ吸収用ダイオードを挿入し、サージを吸収します。

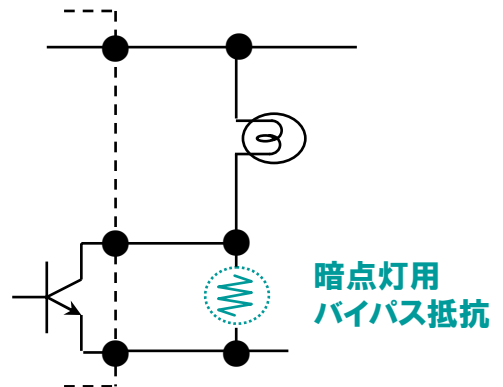


出力トランジスタのコレクタ・エミッタ間に電圧保護用ツェナーダイオードを挿入、コイルのサージをバイパスさせます。

### ランプ使用例



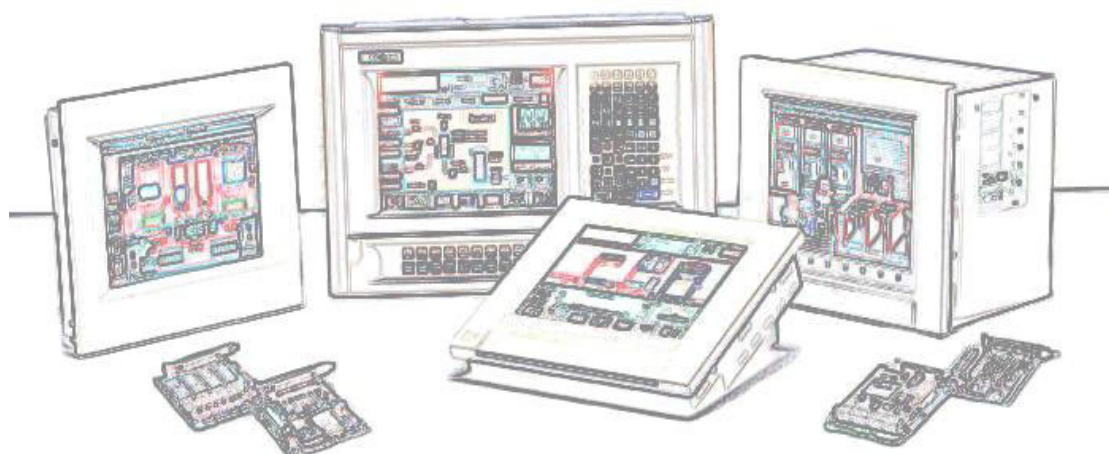
ランプと直列に突入電流防止用として、電流制限抵抗を挿入します。



ランプと直列に暗点灯用にバイパス抵抗を挿入し、常時微電流を流して突入電流を防止します。

# 第3章

## デジタル入出力の基礎知識



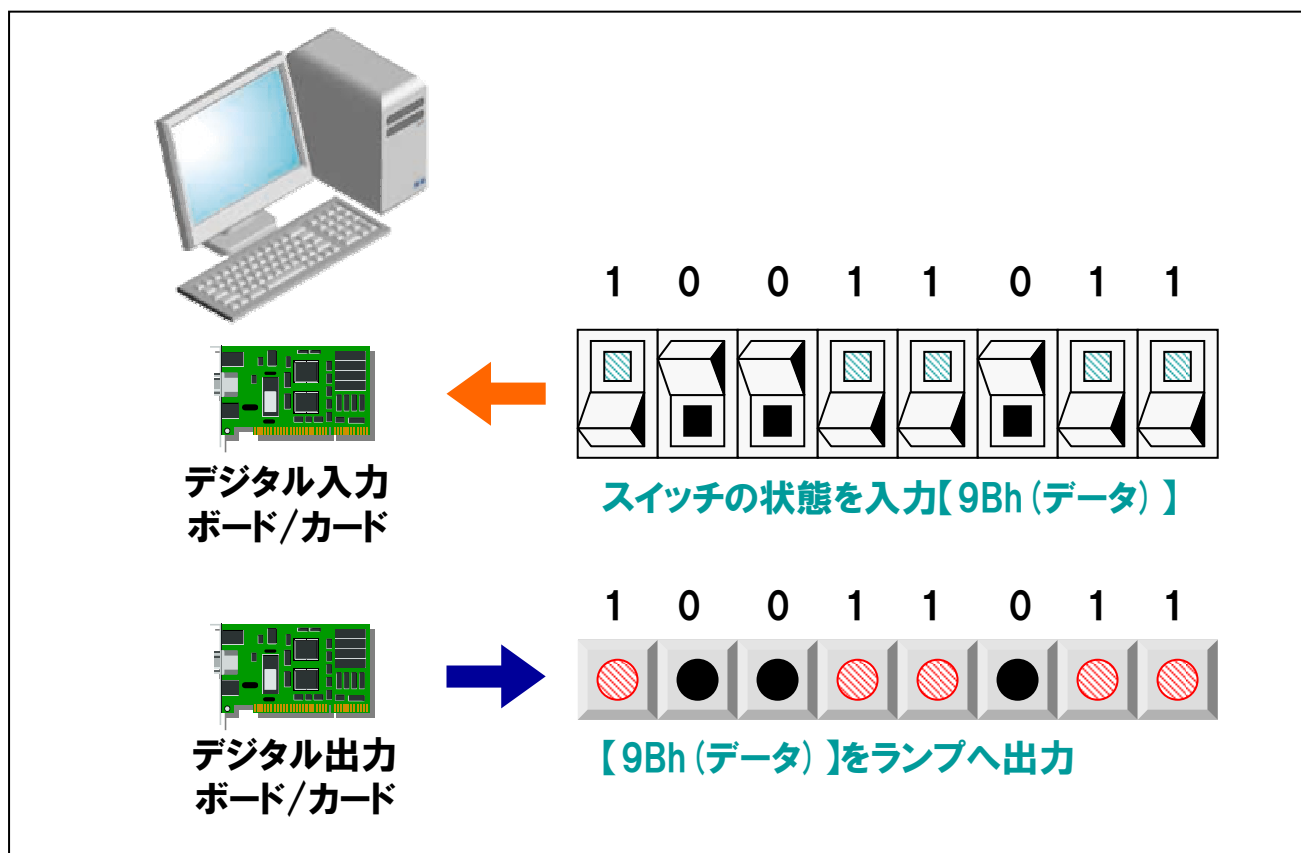
## 第3章 デジタル入出力の基礎知識

### 3-1. デジタル入出力とは

ご存知のとおりコンピュータは『1 (ON)』か『0 (OFF)』かの情報、すなわちデジタル信号で動作しています。私達の身の回りにもデジタル信号は多く存在しています。例えばスイッチの『入/切』やドアの『開/閉』、ランプの『点灯/消灯』などです。これらはすべて『1 (ON)』か『0 (OFF)』で表現することができます。

このデジタル信号をパソコンに入力するインターフェイスが『デジタル入力』であり、逆にコンピュータからデジタル信号を出力するのが『デジタル出力』です。これらは、シリアル通信(1ビット毎の送信)と異なり、多点の情報を一度に入出力することが可能です。そのため、デジタル入出力をパラレル入出力と呼ぶこともあります。

パソコンにデジタル信号をパラレル(並列)に入力・出力する機能を増設するためのインターフェイスボードをデジタル入出力ボードと呼んでいます。この、デジタル入出力ボードを使えば、各種制御回路のリレーや操作スイッチの状態を監視したり、各種表示器やスイッチ、リレーなどの制御を行うことができます。



### 3-2. デジタル入出力ボード/カードの分類

#### ●デジタル入力(パラレル入力)・・・Digital Input

外部装置のスイッチの状態がONかOFFかという情報をパソコンに入力するためのボード/カードです。

#### ●デジタル出力(パラレル出力)・・・Digital Output

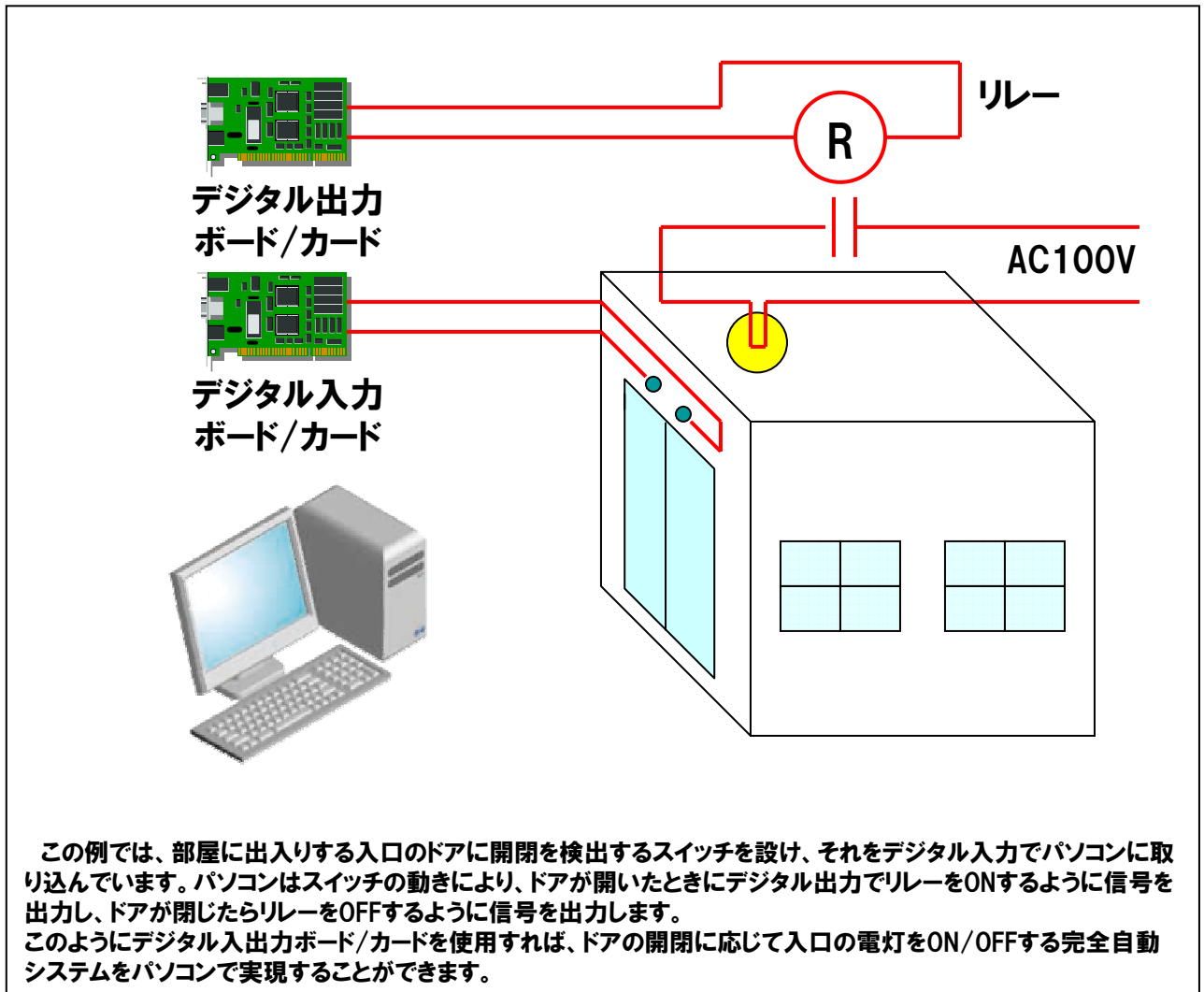
外部装置に対してONかOFFかの指令をパソコンから行うことのできるボード/カードです。

#### ●デジタル入出力ボード(パラレル入出力ボード)・・・Digital Input/Output

デジタル入力とデジタル出力の両方の機能を持ったボード/カードです。

### 3-3. デジタル入出力の適応例

例えば、デジタル入力ボード/カードによってドアの開閉状態を調べたり、デジタル出力ボード/カードによって部屋の明かりをつけたり消したりすることができます。さらにデジタル入出力ボードを使うと、部屋のドアが開いたら明かりを点灯させ、閉じたら消灯するといったことをパソコンで制御することが可能になります。

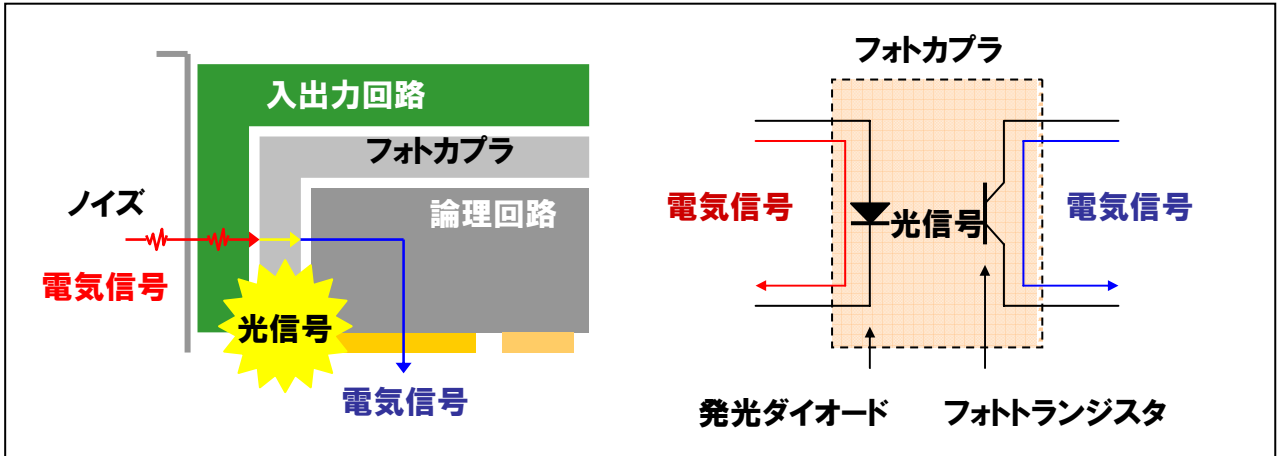


### 3-4. デジタル入出力ボードの種類と用途

デジタル入出力ボードは、大別すると『絶縁型』と『非絶縁型』に分類することができます。それぞれの特長と用途、機能比較を解説していきます。

#### 3-4-1. フォトカプラ絶縁入出力タイプ

ボード内部の論理回路と入出力回路がフォトカプラで絶縁されているため、操作回路に発生した電気的外乱の侵入を防ぐことができます。ただし、フォトカプラを駆動するために外部からDC電源の供給が必要となります。デジタルスイッチや表示器など、操作回路がDC12～48Vの弱電機器との接続に使用します。



#### 3-4-2. 高速フォトカプラ絶縁入出力タイプ

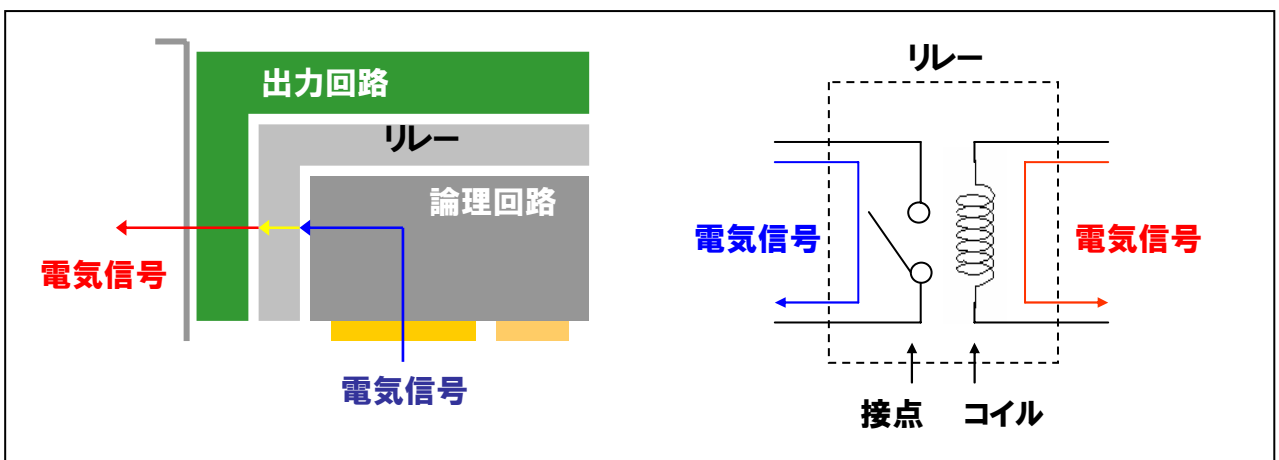
絶縁素子に高速フォトカプラを使用して高速化したタイプです。

#### 3-4-3. 電源内蔵フォトカプラ絶縁入出力タイプ

内部の論理回路とは絶縁されたDC電源を搭載したタイプです。フォトカプラの駆動と操作回路への電源供給が可能のため、外部電源を用意できない場合に便利です。

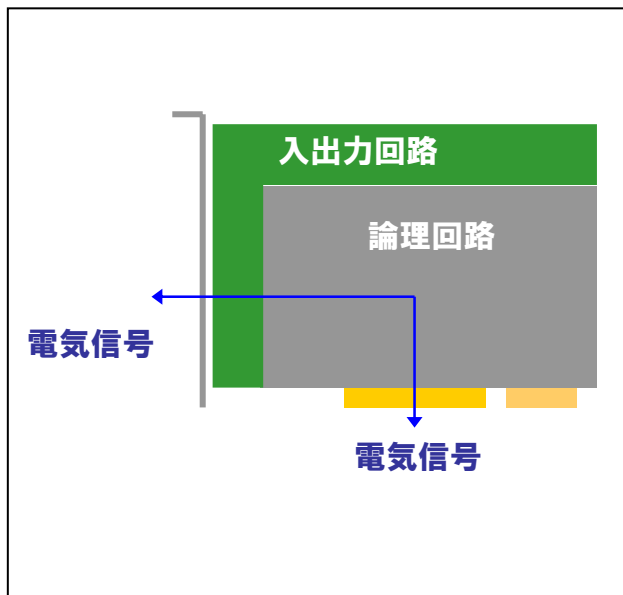
#### 3-4-4. 接点出力タイプ

出力回路に接点リレーを使用しているタイプです。ボード内部の論理回路と出力回路を接点リレーにより絶縁しています。電流を流す方向に制限がないため、AC (交流) 負荷も接続可能です。操作回路がAC、またはDC48Vを超える強電機器を直接制御する場合に使用します。



### 3-4-5.非絶縁入出力タイプ

ボード内部の論理回路と入出力回路を絶縁せずに直結したタイプです。フォトブラ絶縁入出力タイプに比べて高速な応答が可能です。ただし、電気的外乱の影響を受けやすいため、ノイズ環境が良好で配線距離が短い場合にのみ使用できます。半導体リレーやコントローラなど、入出力回路がDC5VやTTLの機器との接続に使用します。



#### TOPICS

「バッファ(ドライブ回路)付きボードとは・・・」

ドライブ回路を実装することにより、スタンダードなタイプと比較して、より大きな電流を外部に流せるようになっているタイプです。『スタンダードタイプ』の最大定格が、『DC5V 1mA』に対して、「バッファ付き」は、『DC5V 24mA(弊社製品比)』となります。

#### TOPICS

「i8255とは・・・」

i8255は、プログラマブル汎用I/Oデバイスです。3組(8ビット)の入出力ポートがあり、プログラムによる制御でデータ入力、データ出力、ステータス信号入力、コントロール信号出力に使用することができます。

### 3-4-6.双方向入出力タイプ

ボード内部の論理回路とi8255または同等の双方向入出力回路を絶縁せずに直結したタイプです。入出力点数(方向)をプログラムで8点単位に変更可可能です。入出力回路がTTL(DC5V)で、双方向の入出力が必要な機器との接続に使用します。

## 3-5. 絶縁タイプと非絶縁タイプとの機能比較

項目	絶縁タイプ	非絶縁タイプ
主な動作電圧	DC12~48V、AC/DC100V	DC5V、DC3.3V
絶縁性	あり	なし
応答速度	1m~5μsec程度	200nsec程度
信号延長可能距離	50m程度	1.5m程度
主な接続対象	スイッチ、リレー接点の入力 リレー、ランプへの出力	小型計測器 (BCD、バイナリデータ)
主用途	制御	計測

#### TOPICS

「TTL、LVTTTLとは・・・」

TTLは「Transistor Transistor Logic」の略で、トランジスタを用いた論理回路のことを示します。一般的な駆動電圧は0~5Vの範囲です。LVTTTL(Low Voltage Transistor Transistor Logic)の駆動電圧は0~3.3Vの範囲です。電気回路の世界では、バイポーラ素子を使用した最も一般的に使われている回路です。

TTL/LVTTTLレベルには、閾値があり、『0.8V以下ならば、“Low”』、『2.0V以上ならば、“High”』と認識されます。すなわち、2.0V以上の電圧が入力端子にかかっているならば、『“High”』が入力されていると認識され、入力されている電圧が0.8V以下であれば、『“Low”』が入力されていると認識します。

## 3-6. 出力回路の種類

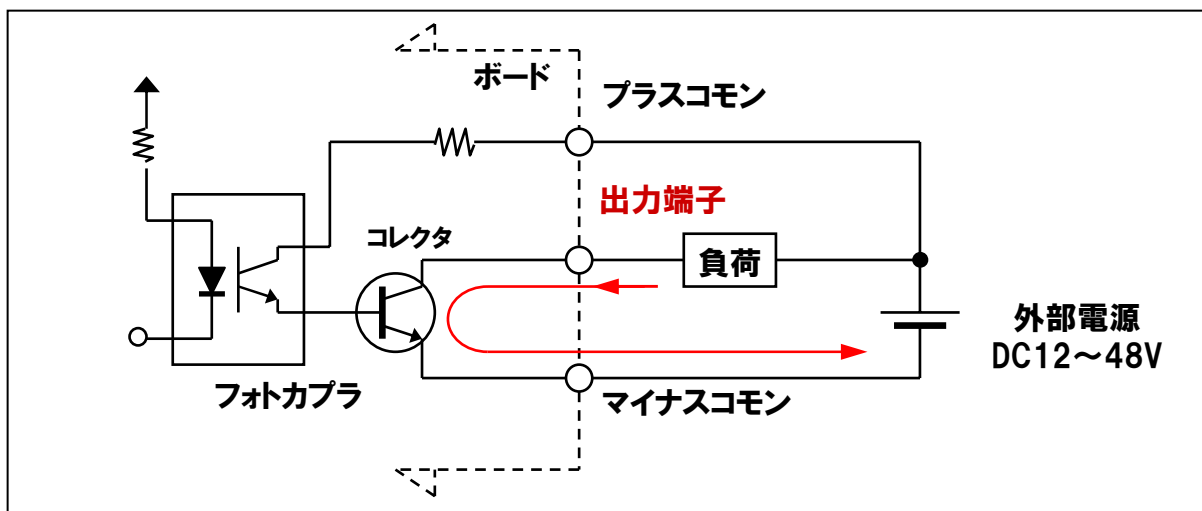
弊社の総合カタログをはじめ、デジタル入出力ボードを選定する際に参考となる仕様表には、必ずと言って良いほど、入力形式・出力形式の欄があります。これらは接続対象により、ボード/カード側の形式を合わせるのが一般的ですが、回路の構造を理解していないとボード/カードの選定や外部機器との接続が困難になってしまいます。ここでは、弊社の製品を例にとって、それぞれの回路構成に関して、解説していきます。

### 3-6-1. トランジスタ出力 (無接点出力)

半導体素子であるトランジスタを使用して、DC負荷を駆動・開閉する出力回路です。接点出力と違い、実接点が無いことから無接点出力とも呼ばれています。

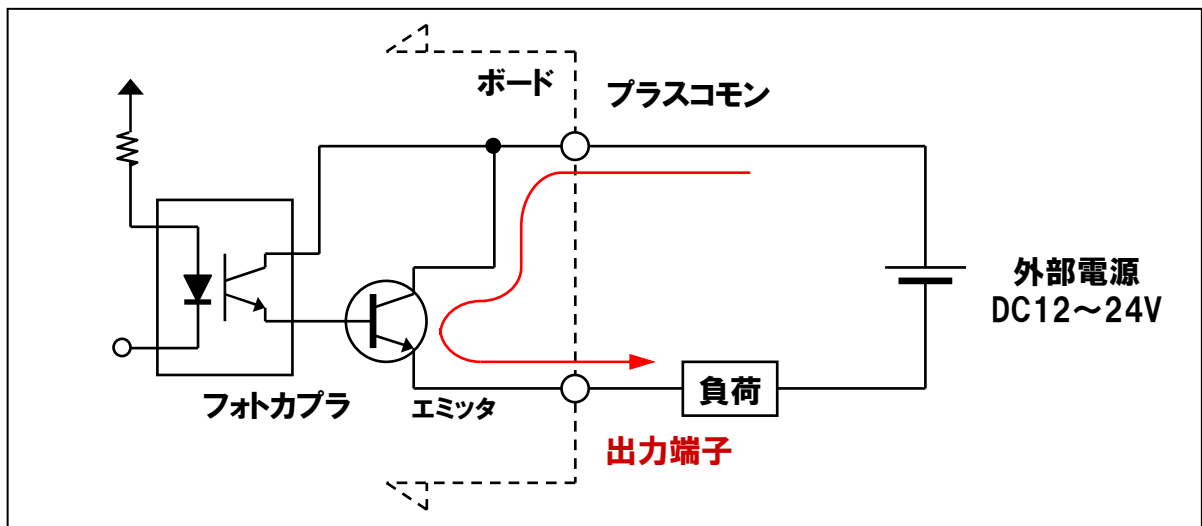
#### ① フォトカプラ絶縁オープンコレクタ出力 (シンクタイプ)

出力トランジスタのコレクタが出力端子となり、開放 (オープン) 状態になっている出力回路です。内部論理は、『ON (短絡) : 1、OFF (開放) : 0』です。出力トランジスタが『ON (負荷を作動)』した際、電流が負荷から出力端子に流れ込む (シンク) タイプです。DC12~48Vの弱電機器への信号出力に使用します。



#### ② フォトカプラ絶縁出力 (ソースタイプ)

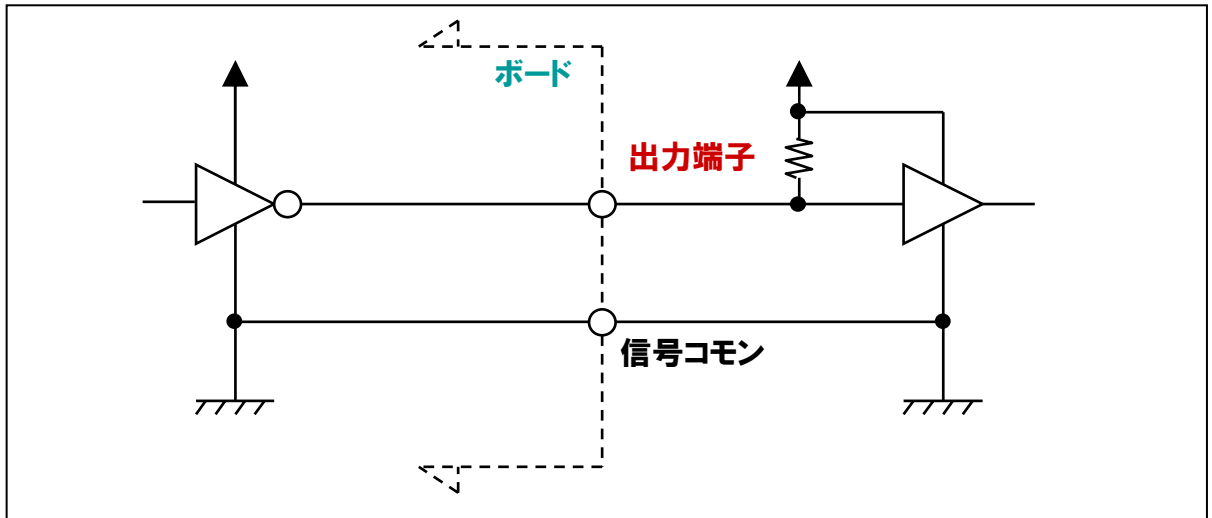
出力トランジスタのエミッタが出力端子になっている出力回路です。内部論理は『ON (短絡) : 1、OFF (開放) : 0』です。出力トランジスタが『ON』した際、電流が出力端子から負荷に向けて流れ出る (ソース) タイプです。DC12~24Vの弱電機器への信号出力に使用します。





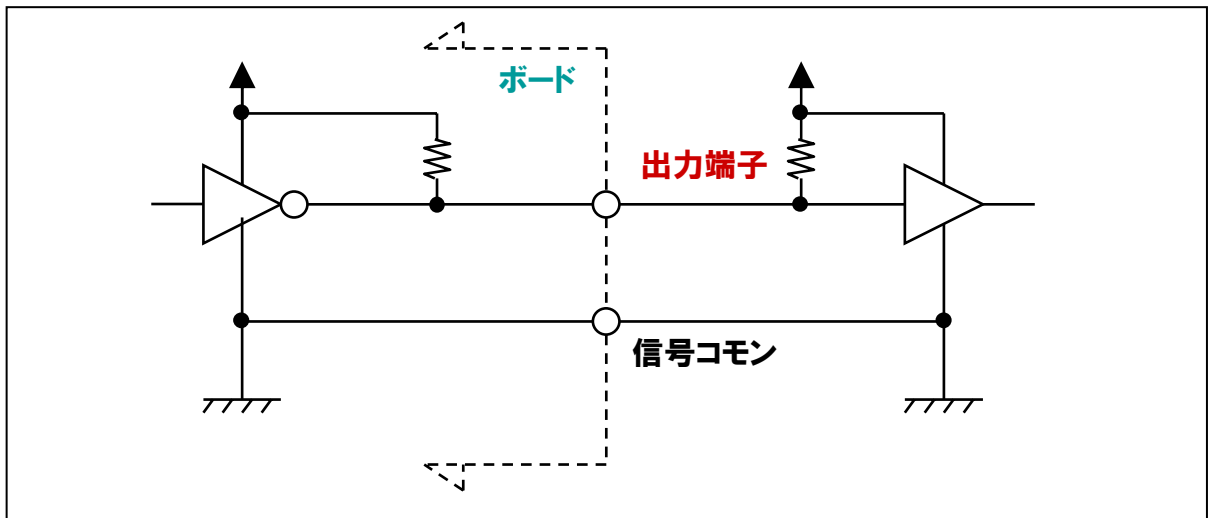
### ③非絶縁TTL オープンコレクタ出力(負論理)

出力トランジスタのコレクタが出力端子となり、開放(オープン)状態の出力回路で、入力回路側でプルアップします。内部論理は「Low:1、High:0」の負論理です。入力回路がTTLの機器やDC5Vの機器への信号出力に使用します。



### ④非絶縁TTL レベルドライバ出力(負論理)

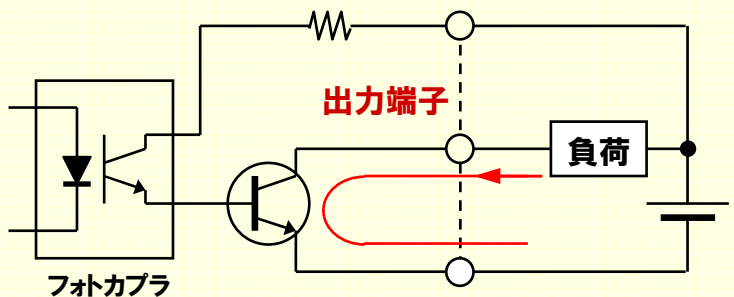
出力トランジスタのコレクタが出力端子となり、パソコンのVCC(DC5V)でプルアップされている出力回路です。内部論理は「Low:1、High:0」の負論理です。入力回路がTTLの機器やDC5Vの機器への信号出力に使用します。



#### TOPICS

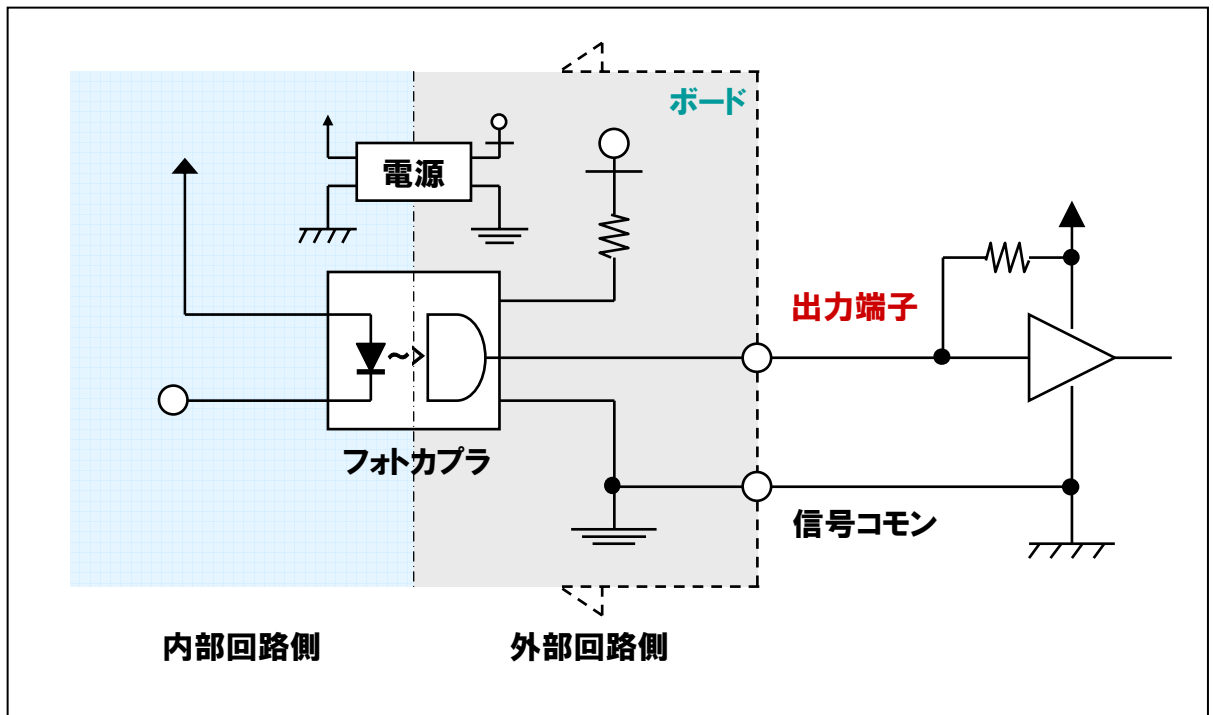
##### 『オープンコレクタ出力とは・・・』

出力トランジスタのコレクタ側を開放(オープン)し、外部機器側から負荷の駆動に必要な電源を供給する方式です。フォトカブラの出力(エミッタ)をドライブ能力の高い出力トランジスタのベースに接続する方法をダーリントン接続と呼んでいます。



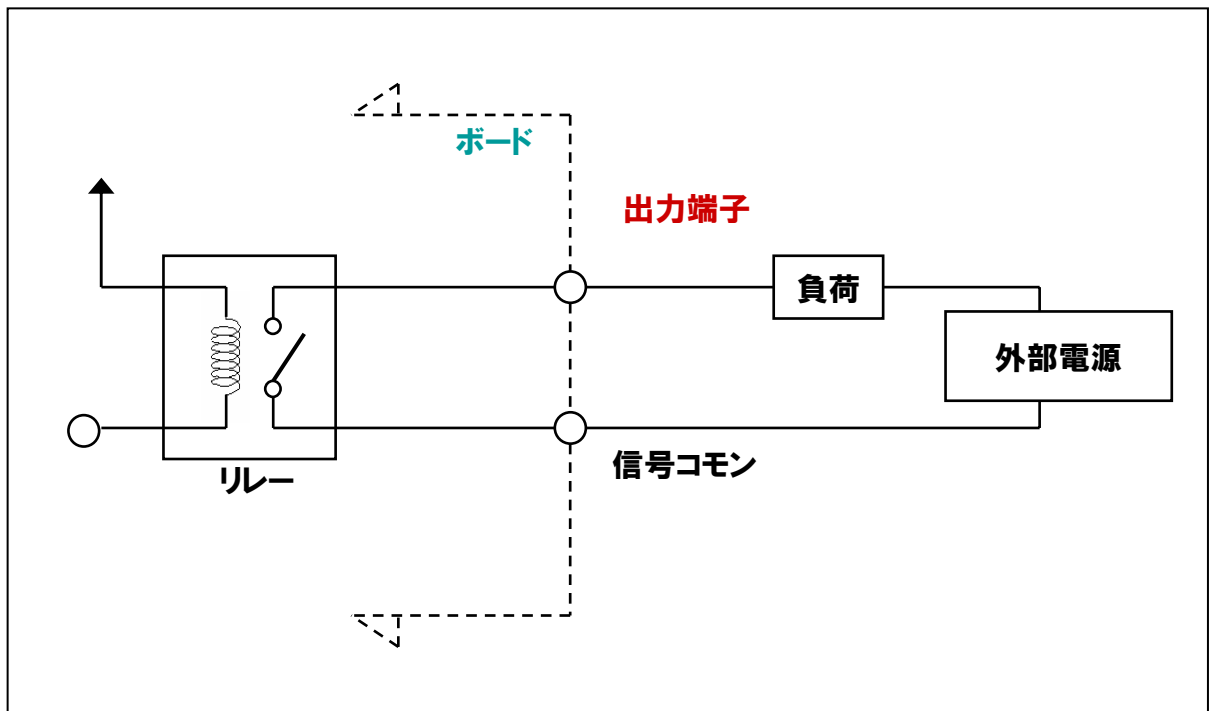
### ⑤フォトカブラ絶縁TTLドライバ出力(負論理)

出力トランジスタのコレクタが出力端子になっている出力回路です。内部論理は『Low:1、High:0』の負論理です。入力回路がTTLの機器やDC5Vの機器で、操作回路の配線が長くなる場合や、絶縁したい場合に使用します。



### 3-6-2.接点出力(有接点出力)

リレー接点を使用して、内部の論理回路と絶縁した出力回路です。実接点により負荷を駆動・開閉することから、有接点出力とも呼ばれています。電流を流す方向に制限がないため、DC(直流)負荷でもAC(交流)負荷でも接続することができます。ACまたは、DC48Vを超える強電機器への信号出力に使用します。



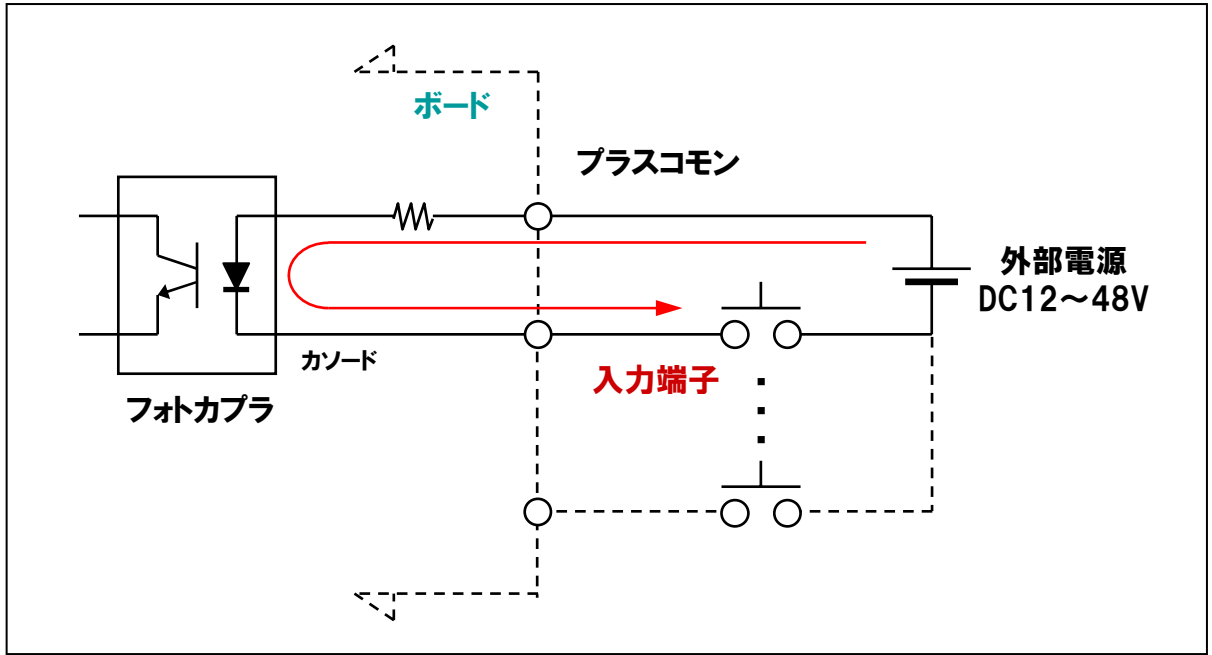
## 3-7. 入力回路の種類

### 3-7-1.DC入力

トランジスタ出力や操作回路がDC(直流)の接点出力回路と接続可能な入力回路です。

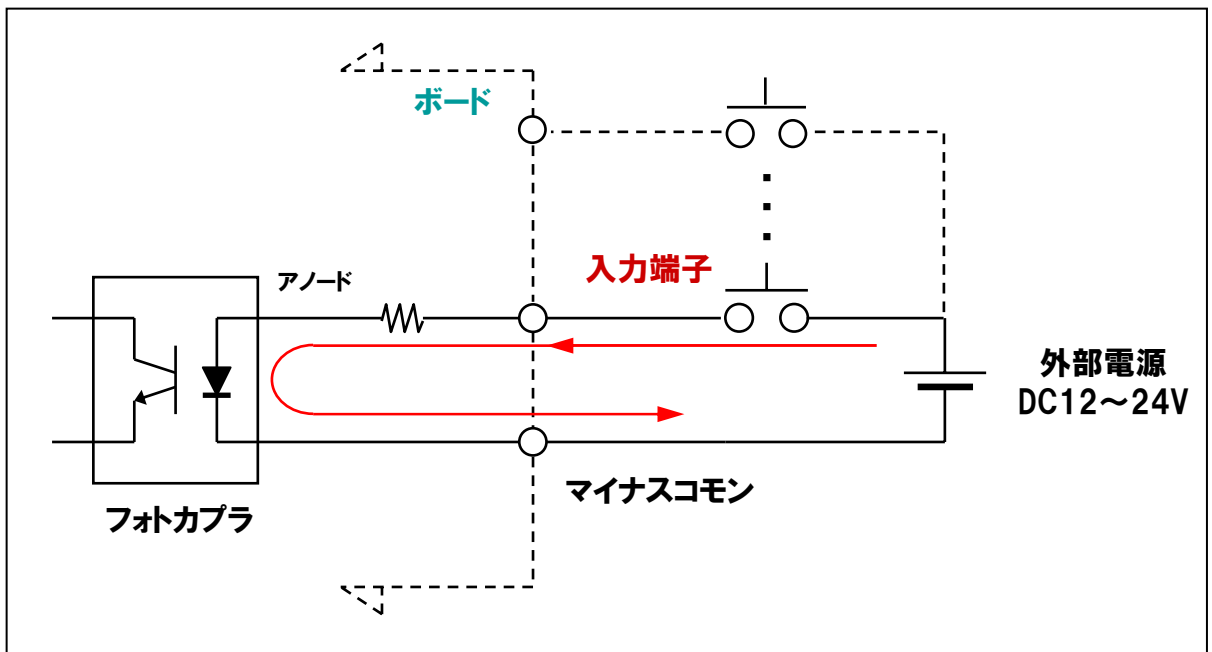
#### ①フォトカプラ絶縁入力(電流シンク出力対応)

フォトカプラのカソード側が入力端子になっている入力回路です。シンクタイプのトランジスタ出力、またはリレースイッチ等からの出力信号を入力します。内部論理は『ON:1、OFF:0』です。入力端子は、電流が流れるソースタイプです。DC12~48Vの弱電機器からの信号入力に使用します。



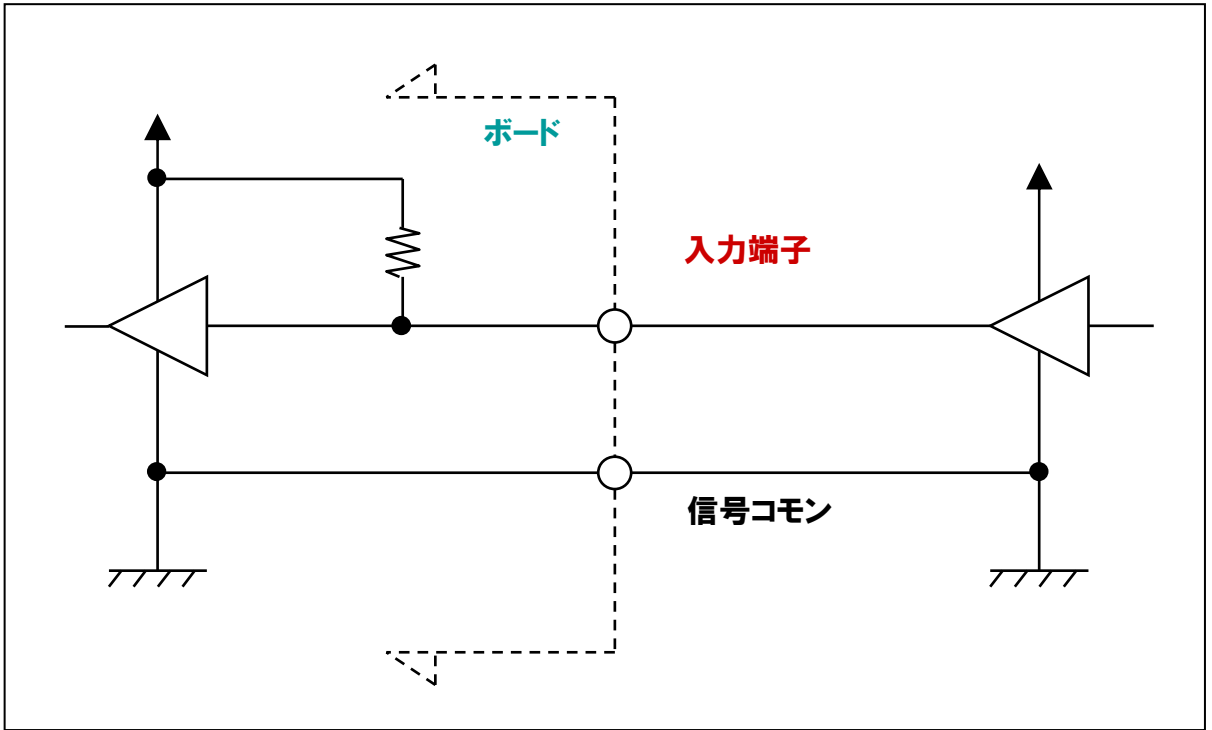
#### ②フォトカプラ絶縁入力(電流ソース出力対応)

フォトカプラのアノード側が入力端子になっている入力回路です。ソースタイプのトランジスタ出力、またはリレースイッチ等からの出力信号を入力します。内部論理は『ON:1、OFF:0』です。入力端子は、電流が流れ込むシンクタイプです。DC12~24Vの弱電機器からの信号入力に使用します。



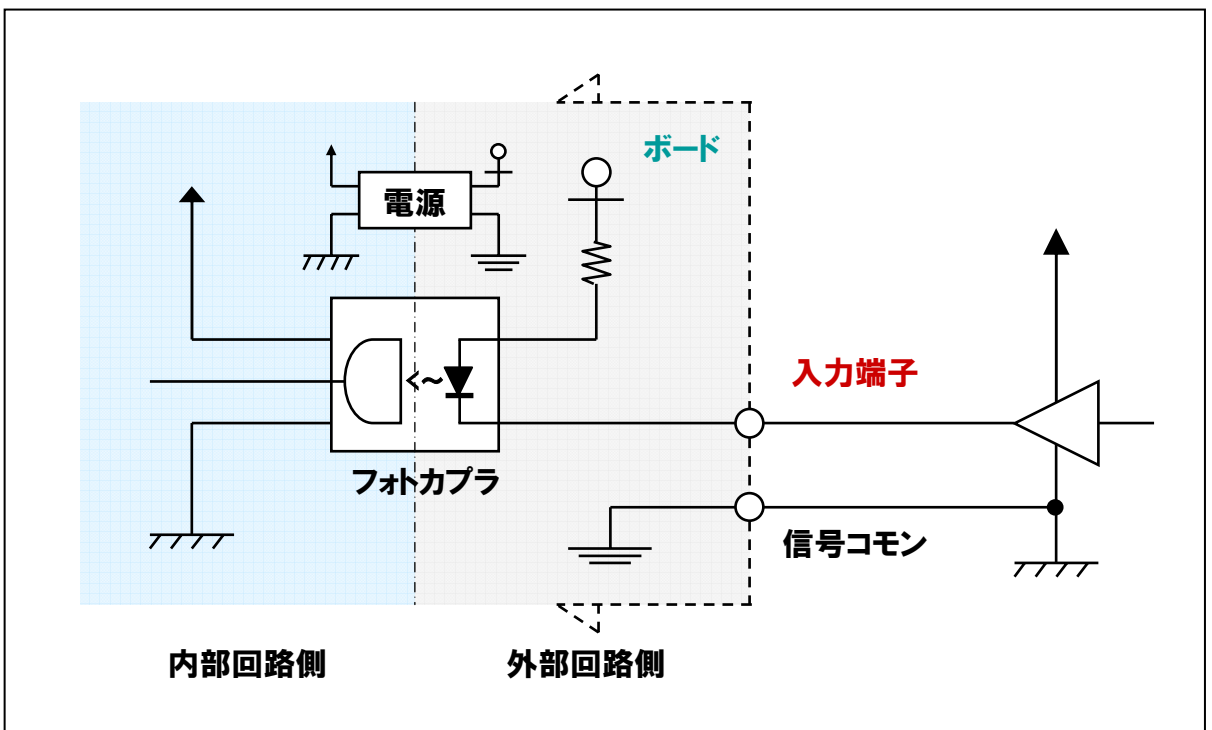
### ③非絶縁TTLレベル入力(負論理)

トランジスタのベースが入力端子になっており、DC5Vでプルアップされた入力回路です。内部論理は、『Low:1、High:0』の負論理です。出力回路がTTLの機器やDC5Vの機器からの信号入力に使用します。



### ④フォトカプラ絶縁TTLレベル入力(負論理)

フォトカプラのカソードが入力端子になっている入力回路です。内部論理は『Low:1、High:0』の負論理です。出力回路がTTLの機器やDC5Vの機器で、操作回路の配線が長くなる場合や、絶縁したい場合に使用します。



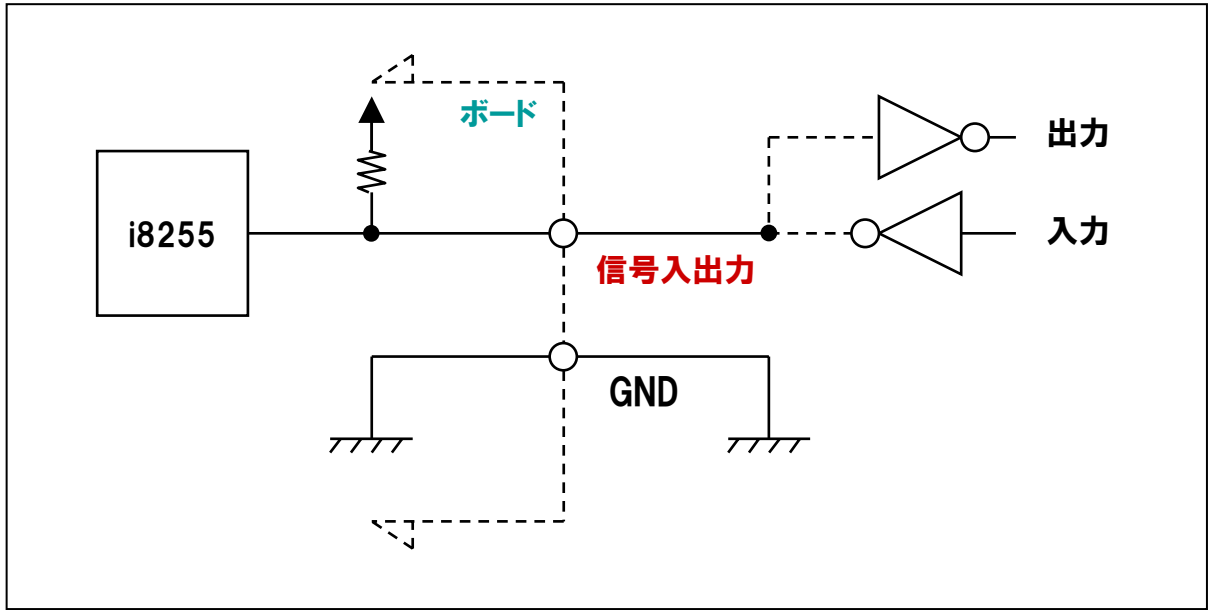
## 3-8. 双方向回路の種類

### 3-8-1. 双方向回路

入出力方向をプログラムで8点単位に変換可能なタイプです。入出力回路がTTL (DC5V) で、双方向の入出力が必要な機器との接続に使用します。

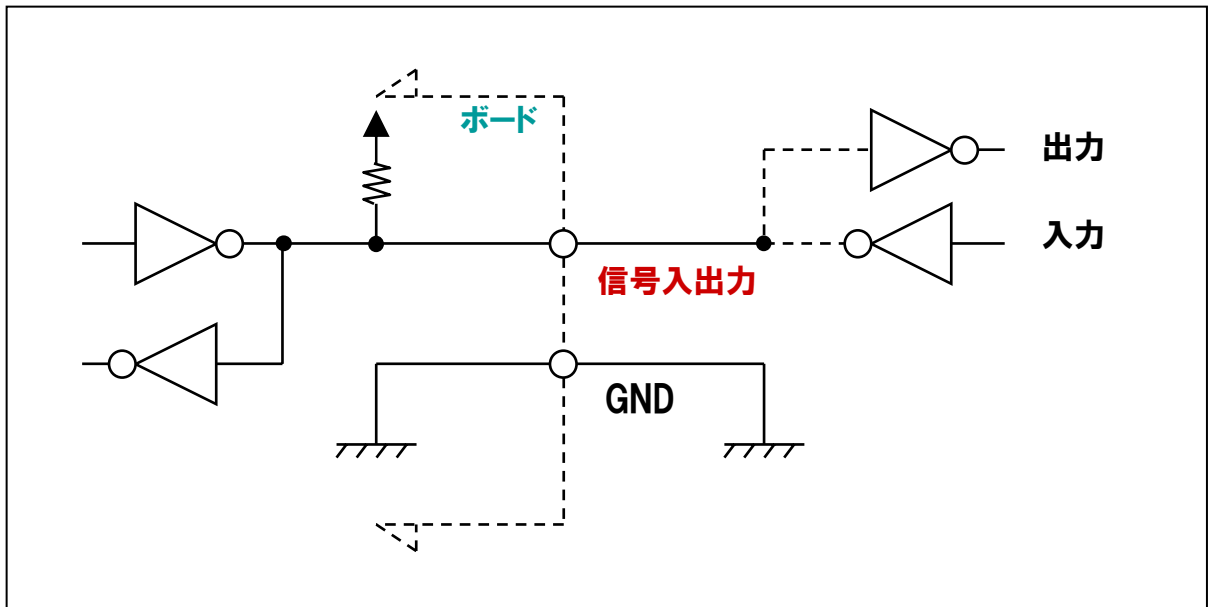
#### ① 非絶縁TTL レベル入出力 (正論理)

i8255 (または相当) の入出力回路です。DC5Vでプルアップしたタイプもあります。内部論理は『Low:0、High:1』の正論理です。入出力回路がTTLの機器やDC5Vの機器、またはTTL (DC5V) の双方向の入出力が必要な機器との信号入出力に使用します。



#### ② バッファ付き非絶縁TTL/LVTTL レベル入出力 (正論理)

i8255 (または相当) にドライブ能力を高めるバッファICを追加し、TTL=DC5V、LVTTL=DC3.3Vでプルアップした入出力回路です。内部論理は『Low:0、High:1』の正論理です。入出力回路がTTLやLVTTLの機器、または双方向入出力が必要な機器との信号入出力に使用します。



## 3-9. 用語解説

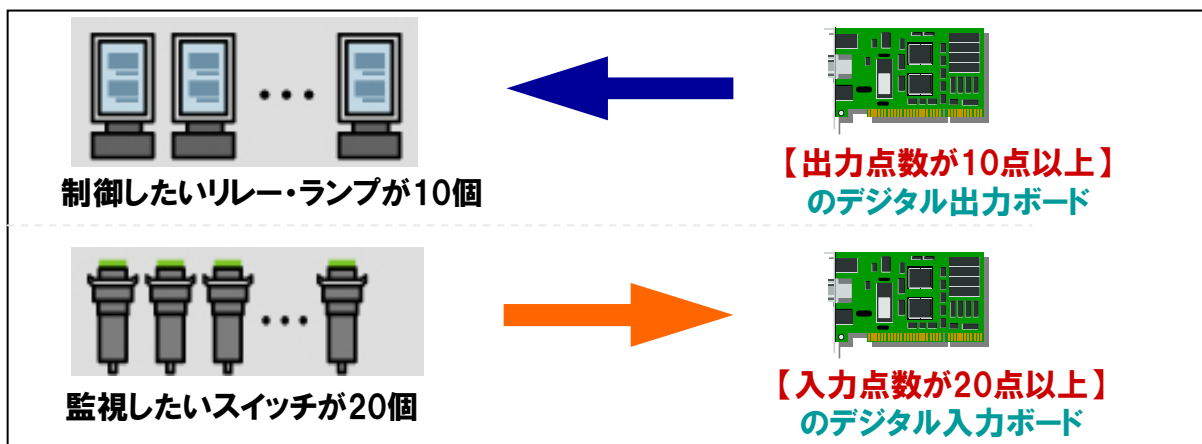
カタログやデータシートを参考にして、デジタル入出力ボード/カードの選定を行う際、仕様の項目や特長に頻繁に出てくるデジタル入出力に関する用語について説明をしていきます。これらの用語を理解することで、システムに最適なデジタル入出力ボード/カードを選定できるようになります。

### 3-9-1. 入出力点数

入出力点数とは、1枚のボード/カードで入力もしくは出力できる点数(ビット数)を表します。あらかじめシステムで使用する点数を調べ、適切な点数を持ったボード/カードを選定します。

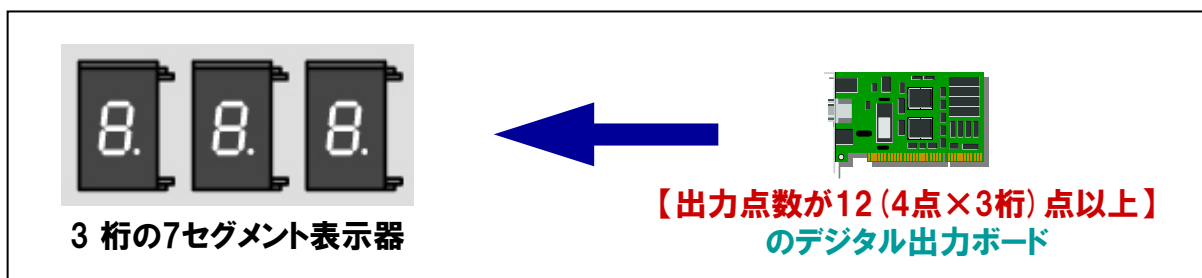
#### 例1:【リレー・ランプの制御/スイッチの監視】

リレーやランプのON/OFF制御やスイッチのON/OFF状態の監視であれば、通常その装置の数が必要な入出力点数(ビット数)となります。ただし、装置によってはアラームやリセット、ハンドシェイクなどの機能を持ったものがあり、装置の数+ $\alpha$ が必要となる装置もあるため、注意が必要です。



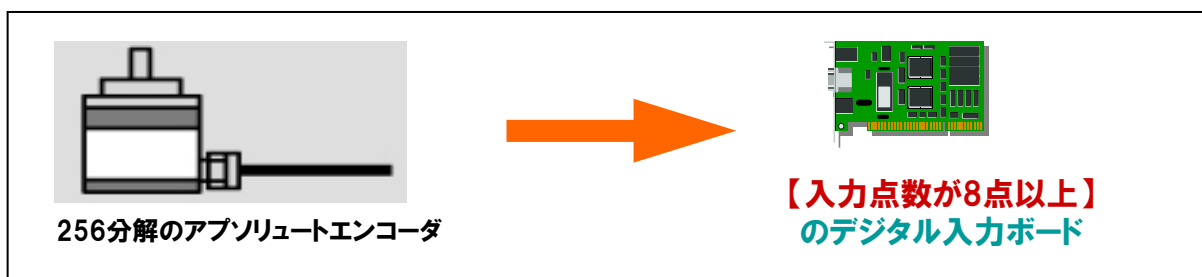
#### 例2:【BCD (Binary Coded Decimal) の7セグメント表示器、デジタルスイッチの場合】

BCD (2進化10進数) とは、10進数を直接2進数で表すのではなく、10進数の各桁を4桁の2進数で表現したものです。そのため、通常BCD1桁につき4ビット(4点)の出力または入力が必要となります。



#### 例3:【バイナリ出力のアブソリュート形のロータリエンコーダの場合】

一般に入力点数は1回転の分解能に関係します。例えば、1回転で256分解するタイプの装置であれば、 $256 = 2^8 \rightarrow 8$ ビット(8点)の入力が必要となります。



### 3-9-2.操作回路電圧

操作回路（設計する回路）に適した電圧、入出力回路の製品を選定します。

**例1:【装置の出力回路がオープンコレクタ、入力回路がフォトカプラ絶縁、操作回路電圧が24VDCの場合】**

→ 入力回路がフォトカプラ絶縁で、出力回路がフォトカプラ絶縁オープンコレクタ出力タイプのボードを選定します（弊社型式:PIO-32/32L (PCI) Hなど）。

**例2:【装置の入出力がTTLレベルで高速通信が必要な場合】**

→ 入出力回路が非絶縁TTLレベルのタイプが適しています（弊社型式:PIO-32/32T (PCI) Hなど）。  
→ 絶縁が必要な場合には、絶縁型TTLタイプを選定します（弊社型式:PIO-16/16TB (PCI) Hなど）。

### 3-9-3.コモン構成

デジタル入出力ボードの入出力回路は何点かの単位で電氣的にグループ分けされており、その間のグラントは独立しています。主な表現・種類は以下のとおりです。

#### ①: \*\*点 / 1コモン

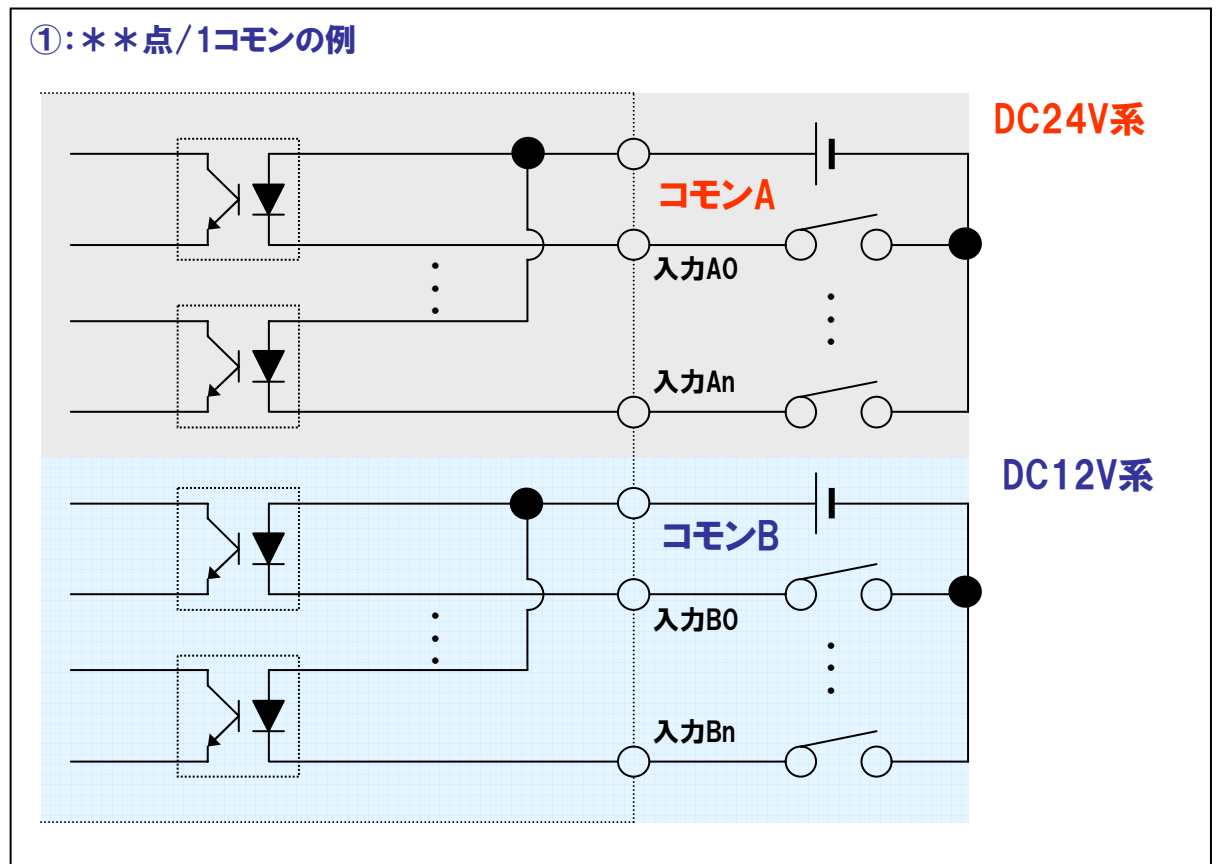
何点かの単位グループで独立しており、例えば16点/1コモンと記述されている場合は、『DC12V系で16点』『DC24V系で16点』といったように1枚のボード内で操作回路電圧を変えることができます。

#### ②: 全点共通

入出力の全点が共通のグラントを使用していることを意味します。操作回路の電圧やグラントレベルが異なる機器を同じボードに接続することはできません。

#### ③: 全点独立 (独立コモンタイプ)

各1点ずつ独立 (+端子と-端子がある) しており、1点ごとに操作回路の電圧やグラントレベルが異なる機器が接続可能なことを意味します。





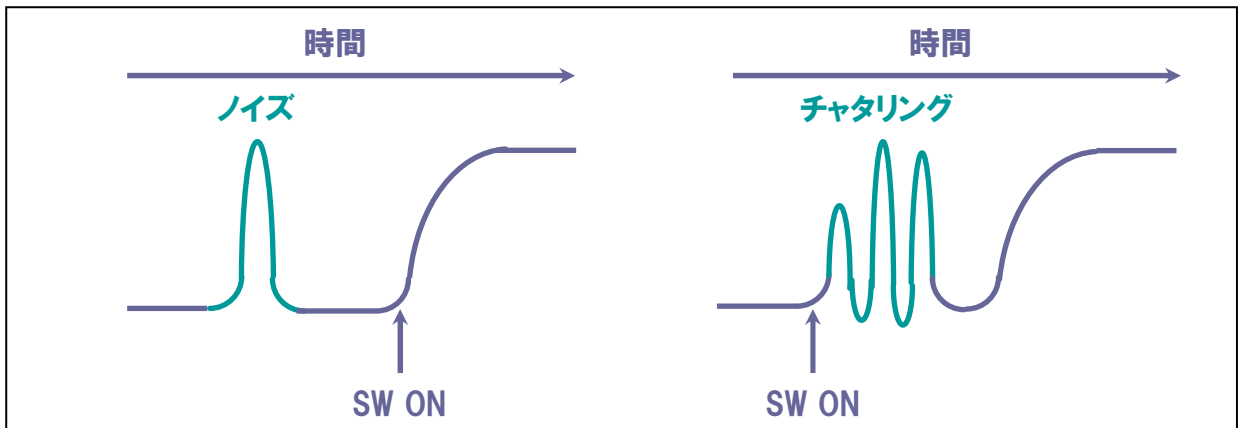
### 3-9-4.応答速度

信号入出力に発生するハード的な応答の遅延を表します。すなわち、信号が『Low:0→High:1』または、『High:1→Low:0』に変化したことをボード/カードが認識するまでの時間です。ただし、システム全体の応答速度は、ソフト的な遅延も含んで考えなければなりません。ハード的な遅延は短縮できませんが、ソフト的な遅延は、パソコンの処理能力、ソフトウェアシーケンスを最適化することで短縮が可能です。

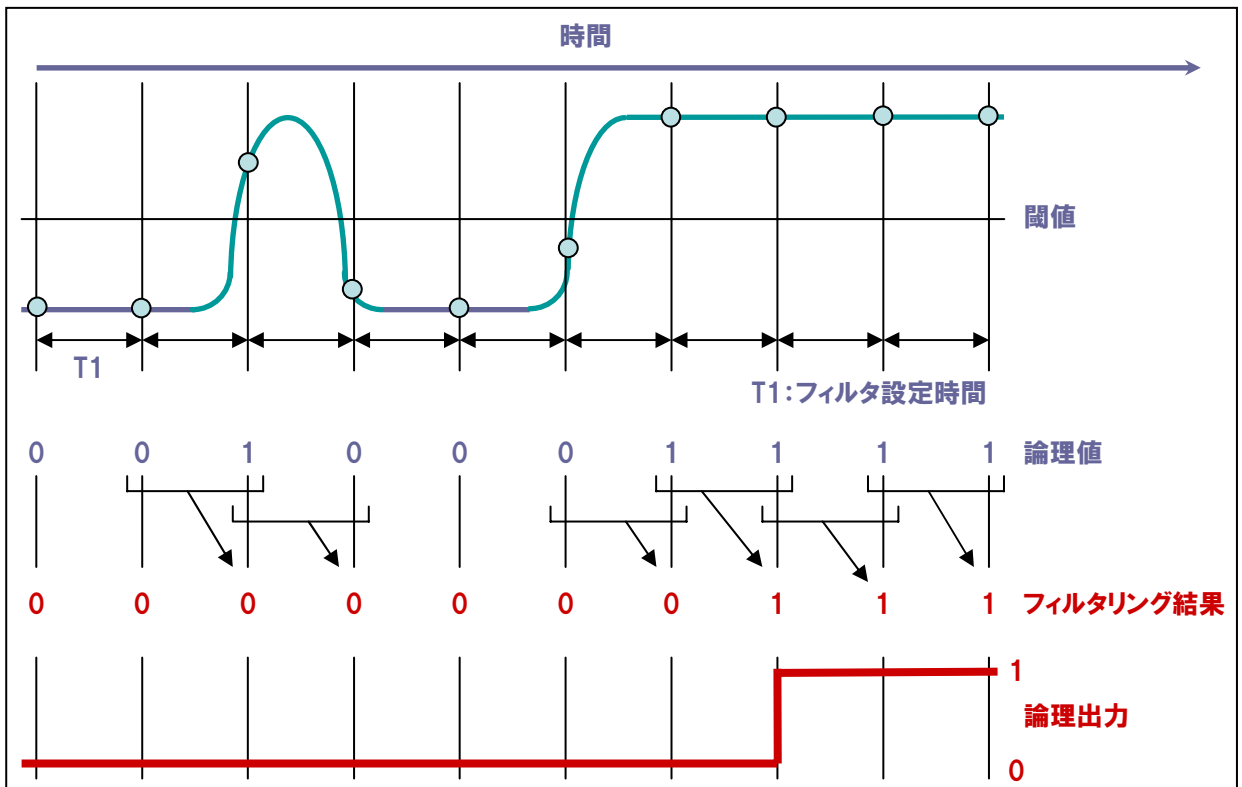
実応答速度は、『関数の実行時間(ソフトウェアの処理時間)』+『ボード/カードの応答速度(ハードウェアの処理時間)』となります。

### 3-9-5.デジタルフィルタ

FA系の制御システムでは、ノイズやチャタリングでシステムが誤動作するということが問題になるケースがあります。ノイズとは、スイッチが『ON』になっていないにも関わらず、信号が入ってくる。チャタリングとは、機械接点が開いている状態から閉じる時(またはその逆)に、機械的に振動して、接地したり離れたったりしてから安定する現象を言います。

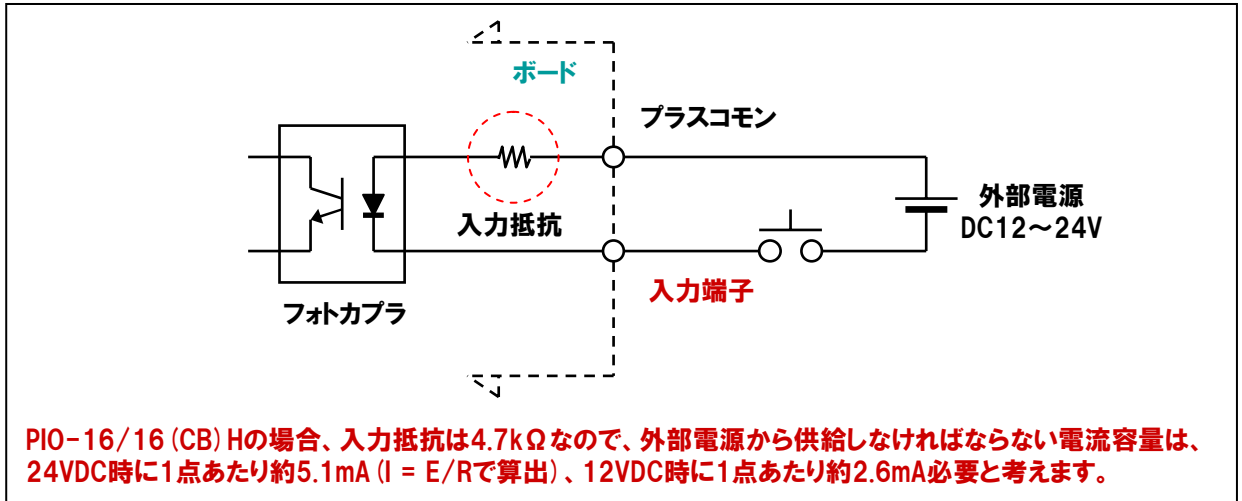


これらノイズ、チャタリング防止に用いられるのがデジタルフィルタです。弊社の場合は時定数によるフィルタが可能です。あらかじめ設定した時間以上その信号の状態が続いている場合のみ有効とし、それより短い場合には無効として処理を行います。ノイズの多い環境で信号を取込む場合、確実にスイッチの立ち上がりを捕らえて制御を行うときなどに有効です。フィルタ回路をハードウェア上に搭載している場合は、ソフトウェア処理と比較して時間精度や処理速度向上が実現します。



### 3-9-6.入力抵抗

フォトカプラ絶縁入力タイプのボードに実装されているフォトカプラに直列に接続される抵抗の値です。フォトカプラへ流す電流を調整する役割をしている抵抗の値とも言えます。TTL製品の場合は『プルアップ抵抗』および『保護抵抗』を『入力抵抗』と表現する場合があります。



### 3-9-7.最大定格

出力回路における接続可能な駆動電圧および出力電流の最大値です。この値を超える回路を接続した場合、ボード/カードの出力回路が破損する場合があります。

### 3-9-8.消費電流

ボードの動作には電源が必要ですが、そのボードがどれくらい電流を消費するかを示します。通常、この電源はパソコンの拡張バスコネクタから供給されています。そして、実装したボードの最大消費電流の総和が、パソコンの定格電源容量（拡張スロットに供給できる電流の最大値）を超えてはならないという注意点があります。もし超えた場合、パソコンの電源がドロップし、暴走などのトラブルが生じる可能性があるため、パソコンのスロットを拡張するための『I/O拡張ユニット』を使用するなどの対策を行う必要があります。

#### ■パソコンが拡張スロットから供給できる電源容量が3.6Aの場合

[1] 最大消費電流が、1.2Aの外部インターフェイスボードを2枚実装すると・・・

$$1.2 (A) \times 2 (枚) = 2.4A \rightarrow \text{パソコンの電源容量 (3.6A) 以下なので OK!}$$

[2] 最大消費電流が、0.8Aの外部インターフェイスボードを6枚実装すると・・・

$$0.8 (A) \times 6 (枚) = 4.8A \rightarrow \text{パソコンの電源容量 (3.6A) 以上なので NG!}$$

### 3-9-9.電源内蔵型ボード

絶縁タイプのボードを使用する場合、フォトカプラを駆動させるために外部から電源をボードに対して供給することが必要です。電源内蔵型ボードの場合には、パソコンのスロットから供給されている電源を、ボードに搭載されているDC-DCコンバータにより電圧変換し、フォトカプラへ供給します。これにより、外部に電源を用意しなくても絶縁タイプが使用できるようになります。しかし、内蔵電源を使用する場合には、ボードの消費電流が大きくなるため、複数枚を同時に一つのパソコン内で使用するには、パソコンの拡張スロットに供給可能な容量を超えない様に注意してください。

#### ■弊社型式:PIO-32/32B (PCI) Hの場合

[1] 外部から電源を供給 → 消費電流 = 5VDC 0.3A

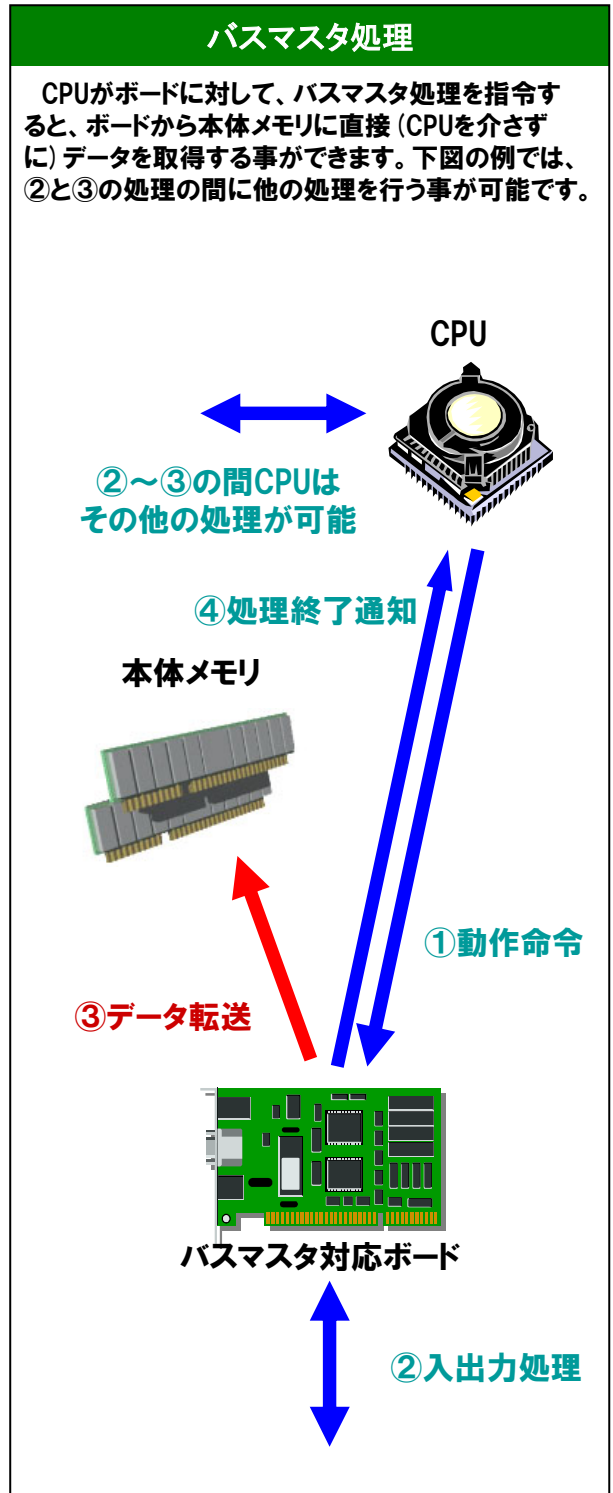
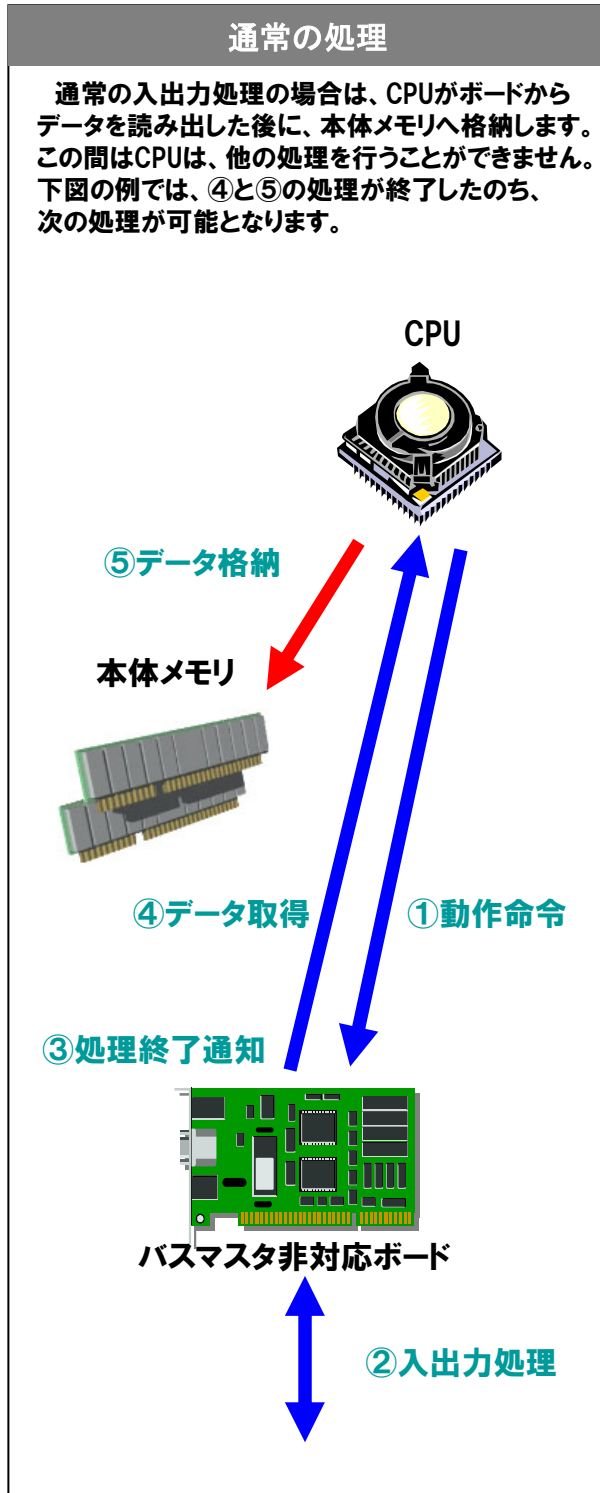
[2] 内蔵電源を使用 → 消費電流 = 5VDC 1.3A

### 3-9-10. バスマスタ転送機能

PCIバスマスタ機能を利用したDMA (ダイレクト・メモリ・アクセス) 転送が可能なボードです。

#### ■特長

- [1] パソコンのCPUに負荷をかけることなく、ボードから直接パソコンのメモリへ80MB/sec (最大133MB/sec) のスピードでデータ転送が可能です。
- [2] CPUがデータ転送処理などに能力を奪われることなく、その他の処理を行うことが可能なため、他のアプリケーションへの影響が少なく済みます。
- [3] 入出力に必要な設定をあらかじめボードにセットし、その情報に基づいてボードが処理を行うため、通常の入出力処理と比較して効率的なアプリケーションが構築可能です。



### 3-9-11.I/OポートとI/Oポートアドレス

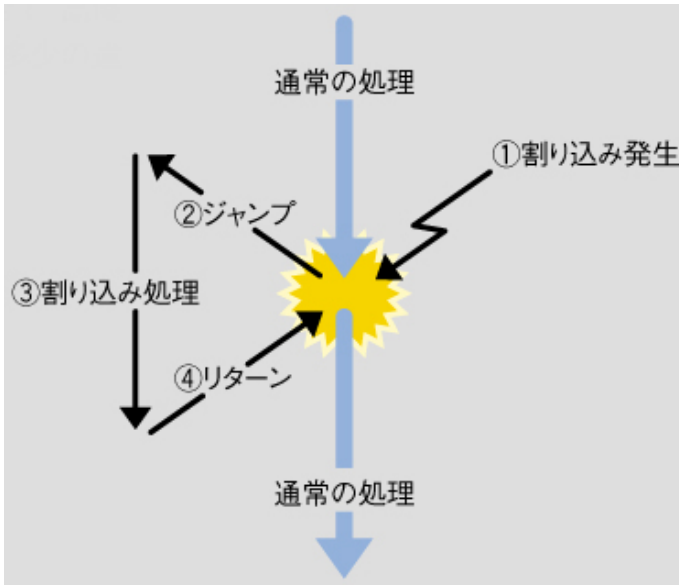
一般にパソコンの入出力命令によってコントロールされる機器 (I/Oデバイス) は、割り当てられた『I/Oポートアドレス』に対して入出力命令を実行し、動作させます。パソコンの内部で、CPUが接続された周辺機器とデータのやり取りを行うための窓口を『I/Oポート (港)』、それを管理・識別するために割り当てられる番号 (住所) を『I/Oポートアドレス』と表現します。現在主流のPC/AT互換機は、メモリ領域『0000h~FFFFh』64KBのアドレス空間が使用されます。『0000h~00FFh』など特定のI/Oアドレスは、システムによって使用済みもしくは予約済みとなっており、ユーザーでの使用が禁止されています。そのため、新たにデバイス (ボードやカード) を追加するためには、これ以外のアドレス、および既に存在しているアドレスと重複しないように設定します (Plug & Play機器は自動設定)。

#### ■占有ポート数

通常ボード/カードは、1枚で複数のI/Oポートを使用するケースが多く、使用するポート数を一般的に『占有ポート数』と表現します。占有ポート数は、コンピュータ内部での回路処理の関係上、『2<sup>n</sup>』が一般的です。

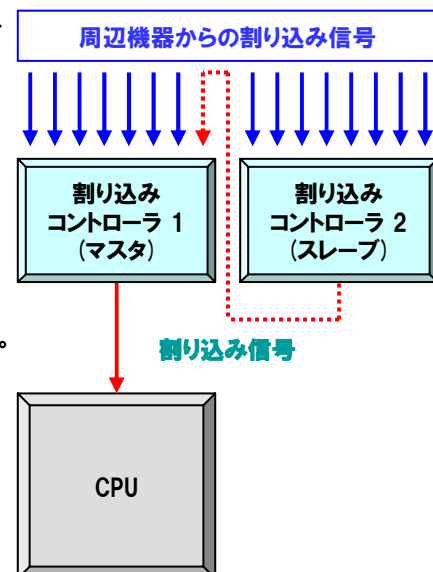
### 3-9-12.割り込み

特定の入力端子をパソコン (CPU) のIRQに接続して、外部から優先処理を発生させる機能です。外部装置の変化を検出して、特定の処理を実行するアプリケーションや、外部からの指令で高優先度の緊急処理などをする場合に使用します。



#### ■参考資料 (割り込みアドレス)

周辺装置はCPUの制御信号線の一つである割り込み信号線を利用して、CPUへデータの入出力などの割り込み要求 (IRQ:Interrupt ReQuest) を通知します。その際の識別番号を割り込みレベル (IRQ番号) と呼んでいます。CPUの持っている割り込み信号線は1本ですが、割り込み要求は複数のデバイスから同時に発生する場合があります。そのためパソコンは、8本の割り込み信号を周辺装置から入力して、その中から1本の割り込み信号だけをCPUに出力する機能を持った『割り込みコントローラ』を2個搭載し、制御しています。複数のデバイスから同時に割り込みが発生した場合には、コントローラで順番を調整し、順次CPUへ信号を送信します。信号を受け取ったCPUは、IRQ番号に対応付けられた処理を実行し、処理が終わったら次の割り込み処理へ制御を移していきます。



# 第4章

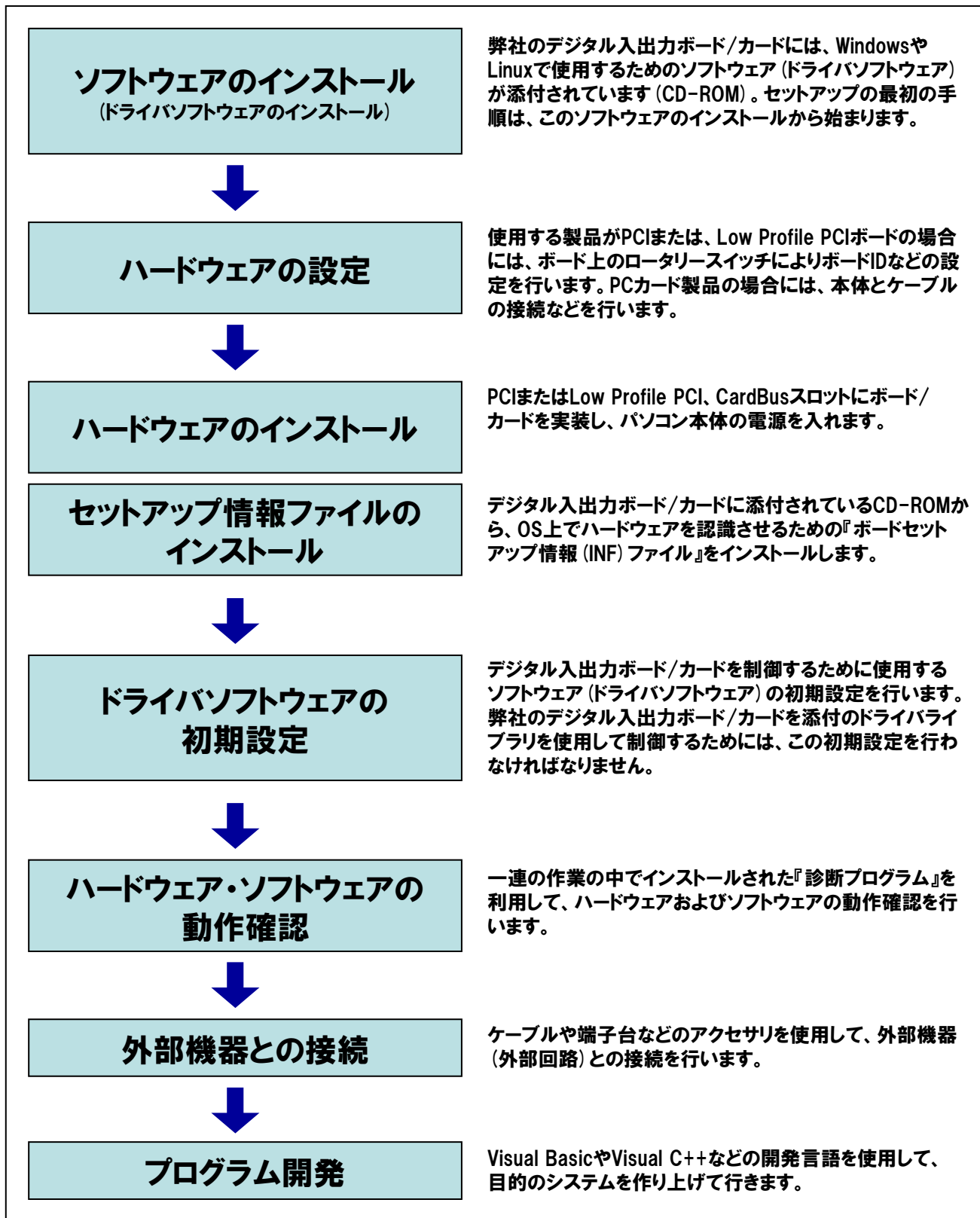
## ハードウェア・ソフトウェアの セットアップ



## 第4章 ハードウェア・ソフトウェアのセットアップ

インターフェイスボード/カードを使用するために必要な事前準備を一般に『セットアップ』と呼んでいます。ソフトウェアとハードウェアそれぞれの準備を行ってはじめてインターフェイスボード/カードが使用可能になります。バス仕様、OSおよびソフトウェアによって『セットアップ』の手順は異なる場合がありますが、本書ではWindows系OSの場合を例として解説を行います。

### 4-1. セットアップからプログラム開発までの流れ (Windows系OSの場合)





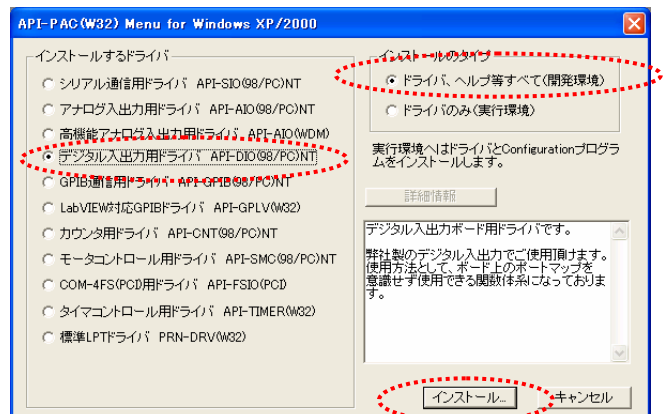
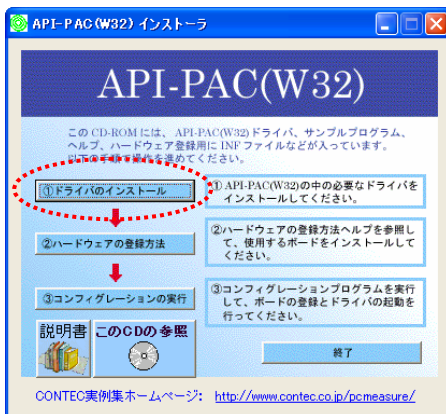
## 4-2.手順 ①:ソフトウェア (ドライバソフトウェア) のインストール

本書にて使用するプログラム開発環境を構築するためのセットアップを行っていきます。なお以降説明を行う手順は、下記構成を前提に進めています (OSおよび開発言語のインストールは既に完了しているものとします)。ご使用のOSによって画面表示が異なる場合がありますが、Windows系OSであれば、基本的な手順は同じです。

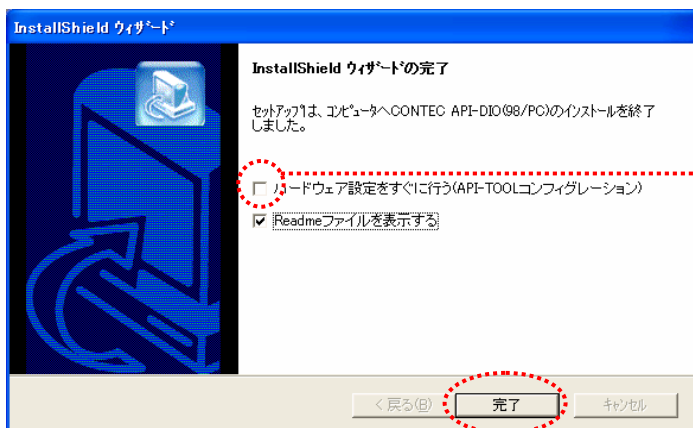
Windows XPおよびWindows 2000をご使用の際、インストール時はAdministrator権限を持つユーザーでログインして、インストールを進めてください。

- |           |   |                        |
|-----------|---|------------------------|
| ●OS       | :Microsoft Windows XP Professional/Home Edition |                        |
| ●開発言語     | :Microsoft Visual Basic 6.0                     |                        |
| ●インターフェイス | :CardBus対応絶縁型デジタル入出力カード                         | 【型式:PIO-16/16L (CB) H】 |
| ●ケーブル     | :37ピンD-SUBコネクタ両側コネクタ付シールドケーブル                   | 【型式:PCB37PS-1.5P】      |
| ●アクセサリ    | :デジタル入出力信号モニタ (チェックメイト)                         | 【型式:CM-32 (PC) E】      |
| ●ソフトウェア   | :Windows版デジタル入出力ドライバソフトウェア (製品添付)               | 【型式:API-DIO (98/PC)】   |

- ① インターフェイスボード/カードに添付されているCD-ROM [API-PAC (W32)] をパソコンにセットします。
- ② 『インストーラ画面』が自動的に表示されます。
- ③ 『インストーラ画面』の『ドライバのインストール』ボタンをクリックします。
- ④ 『インストールするドライバ』と『インストールのタイプ』の選択画面が表示されます。
- ⑤ 『デジタル入出力用ドライバ』を選択します。
- ⑥ 『ドライバ、ヘルプ等すべて (開発環境)』を選択します。
- ⑦ 『インストール』ボタンをクリックします。



- ⑧ 画面の指示に従ってインストール作業を進めます。
- ⑨ ファイルのコピー終了後『ハードウェア設定をすぐに行う』チェックをはずして『完了』ボタンをクリックします。これでソフトウェア (ドライバソフトウェア) のインストールは完了です。

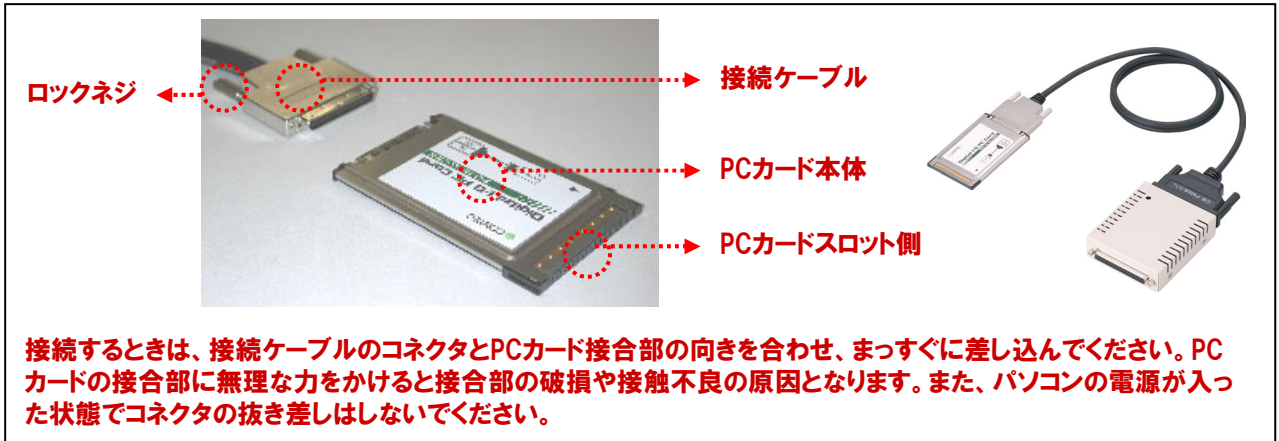


## 4-3.手順 ②:ハードウェアの設定

ここではPCカード (PIO-16/16L (CB) H) をパソコンに実装する手順を説明します。

### ① 接続ケーブルとPCカードの接続します。

絶縁回路ボックス『PIO-16/16L (CB) H-BOX』のPCカード側のコネクタとPCカード本体『PIO-16/16L (CB) H』を接続してください。コネクタの平らな面とPCカードの表面を合わせ、下図のように接続してください。



### ② PCカードをパソコン本体へ挿入します。

パソコンの電源がOFFであることを確認し、パソコンのPCカードスロットにPCカードを挿入してください。PCカードの▼印の向きに従って、下図のようにPCカードスロットに確実に差し込んでください。PCカードには誤挿入防止キーが付いていますが、無理に差し込むとPCカードスロットやPCカードの故障の原因になります。また、パソコンによってはPCカードの表面を下にして挿入するPCカードスロットがありますので、使用するパソコンのマニュアルを確認の上、挿入してください。PCカードを取り外すときも、使用するパソコンのマニュアルを参照してください。



次の行為は、PCカード本体および接合部の破損や接触不良の原因となりますので、行わないでください。

- ◎ PCカードを所定の方向および手順以外の方法で、挿入しないでください。
- ◎ 接続ケーブルまたはケーブルのコネクタをもって、PCカードを挿入しないでください。
- ◎ 接続ケーブルのコネクタを接続したまま、パソコンを移動しないでください。
- ◎ 接続ケーブルのコネクタを無理に引っ張るなどして、PCカードの接合部に力をかけないでください。
- ◎ 接続ケーブルのコネクタの上に、物を置かないでください。

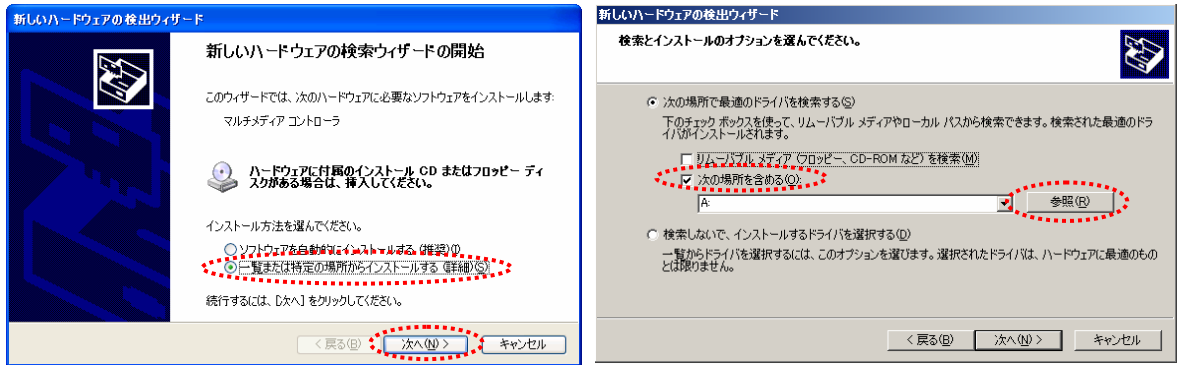
### ③ これで、ハードウェアの設定は完了です。



## 4-4.手順 ③:ハードウェアのインストール

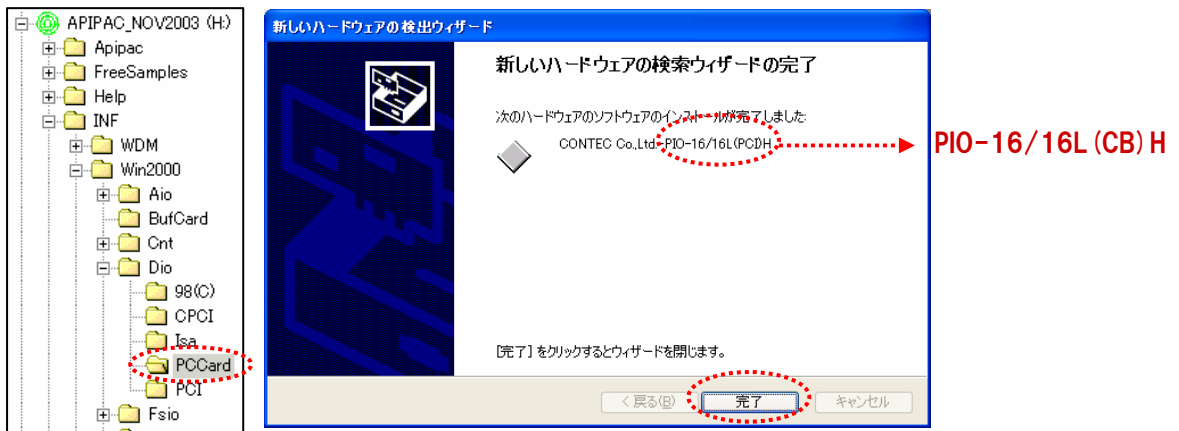
Windowsでは、PCカードが使用するI/Oアドレスと割り込みレベルを0Sに認識させる必要があります。これをハードウェアのインストールと呼びます。複数枚のPCカードを使用する場合は、必ず1枚ずつ設定が完了してから次のPCカードをインストールしてください。

- ① パソコンの電源を投入します(4-3.手順② ハードウェアの設定に従い、あらかじめパソコンにPCカードを挿入しておく必要があります)。
- ② 『新しいハードウェアの検出ウィザード』が起動します。『一覧または特定の場所からインストールする(詳細)』を選択して、『次へ』ボタンをクリックします。次の画面で、『次の場所で最適なドライバを検索する』の『次の場所を含める』にチェックをして、『参照』をクリックします。

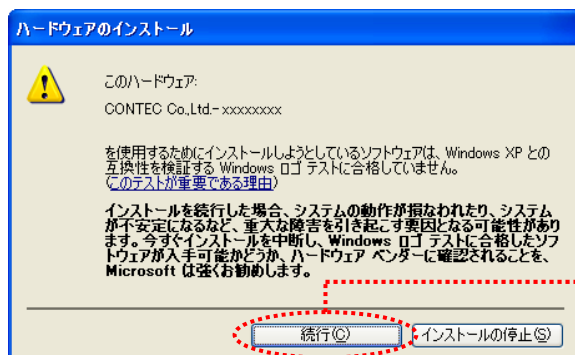


- ③ CD-ROMからセットアップ情報(INF)ファイルのあるフォルダを指定して、登録を行います。セットアップ情報(INF)ファイルは、添付CD-ROMの以下のフォルダにありますので、そのフォルダを指定して『OK』をクリックします。PIO-16/16L(CB)Hがパソコンに設定されますので、完了をクリックします。これで、インストールは完了です。

- Windows XP, 2000    ¥INF ¥Win2000 ¥Dio ¥PCCard
- Windows Me, 98    ¥INF ¥Win95 ¥Dio ¥PCCard



**注意)** Windows XPでは[ハードウェアウィザード]中のINFファイルを指定した後に以下の警告画面がでます。これは対象となるドライバが[Windowsロゴテスト]に対応していない場合に発生しますが、動作上は問題ありません。ここでは、[続行]ボタンをクリックしてください。

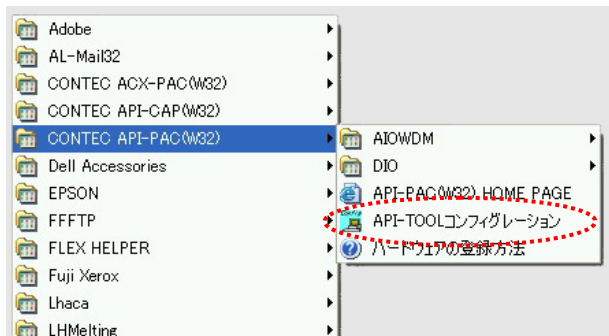


→ [続行] ボタンをクリックしてください

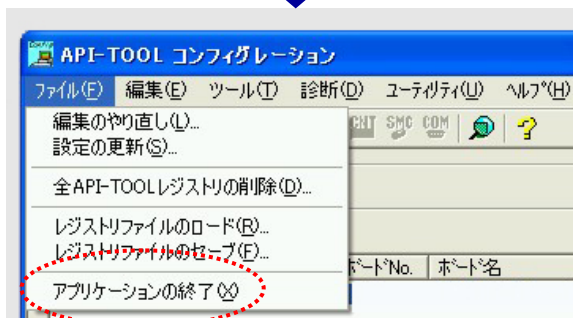
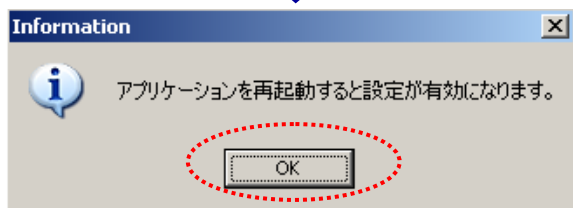
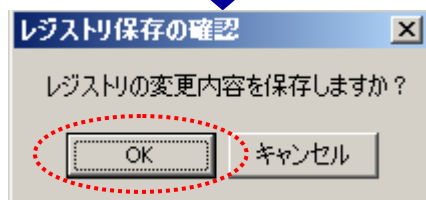
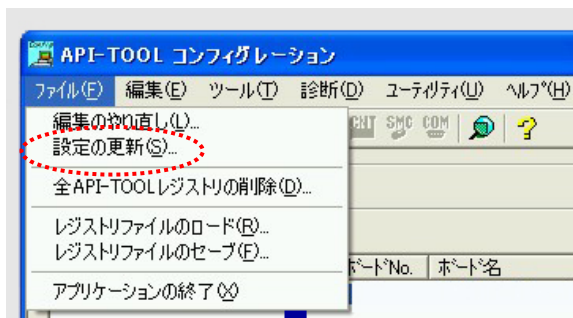
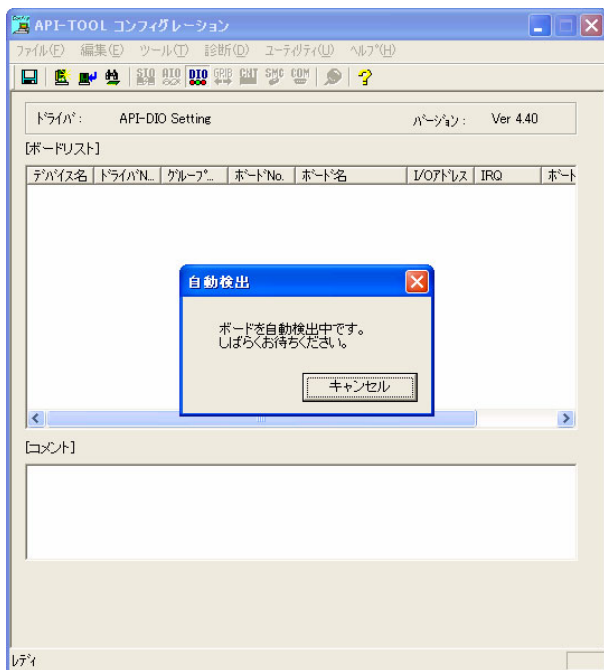
## 4-5.手順 ④:ドライバソフトウェアの初期設定

ドライバソフトウェアでは、実行環境を認識するための最初の設定が必要です。これをドライバソフトウェアの初期設定と呼んでいます。弊社のデジタル入出力ボード/カードを添付のドライバソフトウェアを使用して制御するためには、この初期設定を行わなければなりません。なお、以下の手順でドライバソフトウェアの初期設定を行う場合には、デジタル入出力ボード/カードがあらかじめパソコンに挿入されている必要があります。

- ① 『スタート』メニューの『プログラム』 - 『CONTEC API-PAC (W32)』 - 『API-TOOLコンフィグレーション』を実行してください。



- ② ハードウェアを自動で検出します。検出されたボードのリストが表示されます。
- ③ 『ファイル』 - 『設定の更新』を実行してください。これでソフトウェアの初期設定は完了です。

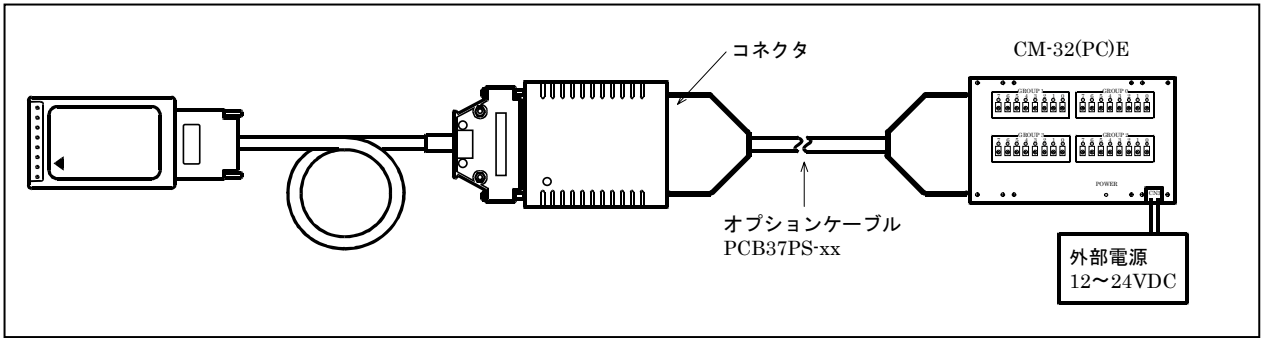


## 4-6.手順 ⑤:ハードウェア・ソフトウェアの動作確認

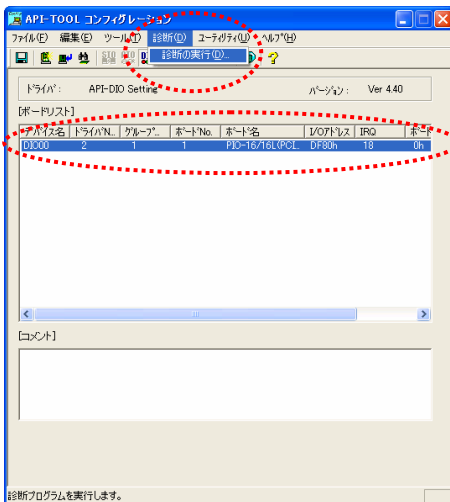
診断プログラムを使用して、ボードやドライバソフトウェアが正常に動作することを確認します。これでセットアップが正しくできたことを確認できます。診断プログラムは、ボードとドライバソフトウェアの状態を診断するプログラムです。実際に外部機器を接続したときの簡易動作確認として使用することもできます。また『診断レポート』機能を使用して、ドライバ設定、ボード存在有無、I/O状況、割り込み状況がレポートとして作成されます。

### 4-6-1.確認方法

- ・相手機器と接続して入出力テストや実行環境の確認を行ってください。
- ・このカードの場合、外部電源 (12~24VDC) をご用意ください。
- ・本書では、デジタル入出力ボードの確認にチェックメイト (CM-32 (PC) E) を使用します。



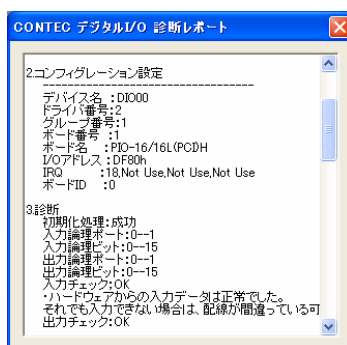
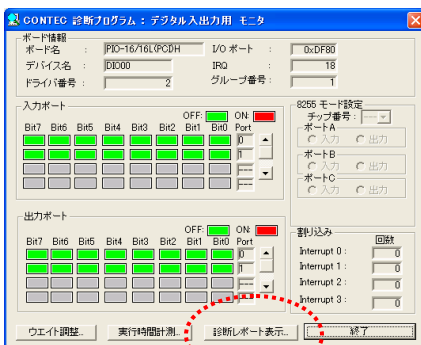
- ① 『スタート』メニューの『プログラム』 - 『CONTEC API-PAC (W32)』 - 『API-TOOLコンフィグレーション』を起動して、『PIO-16/16L (CB) H』を選択後、診断プログラムを実行します。画面の指示に従って操作してください。



PIO-16/16L (CB) Hが選択されている状態であることを確認した後、『診断』 - 『診断の実行』を選択してください。

- ② メイン画面が表示されます。以下のボックス内で現在の動作状態を確認できます。

- **入力ポート** : 一定時間ごとに入力した入力値をビットごとに表示します。
- **出力ポート** : 出力データをマウス操作することでそのデータを出力し、表示します。
- **割り込み** : 検出した割り込みの回数をビットごとに表示します。



診断レポート表示ボタンをクリックすることにより、診断した結果のレポートも出力可能です。

## 4-7.手順 ⑥:外部機器との接続

本書においては、外部機器の代わりに『デジタル入出力信号モニタ (チェックメイト) 型式:CM-32 (PC) E』を使用していますが、実システムの場合には、ボードまたはカードのインターフェイスコネクタと外部機器 (外部回路) とをケーブルまたは端子台を利用して適切に接続していかなければなりません。この接続により、プログラムの内容も決まるため、ここでは、スイッチ/ランプ (LED) / デジタルスイッチ (BCD) / 7セグメント表示器 (BCD) を搭載した仮想の機器を例にとって解説をしていきます。

### 4-7-1.外部機器との接続形態 (接続方法)

本カード (PIO-16/16L (CB) H) と外部機器との接続を行うには、弊社のケーブル・アクセサリを使用する場合、片側がバラ線になっている片端ケーブルを接続して、そのバラ線側と外部機器とを直接接続する方法と、両端コネクタケーブルと端子台を使用して、端子台と外部機器を電線で接続する方法のいずれかとなります。



### 4-7-2.仮想のデジタル入出力機器



#### ■入力

- ・トグルスイッチ×5個
- ・デジタルスイッチ (BCD) ×2桁

#### ■出力

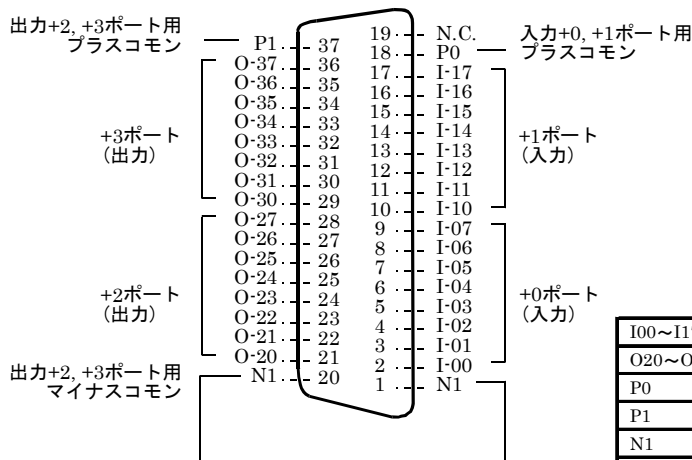
- ・LED×3個
- ・7セグメント表示器 (BCD) ×2桁

### 4-7-3. インターフェイスコネクタの信号配置図とI/Oポートマップ

ボードまたはカードと外部機器を接続するためには、2つの情報が必要となります。それがインターフェイスコネクタの信号配置図とI/Oポートマップ図（ポート・ビットとコネクタ信号ピンの関係を表した図）です。これらは、必ず製品の説明書（解説書）に記載されている項目です。I/Oポートに関しては、第3章で説明していますが、簡単におさらいをしておきましょう。一般にパソコンの入出力命令によって制御される機器は、割り当てられた『I/Oポートアドレス（1ポートは8ビット）』に対して入出力命令を実行し、動作させています。パソコンの内部で、CPUが接続された周辺機器とデータのやり取りを行うための窓口を『I/Oポート（港）』と呼んでいました。

『PIO-16/16L (CB) H』は、入力が16ビット、出力が16ビットのデジタル入出力カードなので、入力・出力それぞれ『2ポート分』をパソコンのI/Oポートアドレスに割り当てて入出力を行います。『PIO-16/16L (CB) H』は、PCカードですから、Plug & Play機能によりI/Oポートアドレスは、カードのインストール時に他のデバイスと重ならないように自動設定されます。付属のドライバソフトウェアや別売のActiveXコントロール集を使用してプログラムを行う場合は、この煩わしいI/Oポートアドレスを意識することなく、直感的なプログラムを実現するため『入力の+0ポート』、『出力の+0ポート』と指定するだけで制御できるようになっています。

#### ① PIO-16/16L (CB) H コネクタの信号配置図



I00~I17	入力信号16点です。他の機器からの出力信号を接続します。
O20~O37	出力信号16点です。他の機器の入力信号に接続します。
P0	外部電源のプラス側を接続します。(入力用)
P1	外部電源のプラス側を接続します。(出力用)
N1	外部電源のマイナス側を接続します。(出力用)
N.C.	このピンはどこにも接続されていません。

#### ② PIO-16/16L (CB) Hのポート・ビットとコネクタ信号ピンの関係を表した図

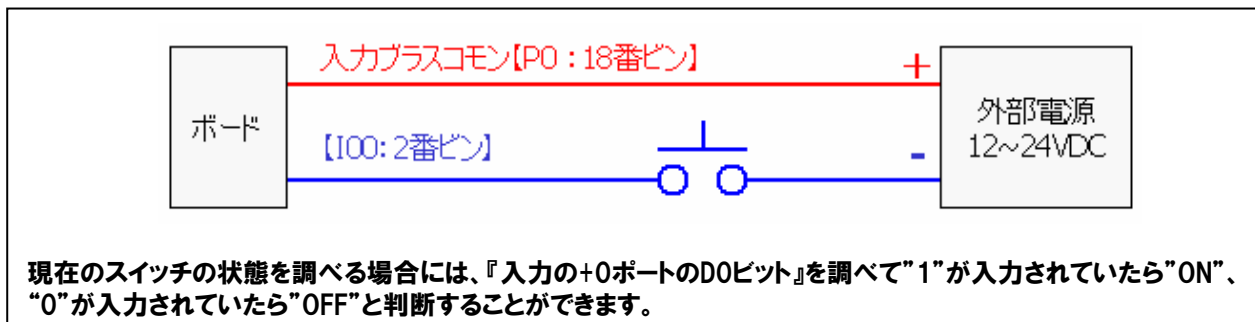
	D7	D6	D5	D4	D3	D2	D1	D0
入力論理ポート0	I-07 [7]	I-06 [6]	I-05 [5]	I-04 [4]	I-03 [3]	I-02 [2]	I-01 [1]	I-00 [0]
入力論理ポート1	I-17 [15]	I-16 [14]	I-15 [13]	I-14 [12]	I-13 [11]	I-12 [10]	I-11 [9]	I-10 [8]
出力論理ポート0	O-27 [7]	O-26 [6]	O-25 [5]	O-24 [4]	O-23 [3]	O-22 [2]	O-21 [1]	O-20 [0]
出力論理ポート1	O-37 [15]	O-36 [14]	O-35 [13]	O-34 [12]	O-33 [11]	O-32 [10]	O-31 [9]	O-30 [8]

説明：I-xxは入力信号であり、O-xxは出力信号です。  
[xx]は、論理ビットです。

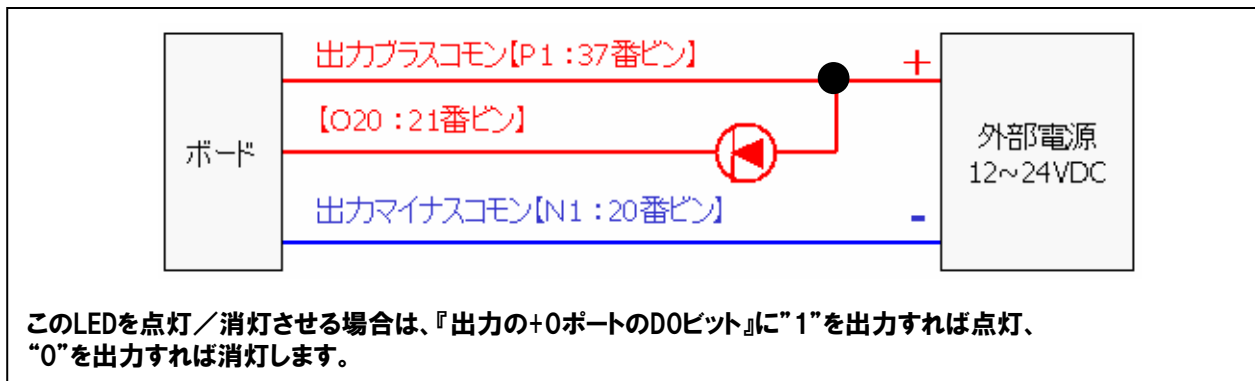
論理ポート番号、論理ビット番号は、ボードのI/Oアドレス、ボードの種類に依存しないプログラミングを可能にするための、仮想的なポート、ビット番号です。詳細は、API-PAC (W32) インストール後のAPI-DIO HELPを参照してください。

#### 4-7-4.外部機器との配線

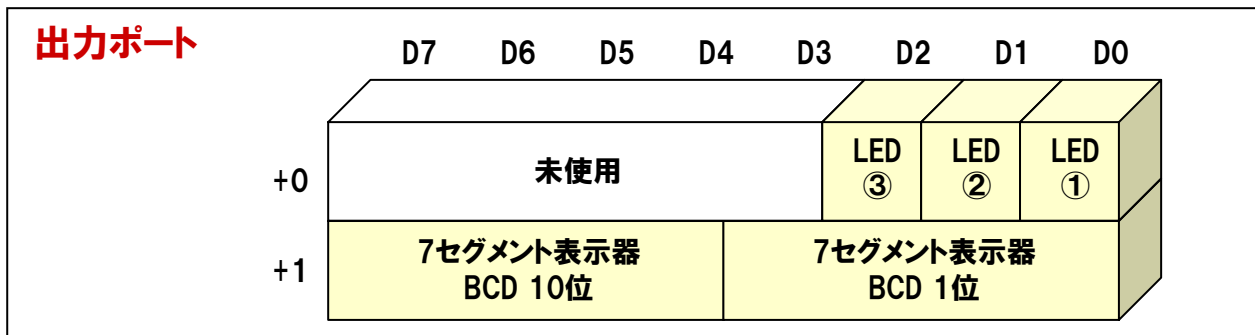
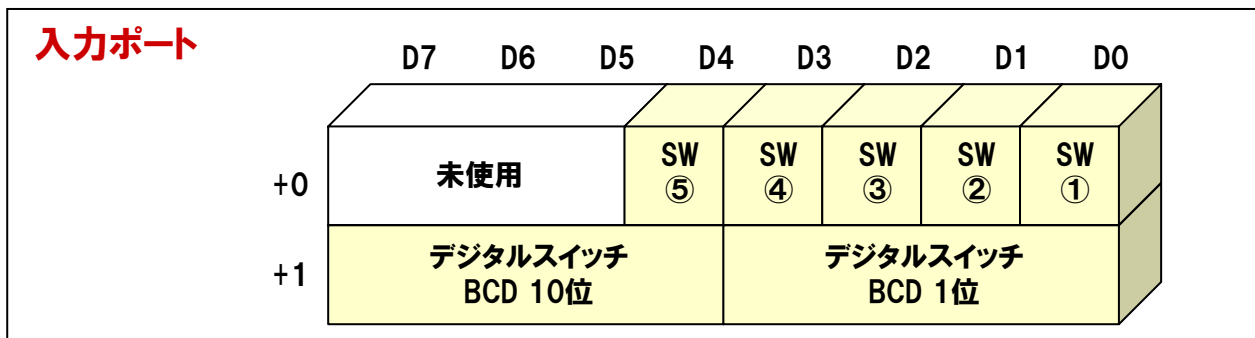
実際の外部機器との配線ですが、例えば『入力の+0ポートのD0ビット』すなわち、信号名『I-00』とスイッチ①との関連付けを行う場合は以下のように配線します。



同様に『出力の+0ポートのD0ビット』すなわち、信号名『O-20』とLED①（ランプ）を関連付ける場合は以下のように配線します。



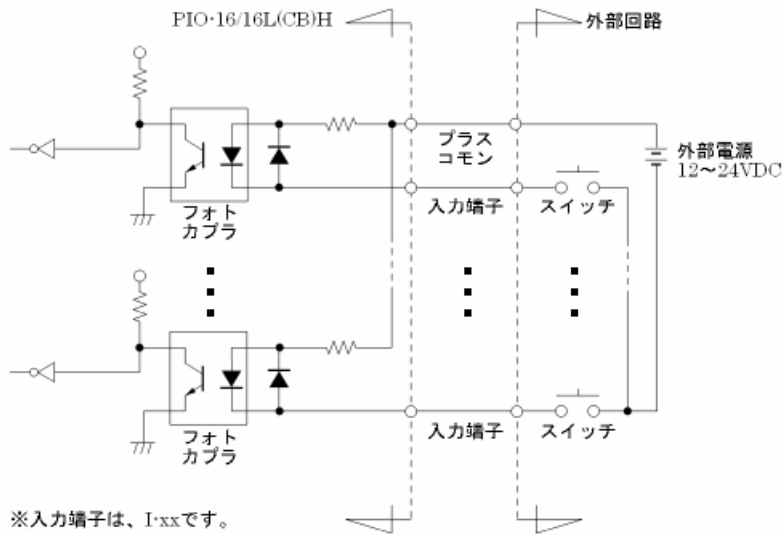
実機器との接続を行うには、下図のような独自のI/Oポートマップ図を作成しておく、実際にプログラムを行う際や、機器との接続をする際に大変便利です。下図は、4-7-2.の仮想機器のポートマップ図例です。





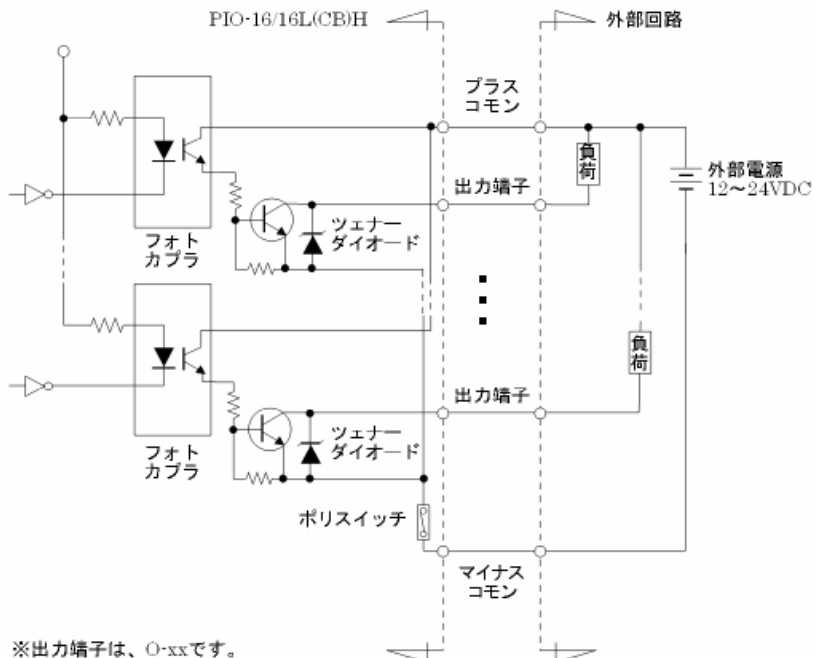
## 4-7-5.PI0-16/16L (CB) Hの入出力回路図

### ① PIO-16/16L (CB) Hの入力回路図



信号入力部は、フォトカブラ絶縁入力（電流シンク出力対応）になっています。したがって、このボードの入力部を駆動するためには外部電源が必要です。このとき必要となる電源容量は、24VDC時入力1点当たり約5.1mA（12VDC時には約2.6mA）です。

### ② PIO-16/16L (CB) Hの出力回路図



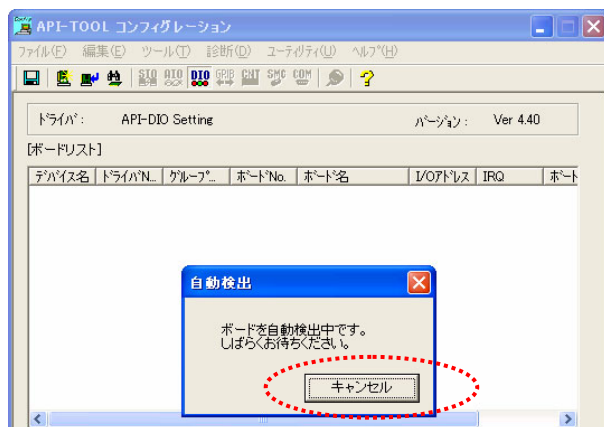
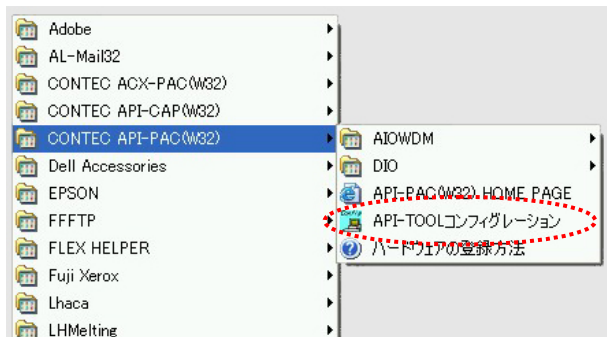
信号出力部はフォトカブラ絶縁オープンコレクタ出力（電流シンクタイプ）で、出力部を駆動するためには外部電源が必要です。出力電流の定格は1点当たり最大100mAです。  
出力に低飽和トランジスタを使用しているため、TTLレベル入力にも接続可能です。出力ON時のコレクタ・エミッタ間の残留電圧（LOWレベル電圧）は、出力電流50mA以内で0.5V以下、出力電流100mA以内で1.0V以下です。  
出力トランジスタには、サージ電圧からの保護のためツェナーダイオードが接続されています。また、過電流保護のためのポリスイッチが、出力トランジスタ8点単位で取り付けられています。

## 4-8.デモボード (仮想ハードウェア) の登録・設定方法

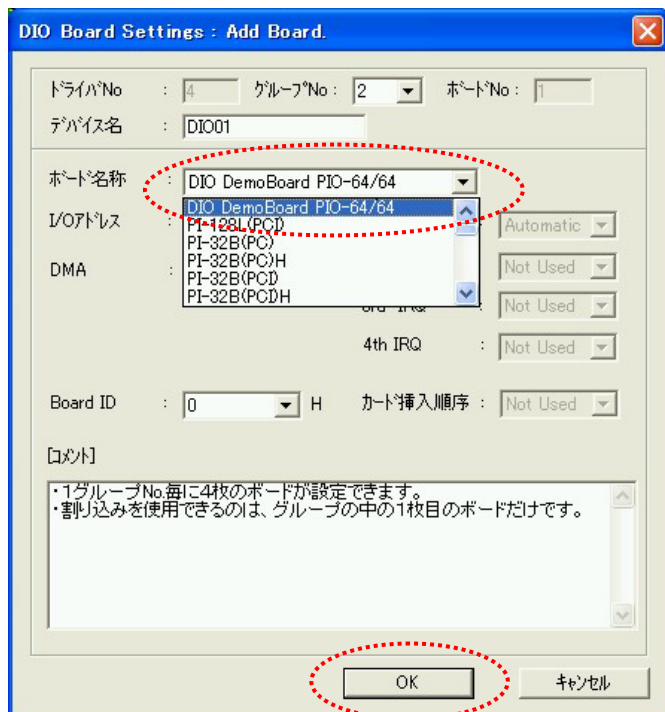
実際のハードウェア (ボードまたはカード) を用意できない場合、ドライバソフトウェアに含まれているデモボード (仮想ハードウェア) とデモボード操作ユーティリティを使用することによって、プログラムの動作確認を行うことができます。ここでは、デモボードの登録方法と操作ユーティリティ (CM\_64) の使用方法を解説します。なお、実際のハードウェアを使用する際には、本項の設定は必要ありません。

### 4-8-1.デモボードの登録方法

- ① ドライバソフトウェアを前述の『4-2.』の手順に従って、インストールを行ってください。
- ② 『4-5.』の手順に従って、『API-TOOLコンフィグレーション』を実行してください。
- ③ 『ボードの自動検出』ダイアログが表示されたら、『キャンセル』をクリックしてください。



- ④ メニューの『編集』 - 『ボードの追加』を実行してください。ボードの追加ダイアログが表示されます。ダイアログ中のメニュー『ボード名称』コンボボックスから、『DIO DemoBoard PIO-64/64』を選択し、『OK』をクリックしてください。これで、デモボードの登録は完了です。

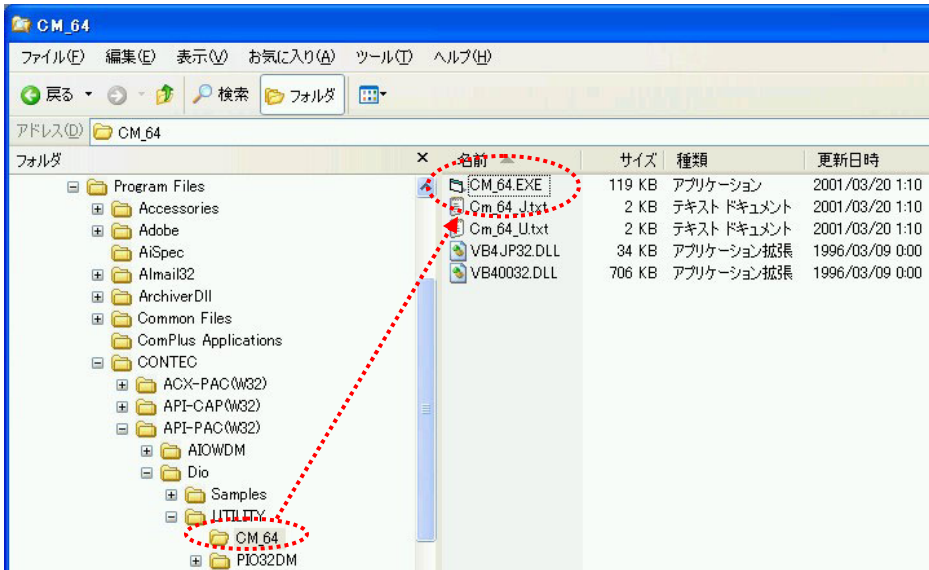




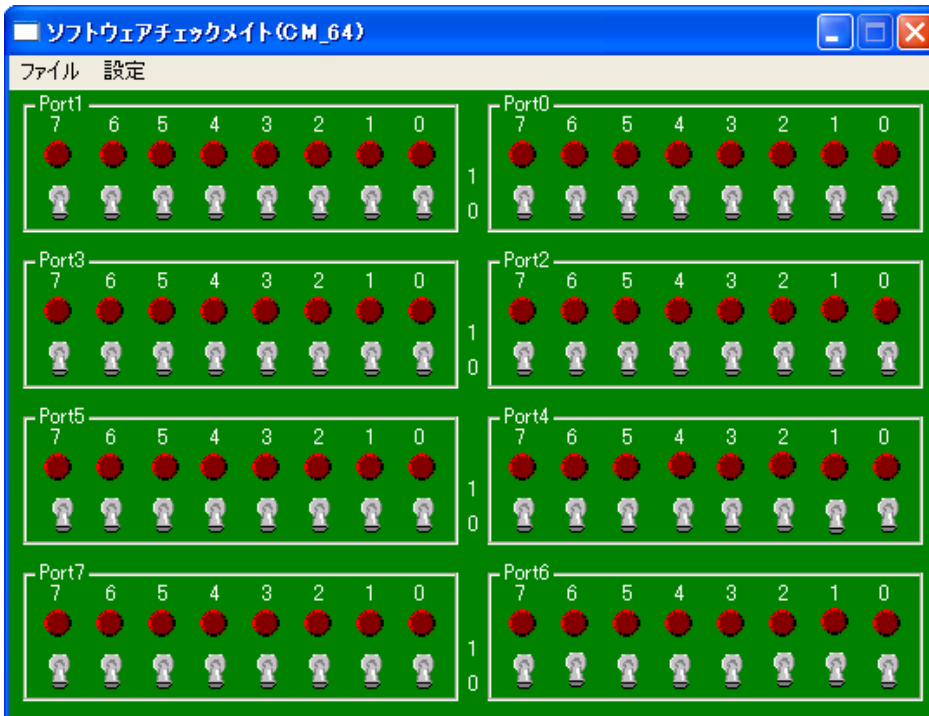
デモボード(仮想ハードウェア)は、入力64点・出力64点を持った仮想のハードウェアです。このデモボードを操作するためのユーティリティがドライバソフトウェアで提供されています。ここでは、その実行方法とデモボードの動作確認を行います。

## 4-8-2.デモボード操作ユーティリティ(CM\_64)の実行方法

① C:\¥Program Files¥CONTEC¥API-PAC(W32) ¥Dio¥UTILITY¥CM\_64の中のCM\_64.EXEを実行してください。

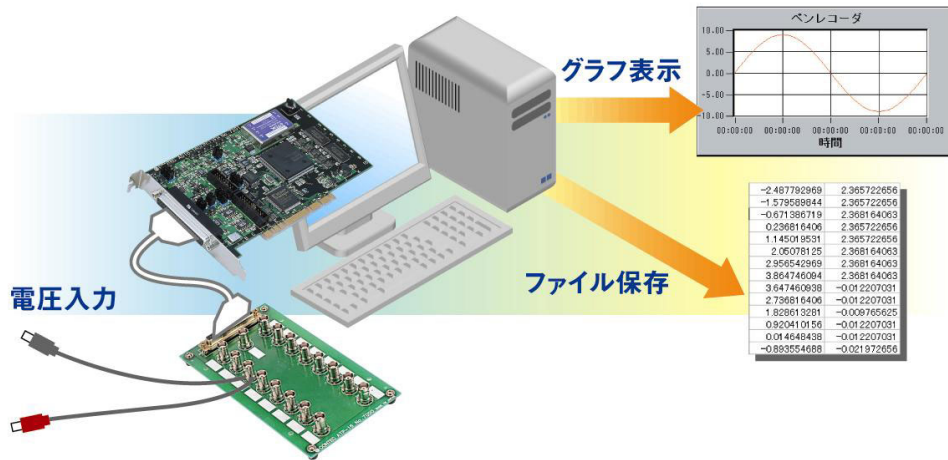


② デモボード操作ユーティリティ『CM-64』が起動します。その後、『4-6.』の手順に従って、デモボードの動作確認を行ってください。『CM\_64』の詳細は、ヘルプファイル『Cm\_64\_J.txt』を参照してください。



## 4-9. 参考資料 (ドライバソフトウェアの必要性)

本章のセットアップにおいて、まず始めの手順として『ドライバソフトウェア』のインストールを行いました。なぜ、ボードやカードを使用する際に、この『ドライバソフトウェア』を使用しなければならないのでしょうか。その理由を説明します。そのためには、計測・制御アプリケーションを作成するにはどうするか、ということから説明しなければなりません。計測・制御アプリケーションを作成するには、一般にVisual Basicなどの開発言語でハードウェア (ボード、カードなど) を操作するプログラムを作成し、接続した機器からデータを収集、機器の制御をする方法が広くとられています。



そのためには、まずハードウェアを操作するプログラムを作成しなければなりません。以前のパソコンOSの主流であったMS-DOSと異なり、現在の主流であるWindowsは、ユーザーアプリケーションからハードウェアを直接操作することができなくなっています。プロテクトモードのOSと呼ばれるWindowsは、不正なハードウェアアクセスを防止するためにユーザーがI/Oポートやメモリ領域に対して直接アクセスすることを制限しているからです。そのため、使用するハードウェアのメーカーが提供しているデバイスドライバと呼ばれるソフトウェアを介して間接的にハードウェアを操作し、アプリケーションを作成します。具体的には、ユーザーアプリケーションからドライバソフトウェア (DLL: ダイナミックリンクライブラリ) が提供している関数を開発言語から呼び出して、ハードウェアの操作を行います。



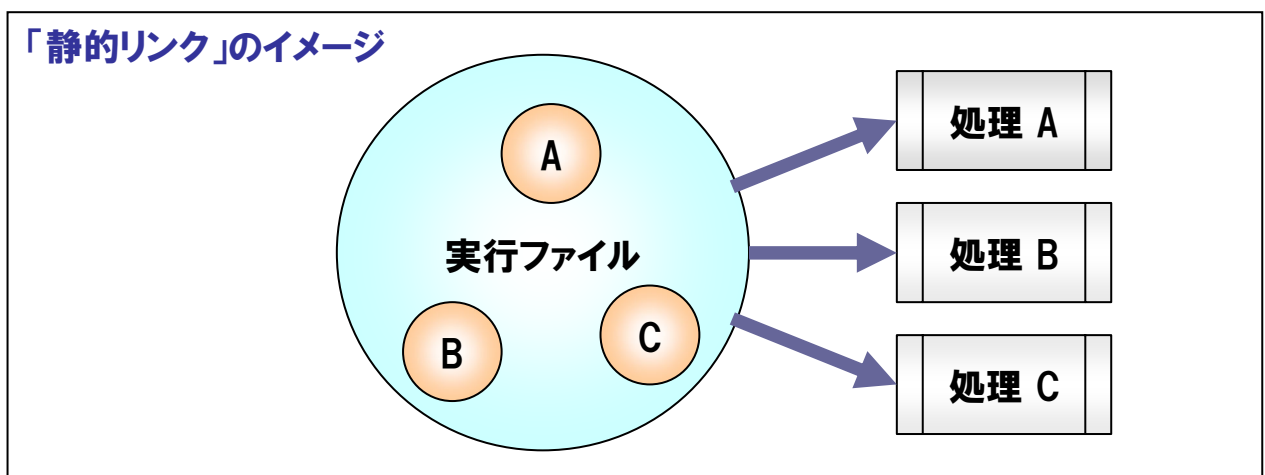
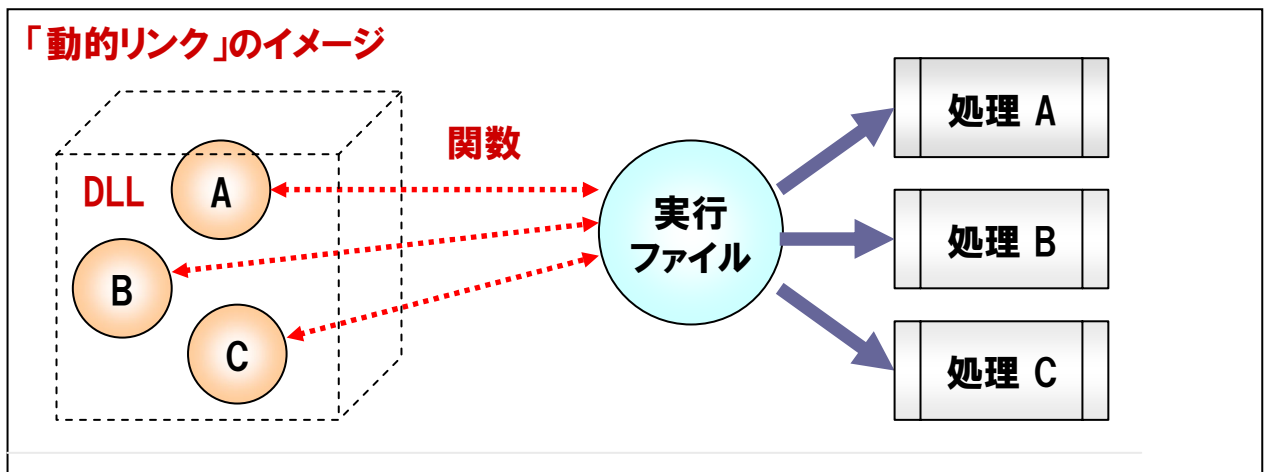
現在、さまざまな機能を持った各種ハードウェアが多数のメーカーから販売されていますが、仮に高機能ハードウェアであっても、それを操作するドライバソフトウェア (関数など) が使いやすく設計されていなければ、結局は、使いにくい物になってしまうということです。すなわちWindows環境では、ハードウェアが持っている機能はもとより、どれだけ使い易いソフトウェアが提供されているかが、製品選定の重要な要素となっているのです。

## 4-10. 参考資料 (DLL: Dynamic Link Libraryとは)

さて、先程出てきたDLL (Dynamic Link Library) とは何でしょう。DLLとは、複数のアプリケーションから共通利用可能なプログラムの集まりのことです。必要に応じてメモリにロードして利用します。DLLとして提供されている機能は、ユーザーがプログラムをすることなく利用できるため、アプリケーション開発効率が向上します。また、複数のアプリケーションで共通利用できるため、メモリの占有量も節約可能です。Windowsは、その機能の大部分をDLLとして提供しています。

### 4-10-1. Windows環境における実行ファイル (.EXEファイル)

Windows環境で作成される多くの実行ファイルは、必要とする処理を逐次、DLLなどのライブラリファイルに格納されている機能呼び出し実現しています(動的リンク)。そのため、MS-DOS環境で作成されるような従来の実行ファイル(静的リンク)と異なり、参照しているDLLなどのライブラリファイルも動作に必要となるのです。Visual Basicは、実行ファイルを作成する際『ディストリビューション・ウィザード』と呼ばれるツールを提供し、DLLを含めたセットアップファイルを簡単に作成できるようになっています。



# 第5章

## Visual Basic6.0による デジタル入出力プログラミング



## 第5章 Visual Basic6.0によるデジタル入出力プログラミング

第4章までの解説と準備で、デジタル入出力のプログラミング環境が整いました。本章では、Visual Basic6.0を使用して、外部からのデジタル信号入力とパソコンから外部機器へのデジタル信号出力などのプログラミング方法に関して解説をしていきます。実際のプログラミングを始める前に、Visual Basic6.0に関して簡単な説明をしておきましょう。

### 5-1. Visual Basicでのプログラム開発

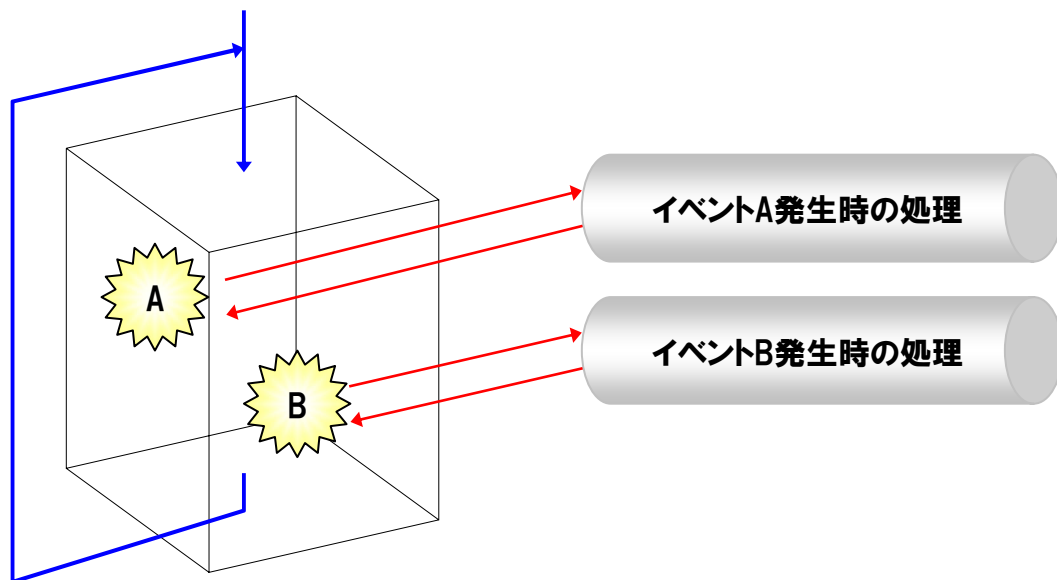
Visual Basicは、GUI (Graphical User Interface) アプリケーションを簡単に作成するため、従来の『手続き型プログラミング』ではなく、『イベント駆動型 (イベントドリブン) プログラミング』という手法を採用しています。Visual Basicは、米国Microsoft社によって開発されたWindowsアプリケーションを簡単に作成できるプログラミング言語です。

#### 5-1-1. イベント駆動型 (イベントドリブン) 言語とは

Visual BasicやVisual Cなど、Windowsのプログラム開発言語の多くは、イベント駆動型 (イベントドリブン) のプログラミングを採用しています。先頭行のコードから最終行のコードまでを順番に処理をしていく従来の『手続き型プログラミング』と何が違うのでしょうか。Visual Basicの実行状態を例にとって説明していきましょう。

Visual Basicは、フォームと呼ばれるプログラムの土台に配置された各オブジェクト (コントロールなど) の『イベント』を常に監視しており、『イベント』が発生した際には、その発生した『イベント』に対応しているプロシージャ (コード) が実行されようになっています。つまり、『イベント』が発生した際 (タイミング) に『何を行いたいのか』を記述することで、プログラムを作成します。

#### Visual Basicの実行状態



上図はVisual Basicの実行状態を図化したものです。プログラムの動作を開始するとVisual Basicは『イベント監視状態』に入ります。あらかじめ『イベントA』が発生した際に『何をやりたいか』、『イベントB』が発生した際に『何をやりたいか』をコードで記述しておきます。例えば、『イベントA』は、フォーム上に配置された『コマンドボタン』をクリックしたという『イベント』と仮定すると、『コマンドボタン』がクリックされた際に画面上に『Hello!!』と表示させるコードを記述しておけば、『コマンドボタン』をクリックすれば、画面上に『Hello!!』が表示され、その処理が終われば再度『イベント監視状態』に戻ります。この『イベント監視状態』は、プログラムの終了まで続きます。



## 5-1-2. Visual Basic 6.0用語解説

### ①フォーム

作成するアプリケーションのウィンドウやダイアログボックスをデザインする部分を指します。絵画に例えれば、キャンバス(土台)です。Visual Basicではこのフォーム上に様々な部品を配置して一つのアプリケーションを作り上げていきます。保存されるファイルの拡張子は『.frm』です。

### ②オブジェクト

Visual Basicでは、『制御の対象となるコードとデータを持った1つの集合体』という意味を持ちます。アプリケーションは、このオブジェクトの集まりで、言い換えるならばオブジェクトとは『アプリケーションを構成する部品の1つ1つ』といえます。

### ③コントロール

ある特定の機能を持ったソフトウェア上の部品を意味します。Visual Basicでは『ボタン』の機能を持つ『コマンドボタンコントロール』やテキストを表示する機能を持った『テキストボックスコントロール』などがあります。

### ④プロパティ

コントロールやオブジェクトが持つ『属性』をプロパティと呼びます。例えば、背景色や形などです。人間に当てはめると、その人(オブジェクト)の服装や髪型などがプロパティと考えれば分かりやすいです。

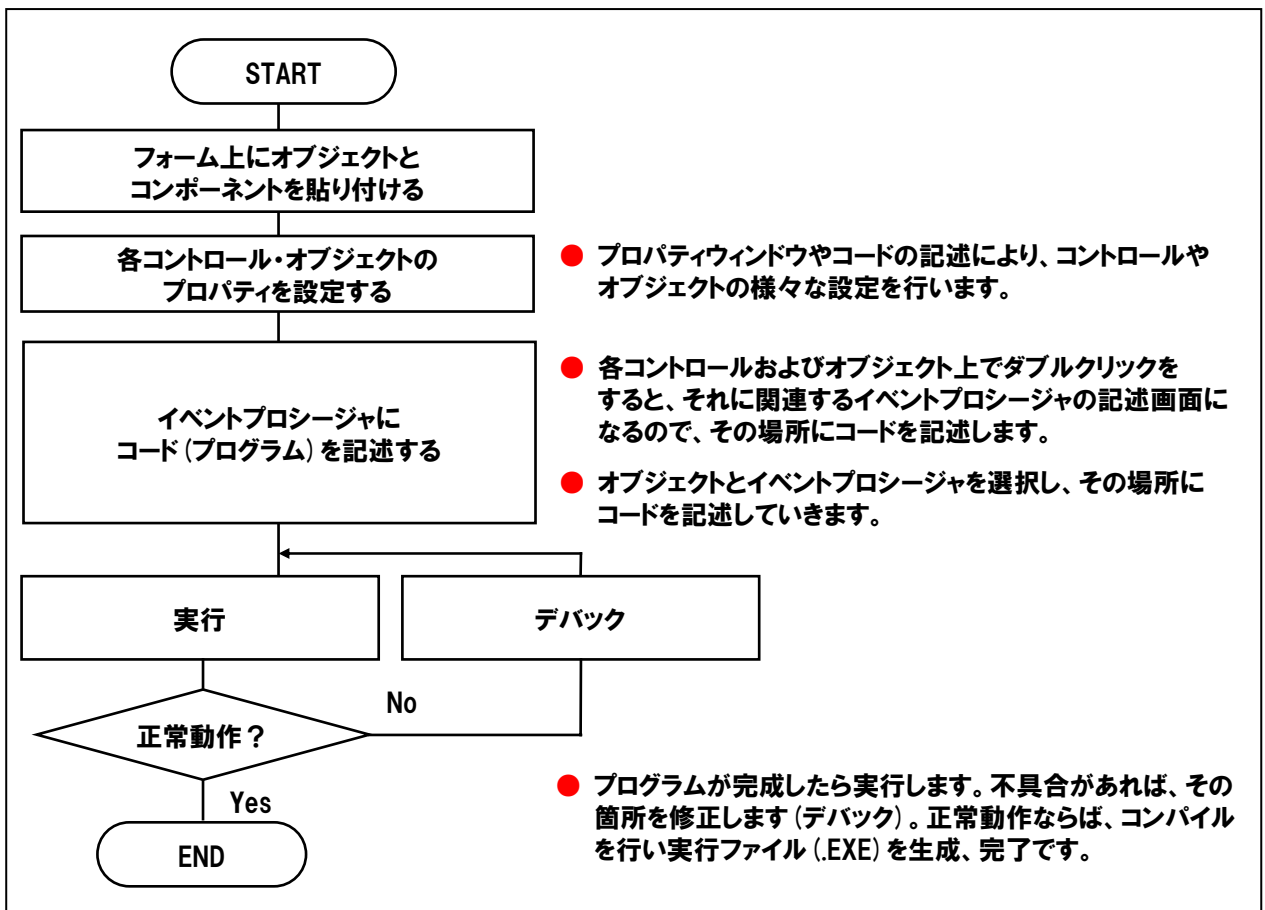
### ⑤イベントプロシージャ

実行時に1つの単位として処理されるコードの集まりを『プロシージャ』と呼びます。Visual Basicでは、『Subプロシージャ』『Functionプロシージャ』『イベントプロシージャ』があります。『イベントプロシージャ』は、イベント発生時に自動的に呼び出され実行されます。

### ⑥メソッド

特定のコントロールやオブジェクトの『状態を変化させるための命令』です。Visual Basicでは、この他の命令として、ステートメントや関数があります。

## 5-1-3. Visual Basicのプログラム作成手順



まず最初に、Visual Basicのプログラミングに慣れるためにVisual Basicの基本機能を利用した簡単なプログラムを作成してみましょう。

## 5-2.タイマコントロールを使用したカウントアッププログラム

Visual Basicの標準コントロール『タイマコントロール』のタイマイベントが発生するたびに、用意した変数の値を+1カウントアップして、その値を画面に表示させてみましょう。『開始ボタン』を押すとカウントアップが開始され、『停止ボタン』でカウントアップを停止させます。プログラム上で動作する簡易ストップウォッチのイメージです。カウントアップは100msec (0.1秒) 毎に行うように設定します。



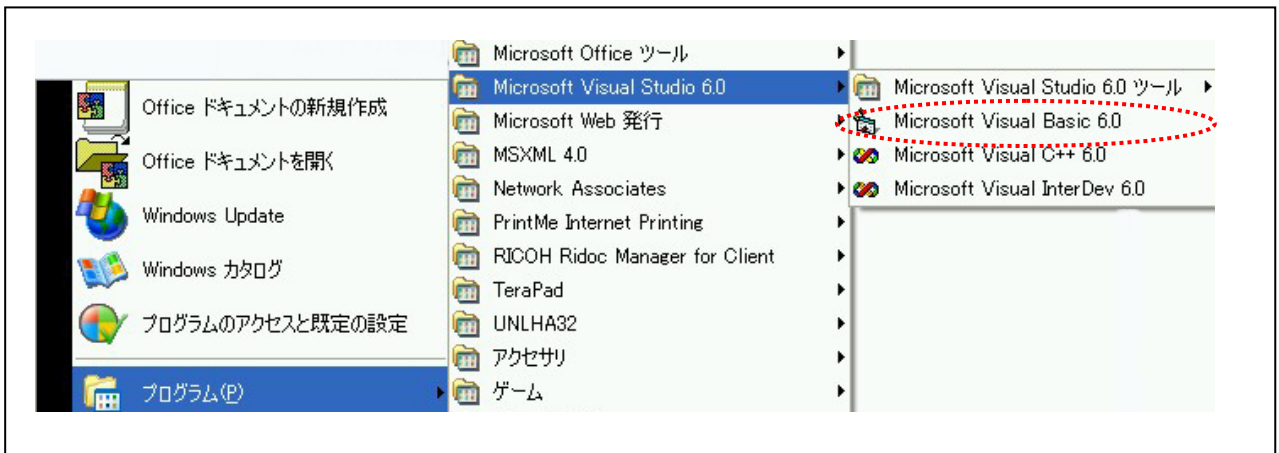
### 5-2-1.プログラム フローチャート



### 5-2-2.プログラム作成手順① (Visual Basicの起動)

まず最初にVisual Basic6.0を起動しましょう。インストール時の設定が標準で、その後変更していなければ、次の手順で起動します。

『スタートメニュー』 - 『プログラム』 - 『Microsoft Visual Studio6.0』 - 『Microsoft Visual Basic6.0』を選択します。





『新しいプロジェクト』ダイアログボックスが表示されますので(※)、『新規作成』タブ内の『標準EXE』を選択して、『開く』ボタンをクリックしてください。

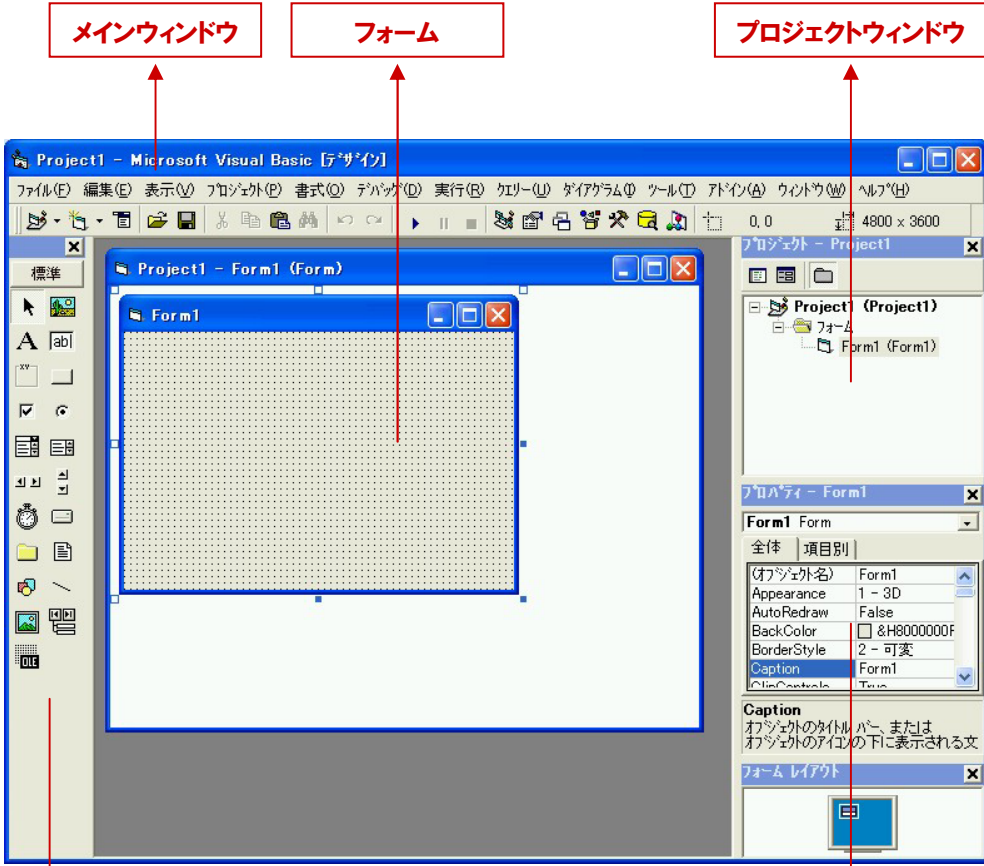
(※:『新しいプロジェクト』ダイアログボックスが表示されない場合は、メニューの『ファイル』-『新しいプロジェクト』を選択すると同様のダイアログボックスが表示されます。)



『新しいプロジェクト』ダイアログボックスが最初に開かれなかった場合は、下記の手順でダイアログを開きます。



### Visual Basic6.0の画面構成



メインウィンドウ

フォーム

プロジェクトウィンドウ

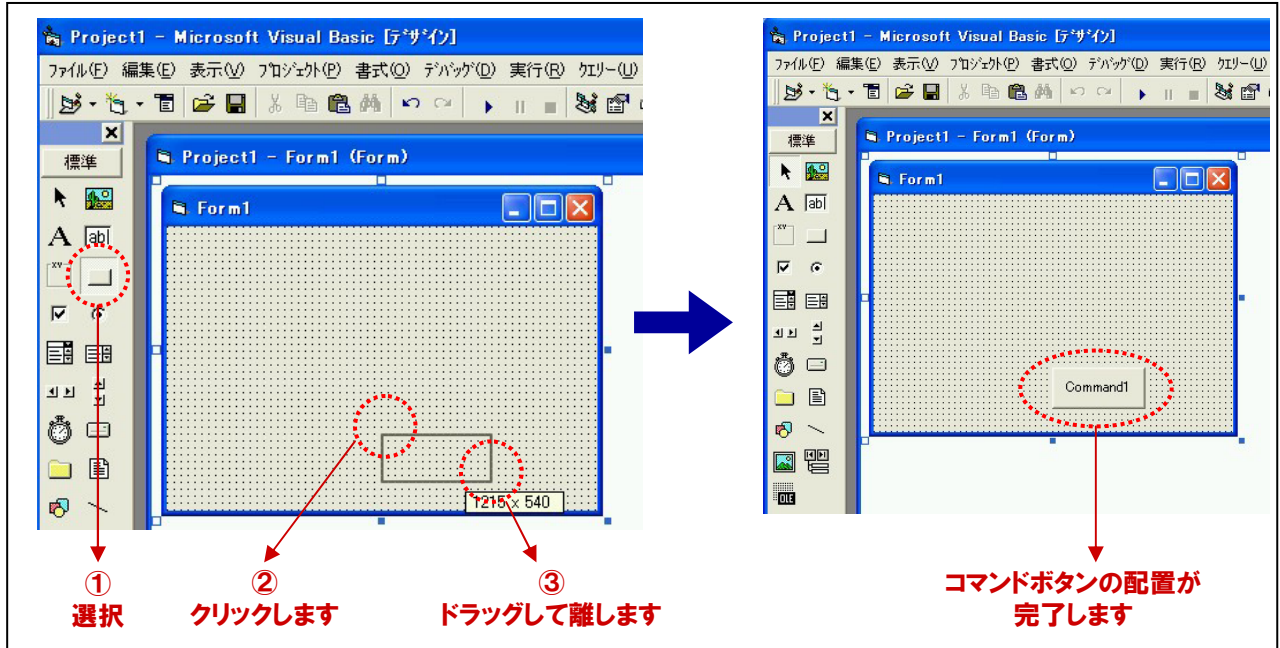
コントロール ツールボックス

プロパティウィンドウ

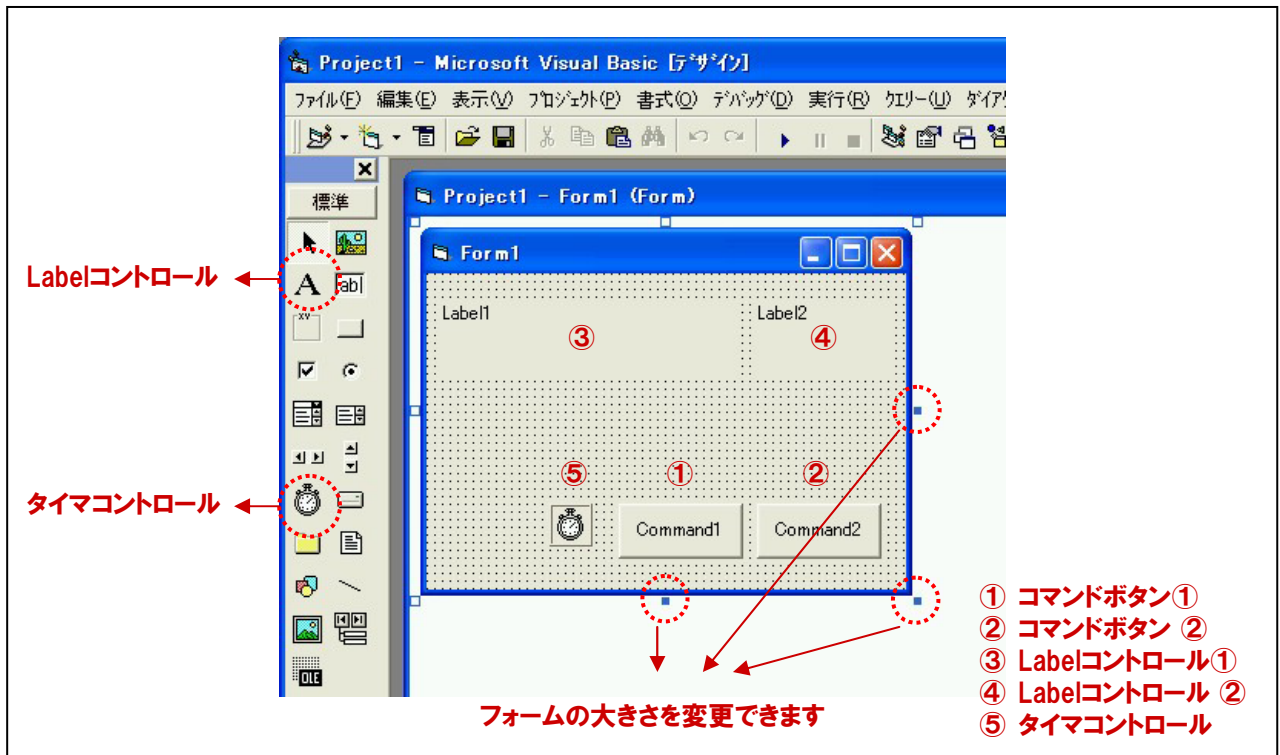
### 5-2-3.プログラム作成手順② (コントロールの配置)

本プログラムにて使用する『コントロール』をフォーム上に配置します。まず最初に『コマンドボタン コントロール』をフォーム上に配置しましょう。

- ① コントロールツールボックスの『コマンドボタン』のアイコンをクリックします。
- ② フォーム上にマウスカーソルを移動してクリックし、クリックしたまま右斜め下方方向にドラッグ (移動) します。
- ③ クリックした左ボタンを離れた時点で、『コマンドボタン』が配置されます (大きさの変更も同様作業です)。
- ④ 配置後の移動は、『コマンドボタン』をクリックして選択した後、ドラッグ操作で任意の場所に移動できます。



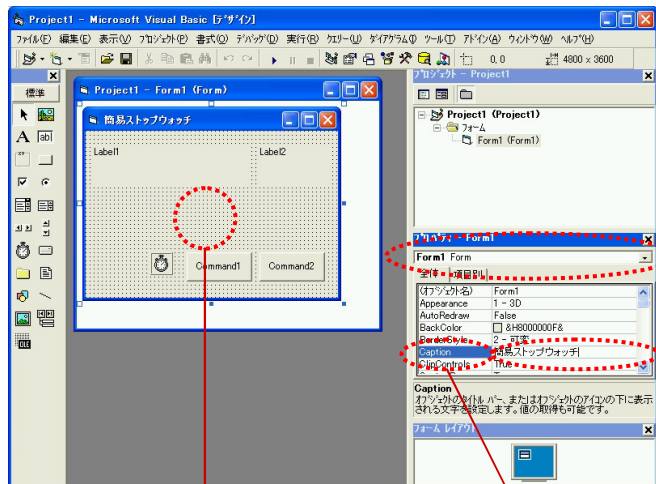
- ⑤ 同様の手順にて、2つ目の『コマンドボタン』と2つの『Labelコントロール』、『タイマコントロール』をフォーム上に配置します。



### 5-2-4.プログラム作成手順③ (フォームのプロパティ設定)

フォームおよび各コントロールのプロパティ設定を行います。プロパティの設定方法は、プロパティウィンドウで設定する方法と、コード(プログラム上)で設定する方法がありますが、本プログラムにおいてはすべてプロパティウィンドウで設定することとします。

フォームの『タイトル』を変更します。フォームをクリックして選択、プロパティウィンドウのプロパティリストの中から『Caption』プロパティを選択して、設定ボックスに『簡易ストップウォッチ』と入力します。



オブジェクトボックスに『Form』が表示されていることを確認してください。『Form1』はオブジェクト名です。オブジェクト名とは、このプロジェクト(プログラム)内で、このフォームを指定する際の名前です。

『簡易ストップウォッチ』と入力します。

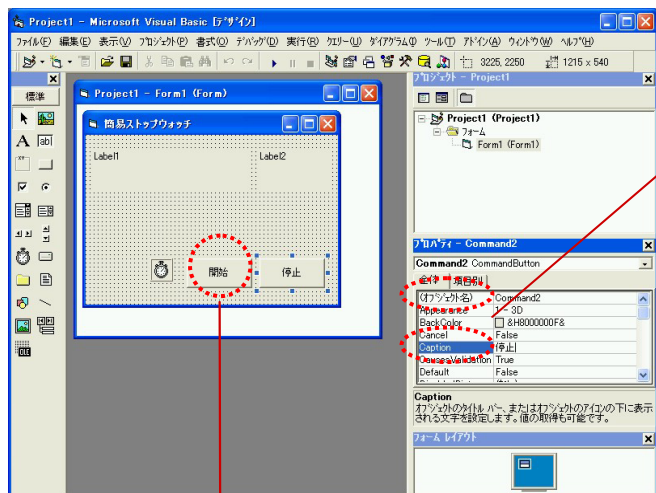
フォームをクリックし、  
フォームを選択状態にします。

『Caption』を選択します。

### 5-2-5.プログラム作成手順④ (コマンドボタンコントロールのプロパティ設定)

コマンドボタンの『オブジェクト名』と『タイトル』を変更します。コマンドボタン①を選択、プロパティウィンドウのプロパティリスト『(オブジェクト名)』を選択して、『cmdStart』と入力します。続いて、プロパティリストの中から、『Caption』プロパティを選択して、設定ボックスに『開始』と入力します。

同様にコマンドボタン②は、オブジェクト名に『cmdStop』、Captionプロパティに『停止』と入力します。



コマンドボタン①のプロパティ設定

- オブジェクト名 : cmdStart
- Caption : 開始

コマンドボタン②のプロパティ設定

- オブジェクト名 : cmdStop
- Caption : 停止

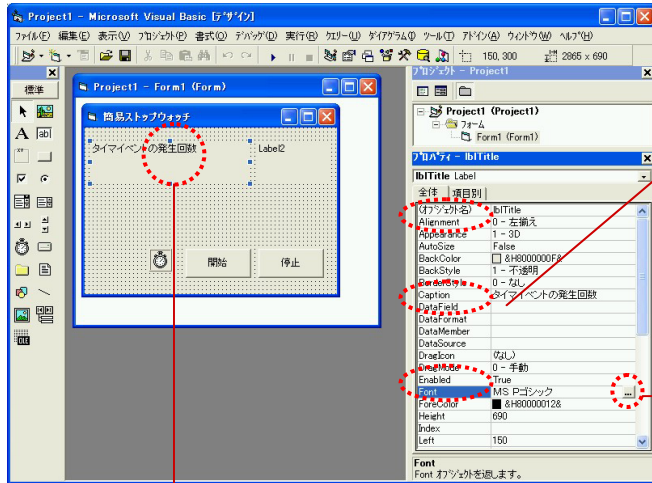
コマンドボタンをクリックし、  
選択状態にします。

## 5-2-6.プログラム作成手順⑤ (Labelコントロールのプロパティ設定)

Labelコントロールの『オブジェクト名』、『Caption』、『font』プロパティを変更します。

Labelコントロール①を選択、プロパティウィンドウのプロパティリスト『(オブジェクト名)』を選択して、『lblTitle』と入力します。続いて、プロパティリストの中から、『Caption』プロパティを選択して、設定ボックスに『タイマイイベントの発生回数』と入力します。最後に『font』プロパティを変更します。『font』プロパティを選択するとプロパティリスト中に□ボタンが表示されますので、そのボタンをクリックします。『フォントの設定ダイアログ』が表示されますので、好みのフォントスタイルに変更してください。

同様にLabelコントロール②は、オブジェクト名に『lblData』、Captionプロパティに『0』と入力します。



Labelコントロールをクリックし、  
選択状態にします。

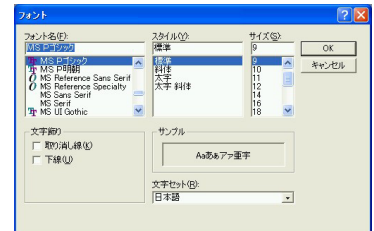
### Labelコントロール①のプロパティ設定

- ◎オブジェクト名 : lblTitle
- ◎Caption : タイマイイベントの発生回数
- ◎font : 任意

### Labelコントロール②のプロパティ設定

- ◎オブジェクト名 : lblData
- ◎Caption : 0
- ◎font : 任意

クリックすると『フォントの設定ダイアログ』  
が表示されますので、任意のフォントスタイルを設定してください。



## TOPICS

### 『Visual Basicのオブジェクト名の付け方 (プリフィックス) に関して』

Visual Basicのオブジェクト名の付け方に規則はありませんが、Microsoft社では、フォームやコントロールのオブジェクト名に関して総称 (プリフィックス) を用いることを推奨しています。これは、Visual Basicのマニュアルにも記載があります。頭三文字を小文字で特長付け、それぞれの役割がコード上で分かりやすくするためのものです。例えば、コマンドボタンの場合には『cmd』、テキストボックスであれば『txt』をオブジェクト名の先頭に付加します。そして、その後にそのオブジェクトがどんな目的のために使われるのかを、分かりやすくするような名前を付けます。その際には、1文字目を大文字にします。本プログラムでは、タイマの開始ボタンには『cmdStart』、停止ボタンには『cmdStop』と設定し、プログラム上でその役割が分かりやすくするようにしています。

### ■主なオブジェクト名とプリフィックスとの関係

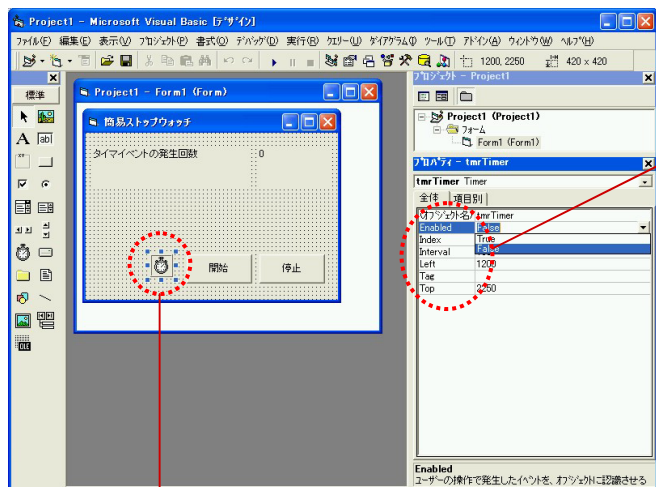
オブジェクト名	プリフィックス
フォーム	frm
コマンドボタン	cmd
ラベル	lbl
テキストボックス	txt
チェックボックス	chk

オブジェクト名	プリフィックス
リストボックス	lst
オプションボタン	opt
コンボボックス	cbo
イメージ	img
タイマ	tmr



## 5-2-7.プログラム作成手順⑥ (タイマコントロールのプロパティ設定)

タイマコントロールの『オブジェクト名』、『Enabled』、『Interval』プロパティを変更します。  
タイマコントロールを選択、プロパティウィンドウのプロパティリスト『(オブジェクト名)』を選択して、『tmrTimer』と入力します。続いて、プロパティリストの中から、『Enabled』プロパティを選択します。『True』または、『False』がコンボボックスから選択できるようになるので、『False』を選択します。  
最後に『Interval』プロパティに『100』を入力して完了です。



### タイマコントロールのプロパティ設定

- オブジェクト名 :tmrTimer
- Enabled :False
- Interval :100

タイマコントロールをクリックし、  
選択状態にします。

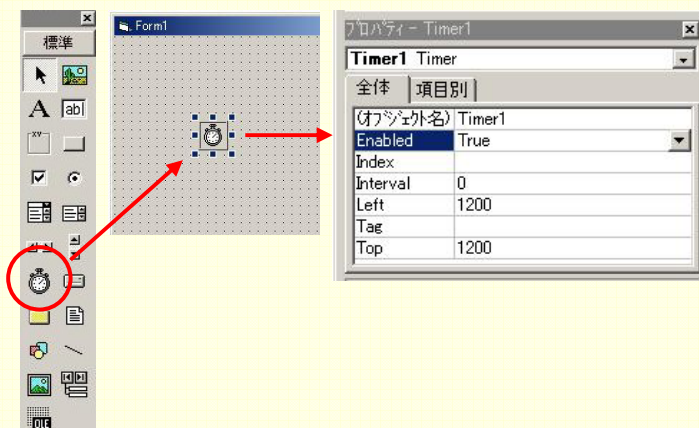
### TOPICS

#### 『タイマコントロールに関して』

タイマコントロールは、パソコンのシステムタイマを簡単に扱える機能を持った、Visual Basicの標準コントロールです。一定時間毎に処理を繰り返す場合などに使用します。

『Enabled』プロパティ: タイマコントロールの有効 (True) / 無効 (False) を設定します。初期値は『True』なので、初期値のままだとプログラム開始と同時にタイマが動いてしまいます。本プログラムでは、この初期値を『False (無効)』に設定し、『開始ボタン』で動くようにしています。

『Interval』プロパティ: タイマイベントが発生する間隔をmsec単位で設定します。『0~65535』の範囲で設定します。初期値は『0』なので、本プログラムでは、プログラム概要にのっとり、『100msec (0.1秒)』と設定しています。



## 5-2-8.プログラム作成手順⑦(タイマ開始処理の記述)

ここからは、簡易ストップウォッチの機能を実現するためのコード(プログラム)を記述していきます。先に述べましたが、Visual Basicは『イベント駆動型(イベントドリブン)言語』です。『何を(が)』、『何した時に』、『何をするか』をコードで表現します。【5-2-1.】のフローチャートに従って進めていきましょう。

まず最初に行いたい処理は、先のプロパティ設定で『False(無効)』に設定したタイマコントロールを起動、すなわち開始させることです。そのために用意したのが『開始ボタン(オブジェクト名:cmdStart)』です。この『開始ボタン』をクリックした時に『タイマコントロール(オブジェクト名:tmrTimer)』のEnabledプロパティを『False(無効)』から『True(有効)』に変える処理を記述します。フォーム上の『開始ボタン』をダブルクリックします。下記のようなコードウィンドウが開かれます。

① 『開始ボタン』をダブルクリックします。

② 『コードウィンドウ』が開きます。

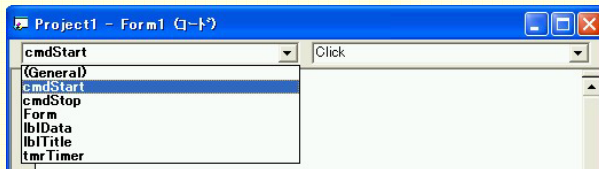
③ Private Sub cmdStart\_Click() から、End Subまでの間に、何を行いたいかをコードで記述します。

Private Sub cmdStart\_Click() からEnd Subまでの間に、下記のコードを記述します。『』から始まる部分は、コメント行として扱われますので、他の人がコードを見ても処理が分かりやすいようにコメントを書いておくのが望ましいです。文法は『何の』、『何を』、『何したいか』の順番で記述します。今回は、『タイマコントロール(tmrTimer)』の『Enabledプロパティ』を『True(有効)』にするという意味になります。

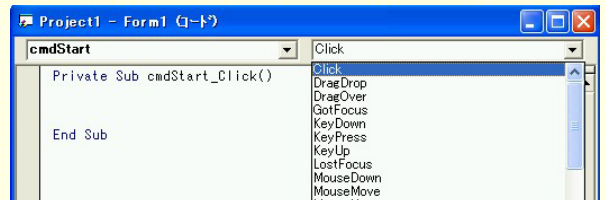
```
tmrTimer.Enabled = True
```

『タイマを起動します』

### TOPICS『オブジェクトの指定方法とプロシージャの指定方法』



オブジェクトボックスには、プロジェクトに含まれるすべてのオブジェクトを選択することができます。例えば『何を(が)』を選択する場所です。

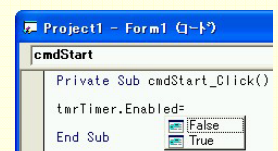
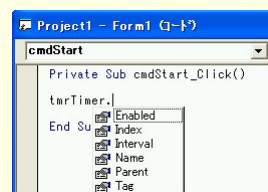


プロシージャボックスには、選択したオブジェクトが持っているプロシージャを選択することができます。例えば『何した時に』を選択する場所です。

### TOPICS

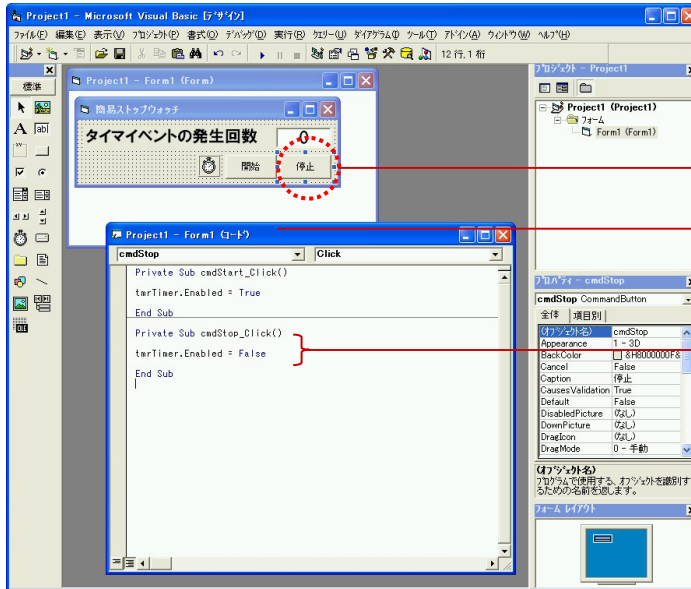
#### 『Visual Basicのインテリセンス機能』

インテリセンス機能は、オブジェクト名のあとに『.(ピリオド)』を入力すると、その後に入力候補がリストから選べる機能です。今回は“.”の後にもEnabledプロパティの『True/False』もリストから選べるようになっています。



## 5-2-9.プログラム作成手順⑧ (タイマ停止処理の記述)

次は、『開始ボタン』で有効にしたタイマコントロールを停止させるための『停止ボタン (cmdStop)』の処理を記述していきます。この『停止ボタン』をクリックした時に『タイマコントロール (オブジェクト名:tmrTimer)』のEnabledプロパティを『True (有効)』から『False (無効)』に変える処理を記述します。フォーム上の『停止ボタン』をダブルクリックします。下記のようなコードウィンドウが開かれます。



① 『停止ボタン』をダブルクリックします。

② 『コードウィンドウ』が開きます。

③ Private Sub cmdStop\_Click () から、End Subまでの間に、何を行いたいかをコードで記述します。

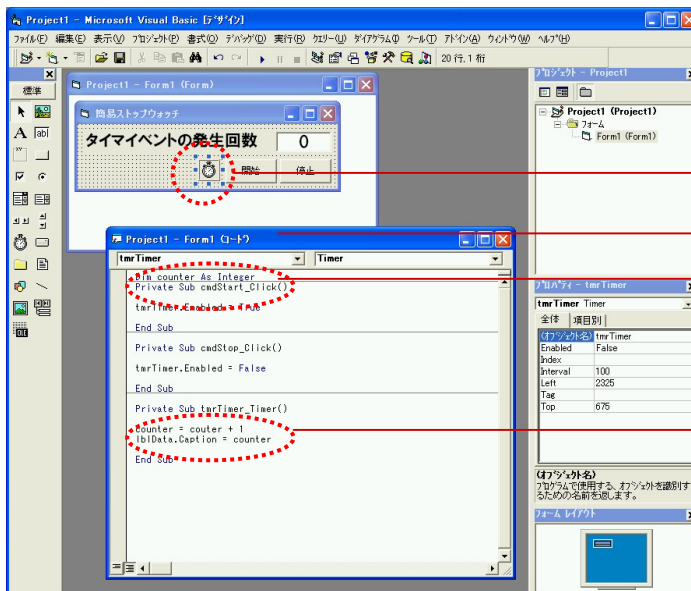
Private Sub cmdStop\_Click () からEnd Subまでの間に、下記のコードを記述します。『タイマコントロール (tmrTimer)』の『Enabledプロパティ』を『False (無効)』にするという意味です。

```
tmrTimer.Enabled = False
```

『タイマを停止します』

## 5-2-10.プログラム作成手順⑨ (変数の宣言とカウントアップ&表示処理の記述)

最後にタイマイベントが発生した際の処理を記述します。このタイマイベントは、タイマコントロールのEnabledプロパティが『True (有効)』の間、プロパティ『Interval』で設定した間隔毎に発生します。今回のプログラムでは、100msec (0.1秒) 毎に繰り返したい処理を記述します。タイマコントロール (tmrTimer) をダブルクリックして、コードウィンドウを開きます。



① 『タイマコントロール』をダブルクリックします。

② 『コードウィンドウ』が開きます。

③ 変数宣言を行います。

④ Private Sub tmrTimer\_Timer () から、End Subまでの間に、繰り返し何を行いたいかをコードで記述します。



繰り返しの処理の中で、変数を+1ずつカウントアップする処理があります。その変数を宣言（作成）します。タイマコントロール (tmrTimer) をダブルクリックして、コードウィンドウを開きます。コードウィンドウの一番上、または一番下の空白部分に下記のコードを記述します。今回は、『Counter』という名前の変数を、Integer型（整数型）で作成します。Private Sub\*\*\*~End Subの間以外で、この宣言を行うと、どのPrivate Subからも参照できるグローバル変数としての扱いとなります。

```
Dim Counter As Integer
```

'カウントアップ用変数

最後にPrivate Sub tmrTimer\_Timer () からEnd Subまでの間に、下記のコードを記述します。変数『Counter』のカウントアップ (+1の繰り返し) と『Counter』の現在値をLabelコントロール『lblData』に繰り返し表示するという処理です。

```
Counter = Counter + 1  
lblData.Caption = Counter
```

'変数に+1を行います  
'変数を画面表示します

### 5-2-11. カウントアッププログラムリスト

```
Dim Counter As Integer
```

'カウントアップ用変数

```
Private Sub cmdStart_Click ()
```

```
tmrTimer.Enabled = True
```

'タイマを起動します

```
End Sub
```

```
Private Sub cmdStop_Click ()
```

```
tmrTimer.Enabled = False
```

'タイマを停止します

```
End Sub
```

```
Private Sub tmrTimer_Timer ()
```

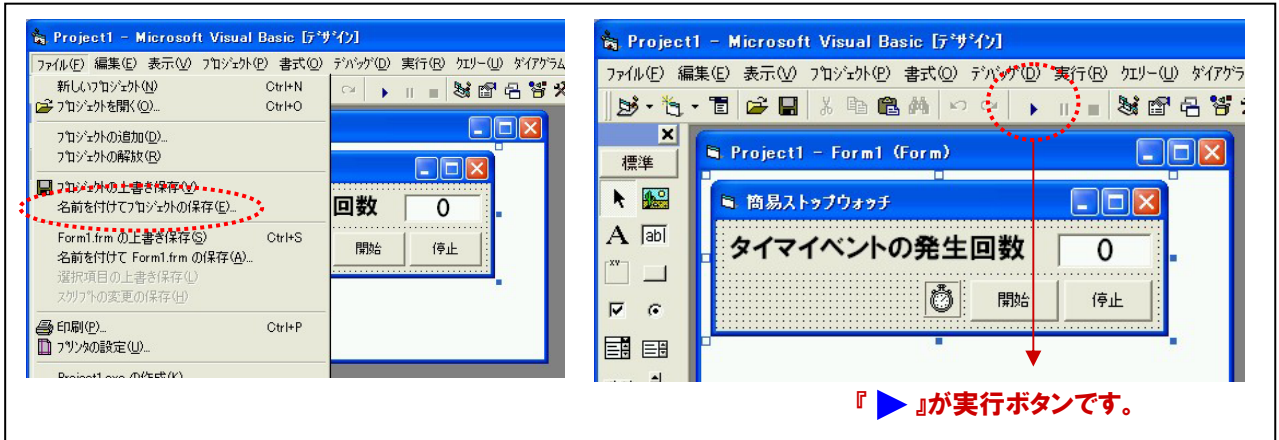
```
Counter = Counter + 1  
lblData.Caption = Counter
```

'変数に+1を行います  
'変数を画面表示します

```
End Sub
```

## 5-2-12.プログラム作成手順⑩ (プログラムの保存と動作の確認)

『ファイル』メニューの中から『名前を付けてプロジェクトの保存』を選択して、任意の場所に、今回作成したプロジェクトを保存します (\*\*\*.frmと\*\*\*.vbpの2つのファイルが保存されます)。保存が終わりましたら、ツールバーの実行ボタンをクリックした後、画面上の『開始ボタン』をクリックして実行してみましょう。



### TOPICS

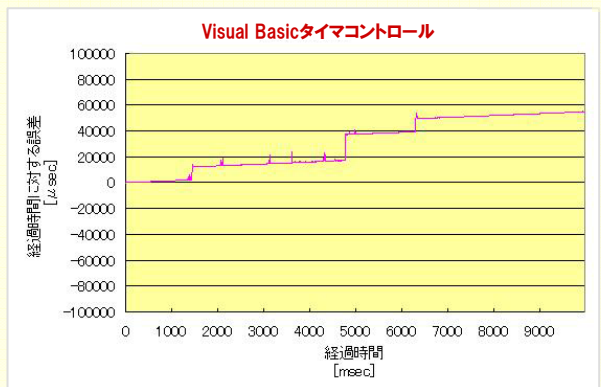
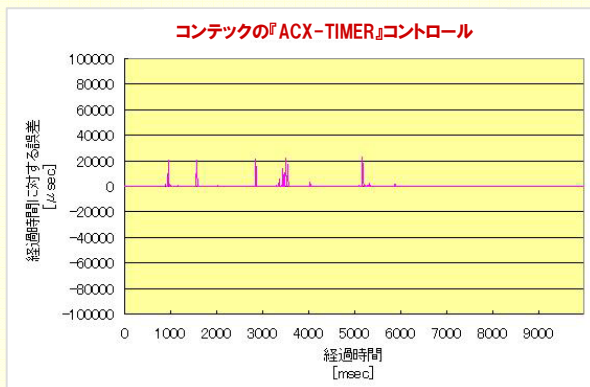
#### 『Visual Basicのタイマコントロールと弊社提供のタイマコントロールとの違い』

Visual Basicのタイマコントロールは、発生した誤差が累積されていくため、短い時間間隔では、正確な周期を得ることが難しいと言われています。これを解決する方法として、ボード上にタイマを搭載し高速・高精度の周期を得る、または独自のタイマコントロールを提供するなどの対策がとられています。

弊社では、オリジナルのタイマコントロール (ACX-TIMER) を製品添付ドライバソフトウェアCD-ROMに収録、または、弊社ホームページから無償ダウンロードにて提供しています。

#### ■参考資料

Visual Basicのタイマコントロールと弊社が提供しているオリジナルのタイマコントロール『ACX-TIMER』との誤差比較です。10msecインターバルのイベント発生中に、MS Wordを起動した場合の誤差 (正確な場合、経過時間に対する誤差が常に0になる) となります。実施環境は、Pentium III 1GHz/メモリ:128MB/OS: Windows2000/Visual Basic 6.0の環境です。



『ACX-TIMER』コントロールは、Visual Basic上のタイマコントロールの役割をするだけでなく、ボード/カード上に搭載されている高精度オンボードタイマを簡単に取り扱うことのできる優れたコントロールです。

『ACX-TIMER』コントロールのダウンロードは、弊社ドライバソフトウェア・デベロッパーズサイトから ↓

<http://www.contec.co.jp/apipac/>

## 5-3.ポート単位のデジタル入出力プログラム

タイマコントロールを使用したカウントアッププログラムの作成で、Visual Basicプログラミングとはどのように行うのかということが理解できたと思います。それでは、このカウントアッププログラムを利用して、デジタル入出力のプログラムを作成していきましょう。今回は、もっとも基本的な『ポート単位の入力と出力プログラム』を作成します。

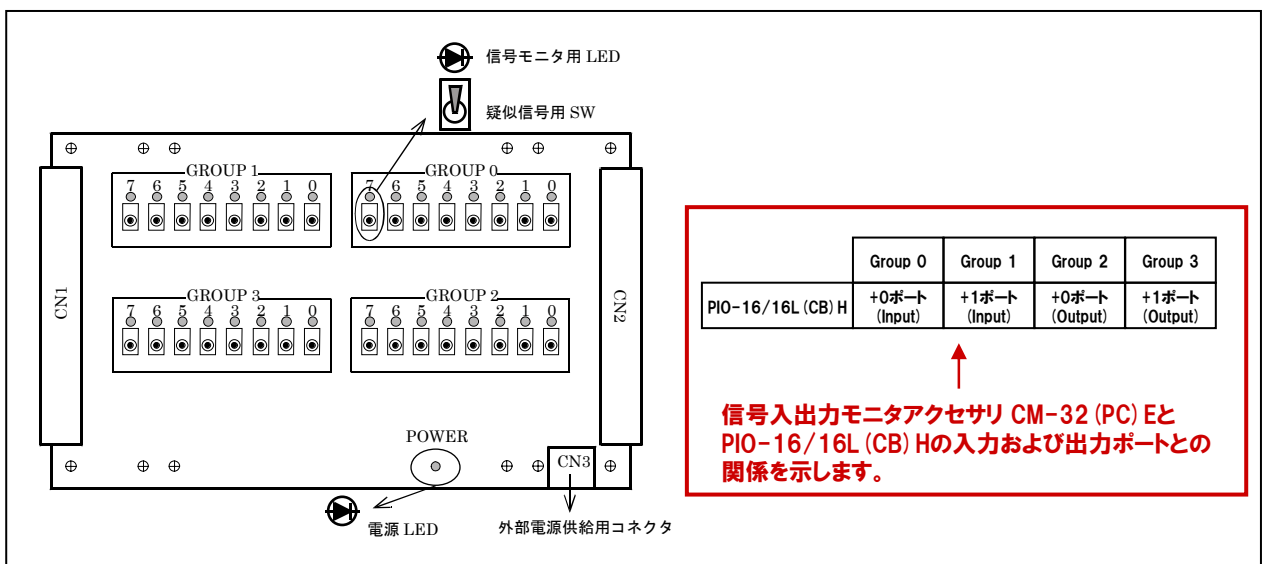
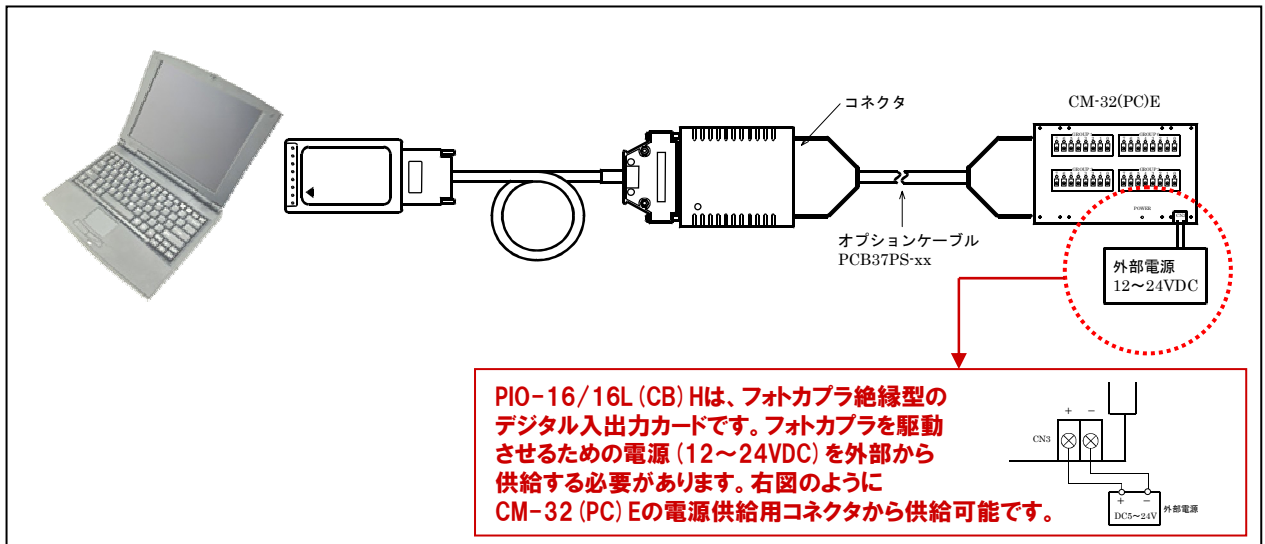
### 5-3-1.プログラム概要

100msec (0.1秒) の周期でデジタル入出力カード『PIO-16/16L (CB) H』の『入力+0ポート』からデータを入力、画面に16進数で表示します。

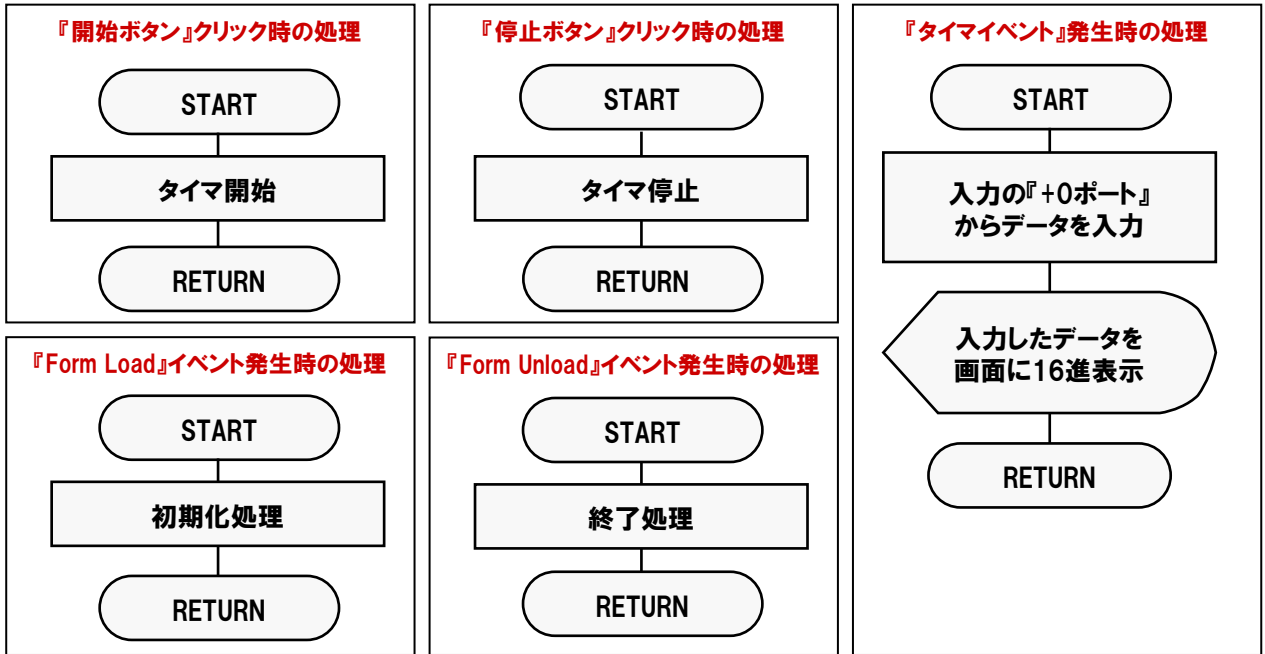


### 5-3-2.実行環境

【4-6.手順 ⑤:ハードウェア・ソフトウェアの動作確認】の環境をそのまま使用します。



### 5-3-3.プログラム フローチャート



### 5-3-4.Windows版ドライバソフトウェア『API-PAC (W32)』の基礎知識

本プログラムでは、弊社製デジタル入出力カードを使用して、外部からのデジタル信号入力およびパソコンから外部へのデジタル信号出力を行います。そのため、デジタル入出力ボード/カードを制御するためのドライバソフトウェアを使用します。インストールおよび初期設定 (API-TOOL Configurationの設定) は、第4章で完了しています。ここではWindows版ドライバソフトウェア『API-PAC (W32)』の紹介と、使用の際に必要な基礎知識を紹介します。

#### ①ドライバソフトウェア『API-PAC (W32)』とは

弊社製デジタル入出力ボード/カードへのコマンドをWindows標準のWin32 API関数 (DLL) の形式で提供するドライバソフトウェアです。Visual BasicやVisual C/C++などのWin32API関数をサポートした各種プログラミング言語で、弊社製ボード/カードの特色を活かした高速なアプリケーションソフトウェアの作成が可能となります。

#### ◎イベント駆動でのデータ収集が可能

イベントドリブン型制御が可能な関数をサポートしているため、ユーザー任意のタイミングでのデータ収集が可能です。

#### ◎論理デバイスでアクセス

プログラミングに対しては、煩わしいI/Oポートアドレスおよびボードに対するアクセスを意識せずにプログラミングが可能です。

#### ◎分かりやすい関数名称

各API関数は処理機能が分かる名称を用いており、読みやすいプログラムが実現します。

#### ◎診断プログラム

ハードウェアとドライバソフトウェアの状態を診断するプログラムが付属しています。ボード/カードのセットアップやドライバソフトウェアが正常かどうかを確認することができます。

#### ◎ボード自動検出機能

初期設定を行う『API-TOOL Configuration』は、ボード/カードの自動検出機能が可能です。パソコンに挿入されている弊社製ボード/カードを自動検出し、登録を行います。

#### ◎充実のサンプルプログラム

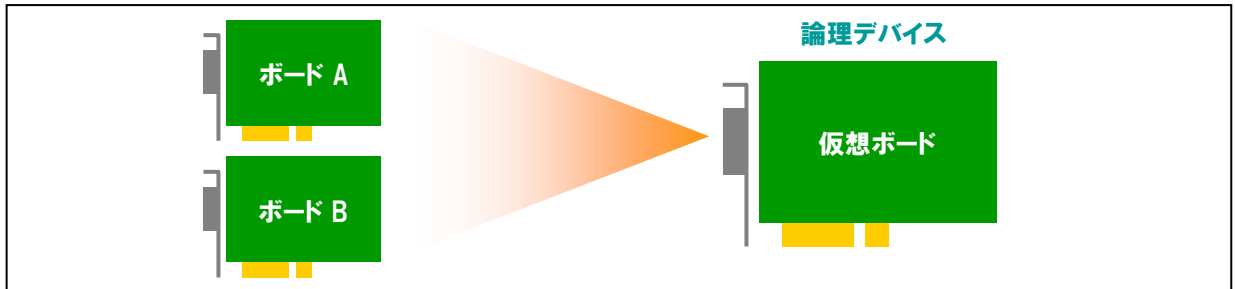
サポートする各言語に対応したサンプルプログラムが多数付属しています。関数の使い方を確認するだけでなく、ボードの実動作を確認できるなどアプリケーションの開発効率が向上します。

## ②初期設定ユーティリティ『API-TOOL Configuration』とは

ボード/カードが使用するI/Oアドレス、IRQ番号などのリソース情報をレジストリに登録するユーティリティです。ドライバソフトウェアは、レジストリに登録されたリソース情報を用いて、ボード/カードの制御を行います。ボード/カードを初めて使用する時や、設定を変更する際、初期設定ユーティリティを実行して初めてドライバソフトウェアが使用可能となります。

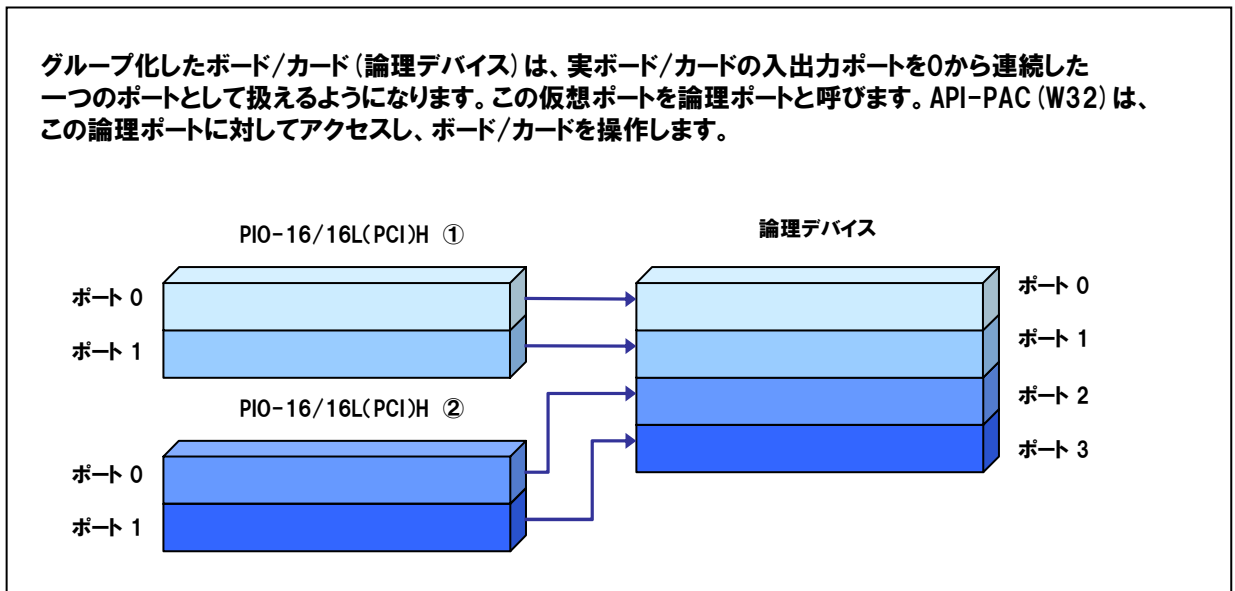
## ③論理デバイスとは

API-PAC (W32) は、複数のボード/カードをグループ化し、ソフトウェア (論理) 上で1つの仮想ボード/カードとして扱うことができます。この仮想ボード/カードを『論理デバイス』と呼んでいます。



## ④グループとは

『API-TOOL Configuration』で、使用するボード/カードを設定した論理デバイスを『グループ』と呼びます。グループには、デバイスドライバが各関数を実行するのに必要なボード/カードのリソース情報 (I/Oアドレス、割り込み) が設定されています。同じ型式のボード/カードのみ、グループ化が可能です。1グループで設定可能なボード/カードは最大4枚、1ドライバで管理可能なグループ数は、最大16グループです。

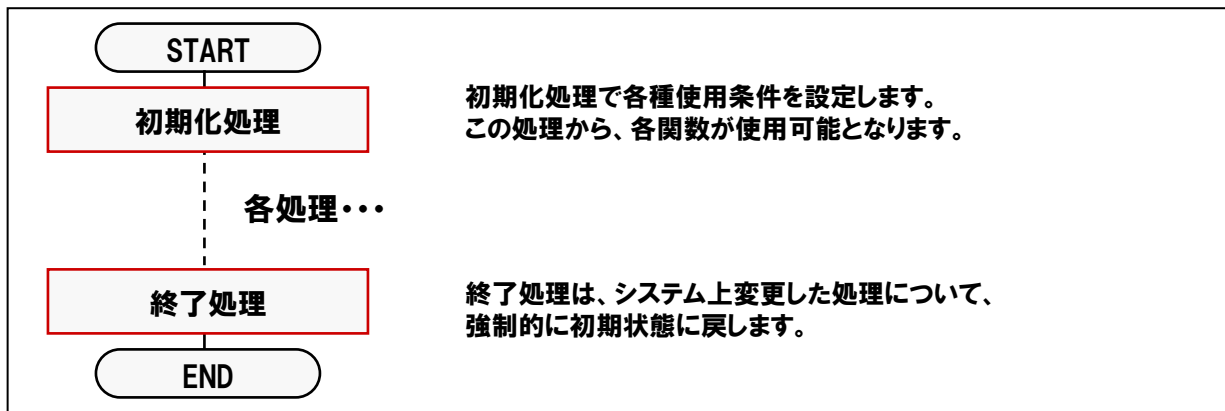


## ⑤ドライバ番号とは

インターフェイスとなるDLLは一つですが、下位では各ボード/カード種別に別々のデバイスドライバが分担して処理を行なっています。それらボード/カード種別毎のデバイスドライバに付けられる番号で、『API-TOOL Configuration』を実行すると自動で割り振られます。

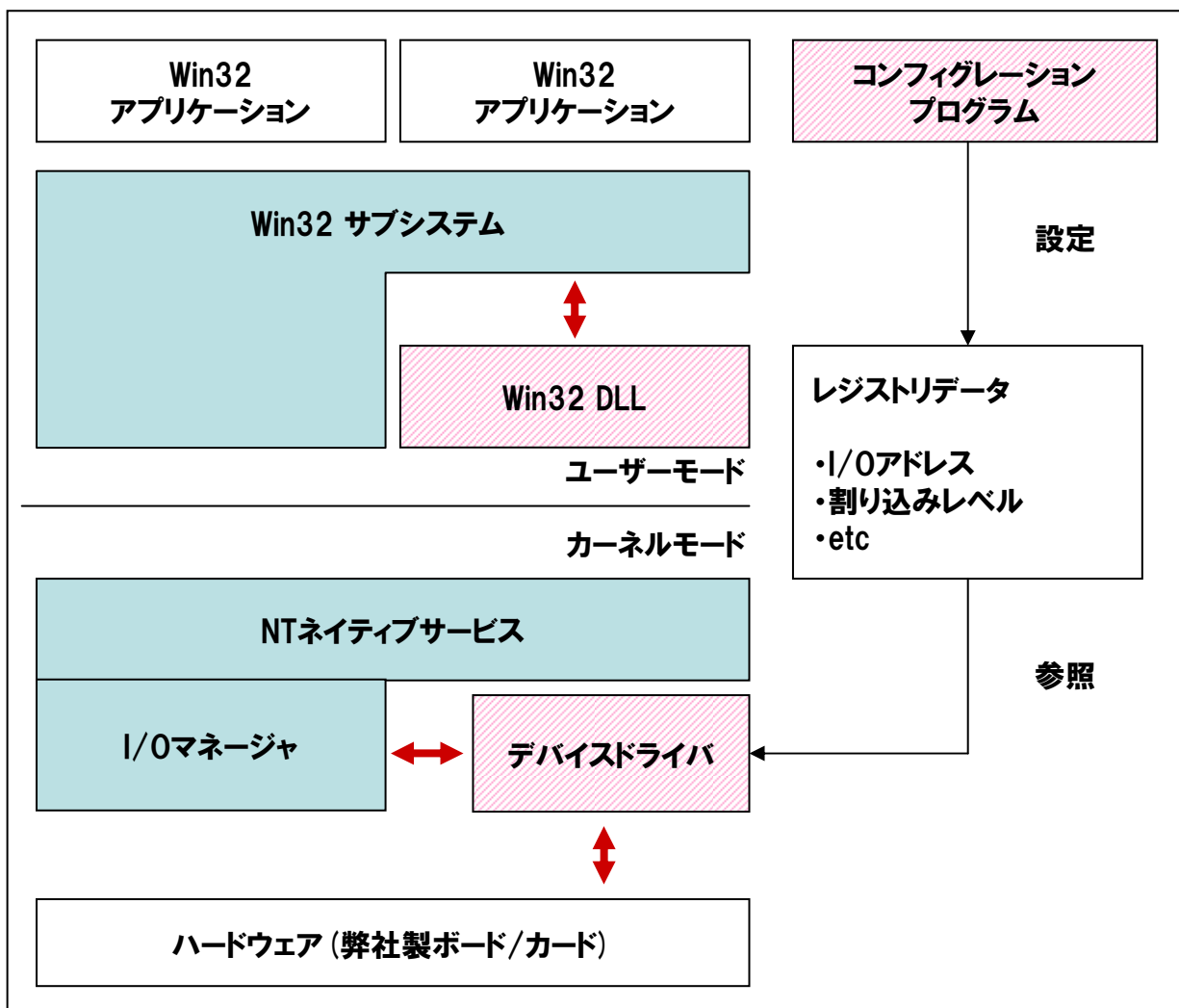
## ⑥API-PAC (W32) の処理体系

API-PAC (W32) は、初期化処理ではじまり、終了処理で終了する決まりがあります。初期化処理および、終了処理の専用関数を用意していますので、それぞれのタイミングでその関数を実行します。



## ⑦プログラム構成

Windows XP/2000/NTではアプリケーションからハードウェアに対して直接IN/OUTすることはできません。したがって、IN/OUTの処理はカーネルモードで動作するデバイスドライバ (\*.Sys) で行わなければなりません。アプリケーションからは、ドライバが提供する関数でDLL (Dynamic Link Library:[ 4-10参照 ]) を呼び出し、DLLがデバイスドライバ (\*.Sys) とのやりとりを行います。





### 5-3-5.プログラム作成手順① (画面の作成:オブジェクトの配置とプロパティ設定)

【5-2-2.】～【5-2-7.】を参考にして、オブジェクトの配置と各オブジェクトのプロパティ設定を行ってください。

- Form  
○Caption = 『ポート単位のデジタル入出力』
- Labelコントロール  
○オブジェクト名 = 『lblData』
- 開始ボタン  
○Caption = 『開始』  
○オブジェクト名 = 『cmdStart』
- 停止ボタン  
○Caption = 『停止』  
○オブジェクト名 = 『cmdStop』
- タイマコントロール  
○オブジェクト名 = 『tmrTimer』  
○Enabled = 『False』  
○Interval = 『100』
- Labelコントロール  
○オブジェクト名 = 『lblTitle』  
○Caption = 『入力+0ポートの値 (16進数)』

### 5-3-6.プログラム作成手順② (標準モジュールファイルの追加)

Visual BasicでAPI-PAC (W32) が提供する各関数を実行するには、使用する関数が参照するDLLの情報などが記述されている『標準モジュールファイル』をプロジェクトに追加する必要があります。

- ① Visual Basicのメニュー『プロジェクト』 - 『標準モジュールの追加』を選択します。
- ② ダイアログが表示されるので、『既存のファイル』のタブをクリックします。
- ③ 標準設定でインストールした場合には下記の場所に標準モジュールファイル (APIDIO.BAS) がありますので、選択して『開く』ボタンをクリックします。

**C:¥Program Files¥CONTEC¥API-PAC (W32) ¥Dio¥Samples¥Inc¥APIDIO.BAS**

- ④ プロジェクトエクスプローラ上に標準モジュールファイルが追加されていることを確認してください。

①

②・③

④

APIDIO.BASをダブルクリックで開くと、ドライバソフトウェアが提供している関数の情報(参照先)や、構造体の宣言などがされているのが分かります。



### 5-3-7.プログラム作成手順③ (変数の追加)

ポート単位のデジタル入出力プログラム中で使用する変数を宣言 (追加) します。変数名は任意ですが、変数の型 (整数型やバイト型) は、プログラム中で使用するドライバソフトウェアの関数の仕様に合わせて決定します。関数リファレンスは、インストールしたオンラインヘルプファイルに記載されていますので参照してください。

『スタートメニュー』 - 『プログラム』 - 『CONTEC API-PAC (W32)』 - 『DIO』 - 『API-DIO HELP』

① 『コードの表示』ボタンをクリックして、コードウィンドウを開きます。

② 空欄のコード記述欄に下記の変数を記述します。

Dim hDrv	As Long	'デバイスハンドル格納用変数
Dim Ret	As Long	'リターンコード (戻り値) 格納用変数
Dim InPort	As Integer	'入力ポート指定用
Dim InPortData	As Byte	'入力データ用
Dim DeviceName	As String	'デバイス名格納用変数

### 5-3-8.プログラム作成手順④ (初期化処理の追加)

API-PAC (W32) は初期化処理ではじまり、終了処理で終了する決まりがあります。初期化処理は、『Form』の『Form\_Loadイベント』内で、初期化処理関数を実行します。『Form\_Loadイベント』は『Form』がロード (立ち上がる) 際に発生するイベントで、各コントロールの既定値の設定や、変数を初期化する際に使われます。

① コードウィンドウを開き、『オブジェクト』から『Form』を選択します。

② 『プロシージャ』から『Load』を選択します。

③ Private Sub Form\_Load () から、End Subの間に初期化処理関数を記述します。

下記のコードを『Private Sub Form\_Load ()』から『End Sub』の間に記述してください。

```
DeviceName = "DIO00"
Ret = DioOpenEx (DeviceName, hDrv)
```

'デバイス名を変数に格納  
'初期化処理関数を実行

# 初期化処理関数 (デバイス名) 『DioOpenEx』 リファレンス

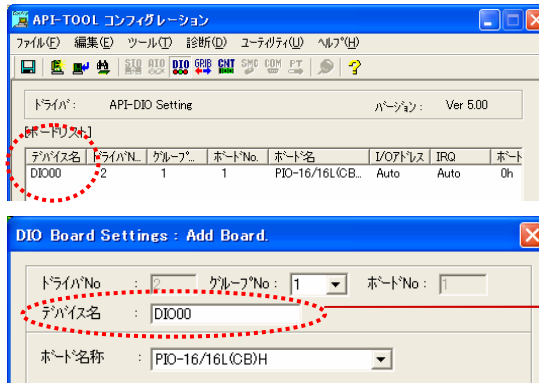
- 機能 使用するボード用のデバイスドライバをオープンし、ドライバの各関数を使用可能にします。
- 書式 Visual Basic6.0の場合

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

Ret = DioOpenEx (DeviceName,hDrv)
```

- 引数 hDrv : 使用するグループのデバイスハンドルが格納されます。他のDioXxxx関数を実行する時には、このデバイスハンドルを指定します。
- DeviceName : オープンするデバイス名を指定します。デバイス名は『API-TOOL Configuration』で設定します。

## デバイス名に関して



ドライバ番号、グループ番号で表される論理デバイスに対して付けられる名前をデバイス名と言います。デバイス名は『API-TOOL Configuration』で設定します。デバイス名は、[ボードの編集画面]で変更できますが、本書ではデフォルトの“DIO00”を使用します。

メニュー [編集] - [ボードの編集] の手順で、ボードの編集画面になります。この画面上で、デバイス名を任意に変更可能です。ただし、複数枚ボード/カードを使用する場合には重複しないように設定してください。

Ret: 終了情報 (戻り値) → 正常終了:0、エラー終了: 0以外 (詳細はヘルプの「戻り値一覧」参照)。

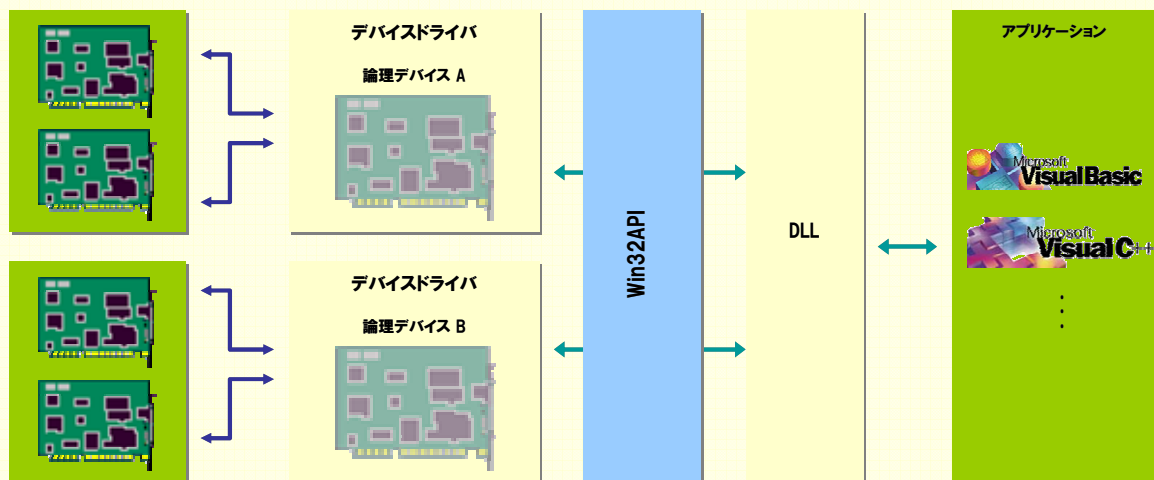
※ 本書では、戻り値の確認 (エラー処理) は、誌面の都合上、割愛しております。実システムにおいては、関数を実行した後にエラー処理のコードを記述します。エラー処理の方法は、各サンプルプログラムを参照してください。

## TOPICS

### 『デバイスハンドルとは』

Windows環境では、ハードウェアに対して直接アクセスすることはできません。このため、実際のハードウェアへのアクセスは論理デバイス (デバイスドライバ) が行います。アプリケーションからは、ドライバが提供しているDLL (Dynamic Link Library) の関数を呼び出し、DLLが論理デバイスとのやりとりを行い、ハードウェアにアクセスします。その際、DLLが個々の論理デバイス (デバイスドライバ) を識別するために、Windowsから割り当てられるID番号をデバイスハンドルと呼んでいます。

ドライバ初期化関数を実行すると、このID番号 (デバイスハンドル) をWindowsから取得後、変数に格納します。



## 5-3-9.プログラム作成手順⑤ (終了処理の追加)

API-PAC (W32) は初期化処理ではじまり、終了処理で終了する決まりがあります。終了処理は、『Form』の『Form\_Unloadイベント』内で、終了処理関数を実行します。『Form\_Unloadイベント』は『Form』が画面から消去(アンロード)される際に発生するイベントです。

① コードウィンドウを開き、『オブジェクト』から『Form』を選択します。

② 『プロシージャ』から『Unload』を選択します。

③ Private Sub Form\_Unload () から、End Subの間に終了処理関数を記述します。

下記のコードを『Private Sub Form\_UnLoad ()』から『End Sub』の間に記述してください。

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

終了処理関数実行 (デバイスハンドル開放)

## 終了処理関数 『DioClose』 リファレンス

- 機能 指定したデバイスの終了処理を行います。
- 書式 Visual Basic6.0の場合

```
Dim hDrv As Long
Dim Ret As Long
Ret = DioClose (hDrv)
```

- 引数 hDrv: 終了するデバイスハンドルを指定します。このデバイスハンドルはDioOpenで取得したものです。
- Ret: 終了情報 (戻り値) → 正常終了:0、エラー終了:0以外 (詳細はヘルプの「戻り値一覧」参照)。

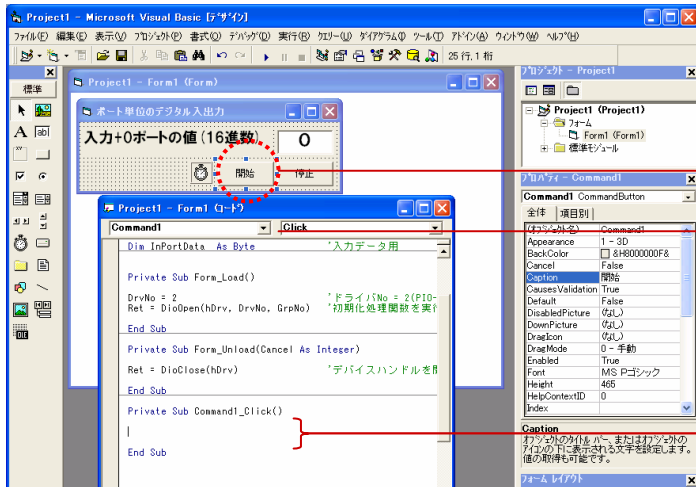
※ 本書では、戻り値の確認 (エラー処理) は、誌面の都合上、割愛しております。実システムにおいては、関数を実行した後にエラー処理のコードを記述します。エラー処理の方法は、各サンプルプログラムを参照してください。

- 補足 終了処理実行後は、指定グループに対して、(DioOpen以外の)ドライバの各関数は実行できません。

### 5-3-10.プログラム作成手順⑥ (タイマ開始処理の記述)

100msec (0.1秒) 毎にデジタル入力処理を行うために、【5-2.】にて使用したタイマコントロールを使用します。タイマコントロールは【5-2.】と同様に『開始ボタン』で開始、『停止ボタン』で停止の処理を行います。

先のプロパティ設定で『False (無効)』に設定したタイマコントロールを起動、すなわち開始させます。『開始ボタン』をクリックした時に『タイマコントロール (オブジェクト名: tmrTimer)』のEnabledプロパティを『False (無効)』から『True (有効)』に変える処理を記述します。フォーム上の『開始ボタン』をダブルクリックします。下記のようなコードウィンドウが開かれます。



① 『開始ボタン』をダブルクリックします。

② 『コードウィンドウ』が開きます。

③ Private Sub cmdStart\_Click () から、End Subまでの間に、何を行いたいかをコードで記述します。

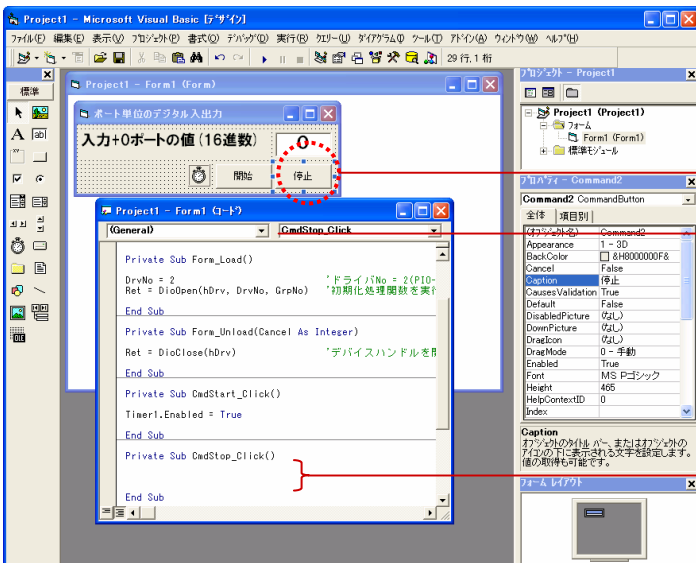
Private Sub cmdStart\_Click () からEnd Subまでの間に、下記のコードを記述します。

`tmrTimer.Enabled = True`

『タイマを起動します

### 5-3-11.プログラム作成手順⑦ (タイマ停止処理の記述)

『開始ボタン』で有効にしたタイマコントロールを停止させるための『停止ボタン (cmdStop)』の処理を記述していきます。この『停止ボタン』をクリックした時に『タイマコントロール (オブジェクト名: tmrTimer)』のEnabledプロパティを『True (有効)』から『False (無効)』に変える処理を記述します。フォーム上の『停止ボタン』をダブルクリックします。下記のようなコードウィンドウが開かれます。



① 『停止ボタン』をダブルクリックします。

② 『コードウィンドウ』が開きます。

③ Private Sub cmdStop\_Click () から、End Subまでの間に、何を行いたいかをコードで記述します。

Private Sub cmdStop\_Click () からEnd Subまでの間に、下記のコードを記述します

`tmrTimer.Enabled = False`

『タイマを停止します

## 5-3-12.プログラム作成手順⑧ (デジタル入力処理の追加)

デジタル入力カード PIO-16/16L (CB) H (またはデモボード) の『入力+0ポート』から、デジタルデータの入力を行う処理を追加します。ポート単位の入力関数がドライバソフトウェアで提供されていますので、その関数を実行します。下記の3行を記述するだけで、100msec (0.1秒) 毎に指定したポートからデジタル入力が繰り返されます。

① 『タイマコントロール』をダブルクリックします。

② 『コードウィンドウ』が開きます。

③ Private Sub tmrTimer\_Timer () から、End Subの間に記述します。  
Labelコントロール (lblData) に入力+0ポートのデータ『InPortData』を表示する際、Visual Basicの関数『HEX』を使用します。『HEX』を使用することにより、16進数にてデータを表示することができます。

Private Sub tmrTimer\_Timer () から、End Subの間に下記3行を記述します。

```
InPort = 0 '入力ポート指定用変数に"0"を設定
Ret = DioInpByte (hDrv, InPort, InPortData) '入力0ポートのデータを変数InPortDataに格納
lblData.Caption = Hex (InPortData) 'lblDataに入力データ (InPortData) を表示
```

## ポート単位のデジタル入力関数 『DioInpByte』 リファレンス

- 機能 指定した1ポート (8ビット) からデジタルデータの入力を行い、そのデータを変数に格納します。
- 書式 Visual Basic6.0の場合

```
Dim hDrv As Long
Dim InPort As Integer
Dim InPortData As Byte
Dim Ret As Long

Ret = DioInpByte (hDrv, InPort, InPortData)
```

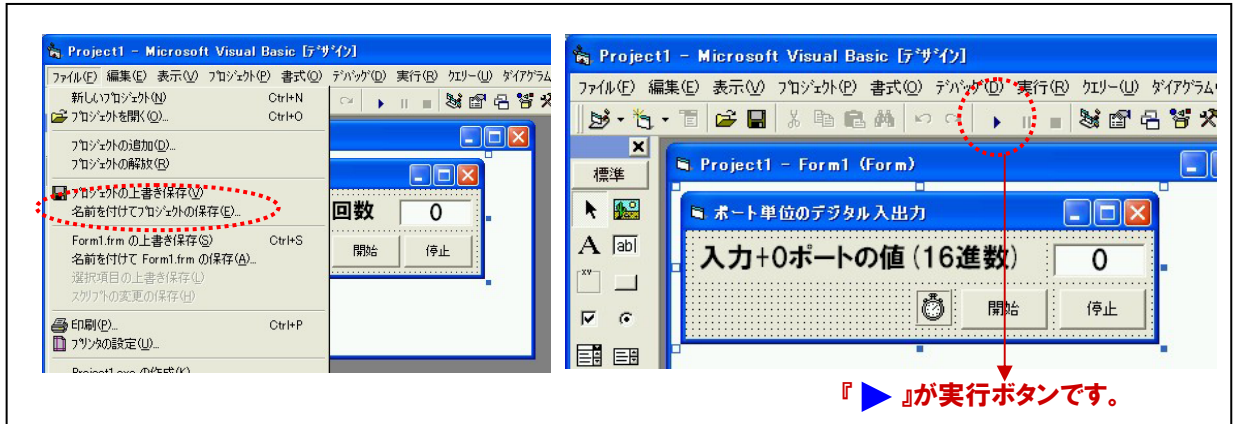
- 引数
  - hDrv: デバイスハンドルを指定します。このデバイスハンドルはDioOpenで取得したものです。
  - InPort : デジタル入力を行う論理ポート番号を指定します。  
指定可能な範囲:  $0 \leq \text{InPort} \leq \text{論理ポートの最大値} - 1$   
設定範囲外である場合、戻り値606hを返します。
  - InPortData: デジタル入力データを格納する変数を指定します。
  - Ret: 終了情報 (戻り値) → 正常終了: 0、エラー終了: 0以外 (詳細はヘルプの「戻り値一覧」参照)。

※ 本書では、戻り値の確認 (エラー処理) は、誌面の都合上、割愛しております。実システムにおいては、関数を実行した後にエラー処理のコードを記述します。エラー処理の方法は、各サンプルプログラムを参照してください。

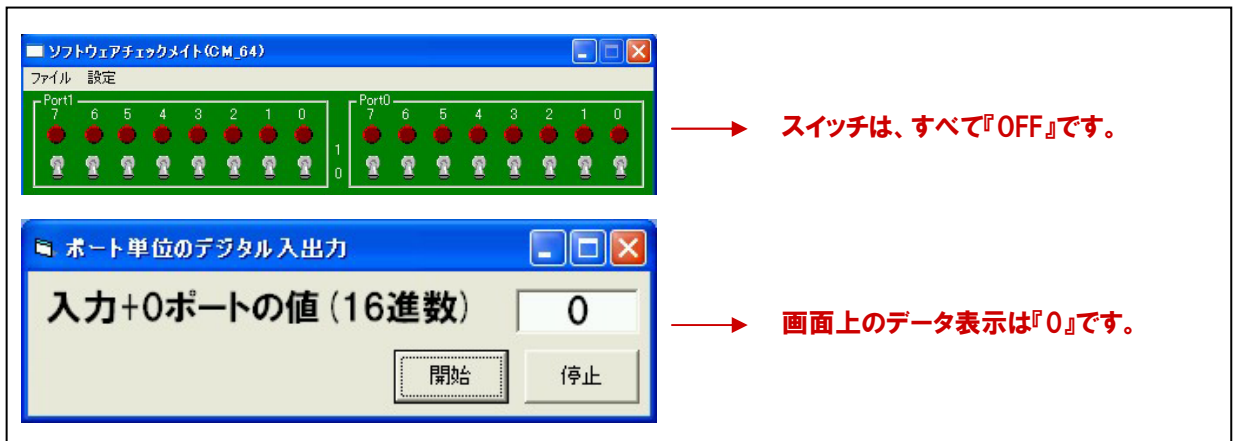


### 5-3-13.プログラムの実行

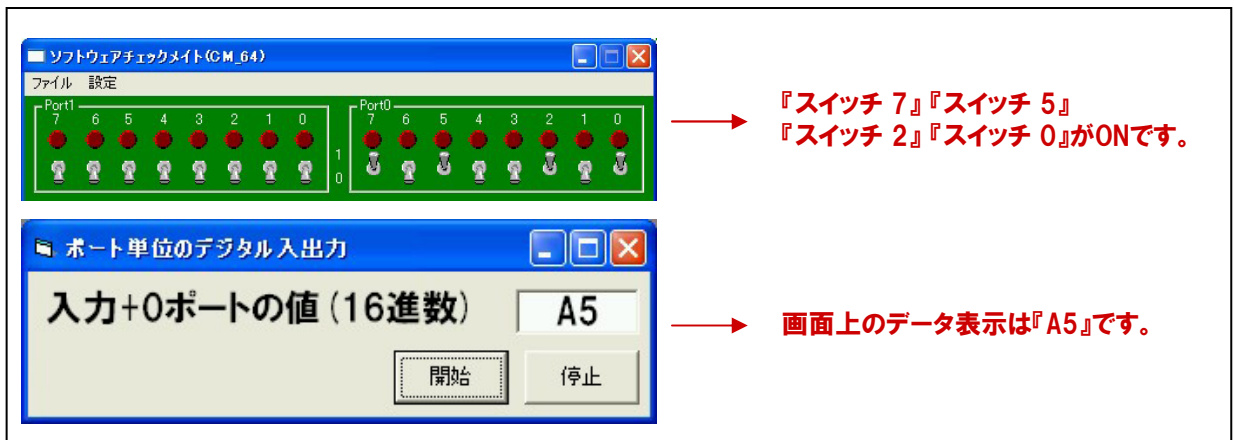
- ① 『ファイル』メニューの中から『名前を付けてプロジェクトの保存』を選択して、任意の場所に、任意のプロジェクト名称にて今回作成したプロジェクトを保存します。保存が完了したら、ツールバーの実行ボタンをクリックし、画面上の『開始ボタン』をクリックして実行してみましょう。



- ② チェックメイト (CM-32 (PC) E) の『GROUP 0』のスイッチが入力+0ポートに割り付けられています。スイッチは上が『ON』で下が『OFF』です。スイッチをすべて『OFF』にすると、Ibldataには『0』が表示されていると思います。なお、ここからのスイッチやLED (ランプ) の状態を表すのに、便宜上ソフトウェアチェックメイト (CM\_64.EXE) の画面を使用することとします。

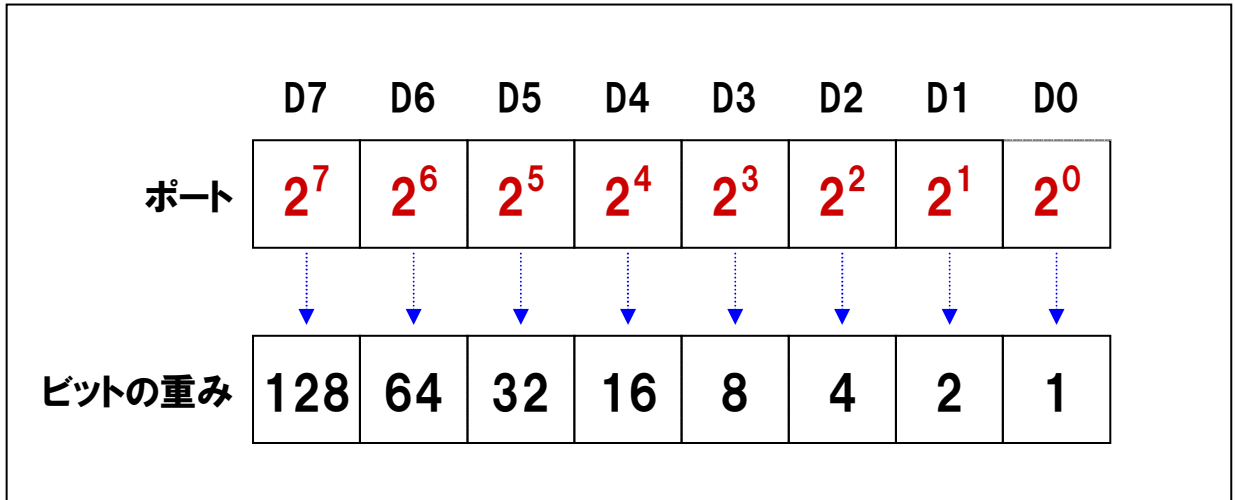


- ③ 次に『GROUP 0 (Port0)』の、『スイッチ 7』『スイッチ 5』『スイッチ 2』『スイッチ 0』を『ON』にしてみましょう。画面上には『A5』というデータが表示されるはずですが。

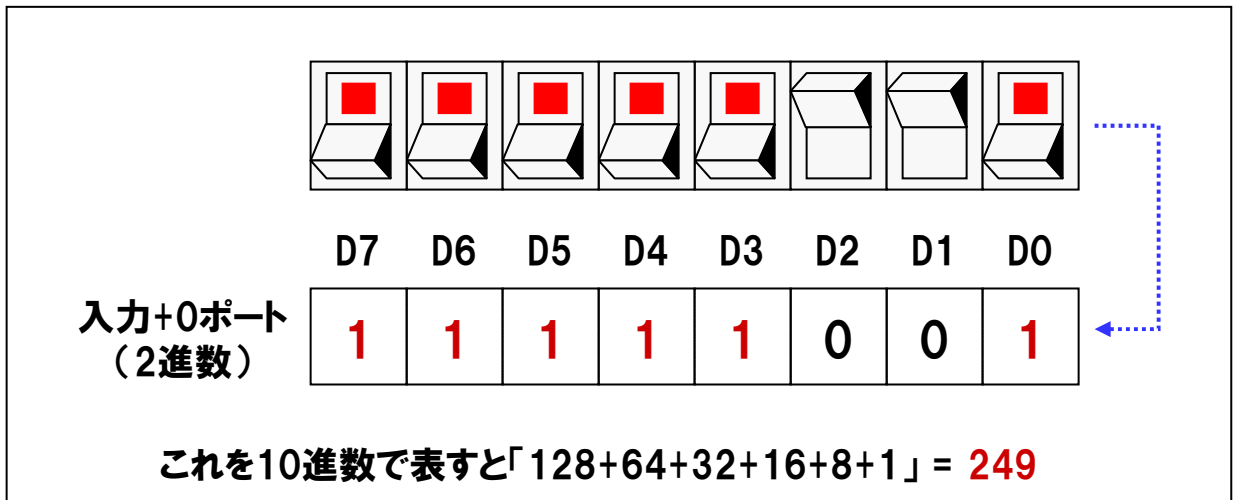




- ④ ここで、『I/Oポート』に関する説明をしておきましょう。なぜ、スイッチの7、5、2、0を『ON』した際、画面上に『A5 (16進数)』が表示されたのでしょうか。『I/Oポート』の1アドレス (1ポート) は8ビットで構成されていることは先程ご説明したとおりです。そして、それぞれのビットは数値 (10進数) による『重み付け』がされています。



例えば、『入力+0ポート』に接続されているスイッチの状態が以下であれば、

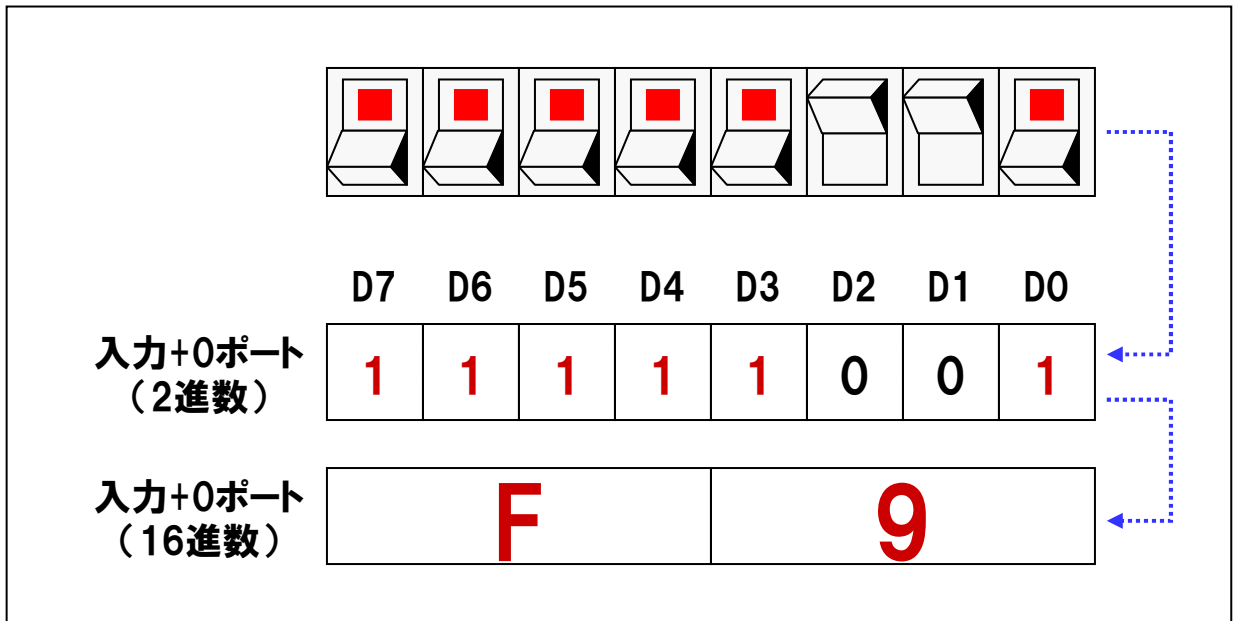


『1』が立っている場所を足すと『 $128+64+32+16+8+1 = 249$ 』となります。2進数 (バイナリ) 表現は、コンピュータにとっては扱いやすいですが、人間にとっては扱いにくい表現です。そこで、『0』と『1』の2進数から10進数への変換を効率良く行うため、コンピュータの世界では2進数を16進数で表現することが多く用いられます。長いビット列の2進数を表現する場合は、4ビットずつに区切り、対応する16進数で表記する方法が一般的に用いられます。ですから、スイッチの7、5、2、0を『ON』した場合のデータは『A5 (16進数)』となっていたのです。

⑤ 10進数・16進数・2進数との関係を示します。

10進数	16進数	2進数 (8ビット)	10進数	16進数	2進数 (8ビット)
0	00	0000 0000	11	0B	0000 1011
1	01	0000 0001	12	0C	0000 1100
2	02	0000 0010	13	0D	0000 1101
3	03	0000 0011	14	0E	0000 1110
4	04	0000 0100	15	0F	0000 1111
5	05	0000 0101	16	10	0001 0000
6	06	0000 0110	17	11	0001 0001
7	07	0000 0111	18	12	0001 0010
8	08	0000 1000	19	13	0001 0011
9	09	0000 1001	20	14	0001 0100
10	0A	0000 1010	21	15	0001 0101

この関係をふまえると、前頁のスイッチの状態(データ)は、16進数で表現すれば『F9』となります。



### 5-3-14.参考資料 (ビットの操作)

【5-4.】ではビット単位デジタル入出力関数を使用して、ビット毎のデジタル入出力プログラムを作成しますが、参考資料として、ビットの操作 (特定ビットの抽出、ON/OFF判断、ON/OFF操作) について解説します。

#### ① 入力した1ポート分のデータ (バイトデータ) から特定のビットを抽出する操作

1ポート分のデータ (バイトデータ) から特定のビットを抽出するには論理演算の『AND (論理積)』を使用します。まず初めに、AND演算について説明しておきます。AND演算は、入力条件がすべて『1』の場合のみ、式の出力が『1』となります。どれか1つでも『0』ならば、出力は『0』になるという性質を持っています。回路で表現すると直列回路になります。

AND (論理積)  $X = (A \text{ AND } B)$

入力		出力	スイッチの直列回路
A	B	X	
0	0	0	
0	1	0	
1	0	0	
1	1	1	

前項で使用したデータの低位4ビット (D4~D0) を抽出してみましょう。その方法は、抽出したいビットのみを『1』にしたデータ『0Fh』とAND演算を行います。

	D7	D6	D5	D4	D3	D2	D1	D0
入力+0ポートのデータ	1	1	1	1	1	0	0	1
0Fh	0	0	0	0	1	1	1	1

---

ANDした結果 = 09h

	0	0	0	0	1	0	0	1
--	---	---	---	---	---	---	---	---

## ② ビットのON/OFF判断と、特定のビットのみを『0』にする操作

①で使用したAND演算を応用すると、特定のビットの状態 (ON (1) かOFF (0) かの判断) を知る事ができます。例えば、『入力+0ポート』のデータのD3ビットが、ONかOFFを調べるには、調べたいビットのみを『1』にしたデータとAND演算を行います。

入力+0ポートのD3ビットが、1すなわち『ON』だった場合には、演算結果は『08h』です。

	D7	D6	D5	D4	D3	D2	D1	D0
入力+0ポートのデータ	?	?	?	?	1	?	?	?
08h	0	0	0	0	1	0	0	0
<hr/>								
ANDした結果 = 08h	0	0	0	0	1	0	0	0

入力+0ポートのD3ビットが、0すなわち『OFF』だった場合には、演算結果は『00h』です。

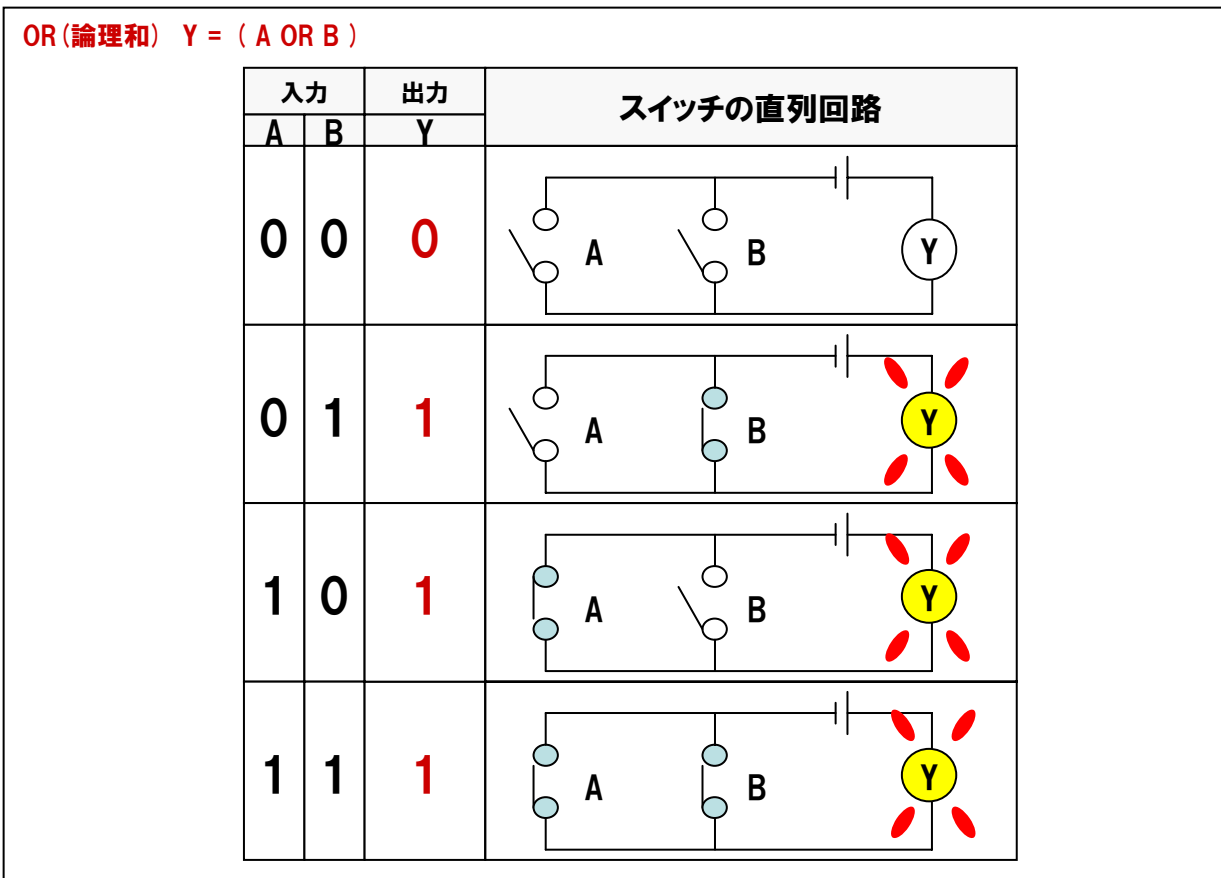
	D7	D6	D5	D4	D3	D2	D1	D0
入力+0ポートのデータ	?	?	?	?	0	?	?	?
08h	0	0	0	0	1	0	0	0
<hr/>								
ANDした結果 = 00h	0	0	0	0	0	0	0	0

AND演算を使用して、特定のビットのみを『0』、すなわち『OFF』にすることもできます。例えば、あるデータのD3ビットのみを『0』にしたい場合には、D3ビットに『0』を、その他すべてのビットを『1』にしたデータとAND演算を行います。結果、D3ビット以外は現在の状態のまま、D3ビットだけを『0』にしたデータが生成されます。

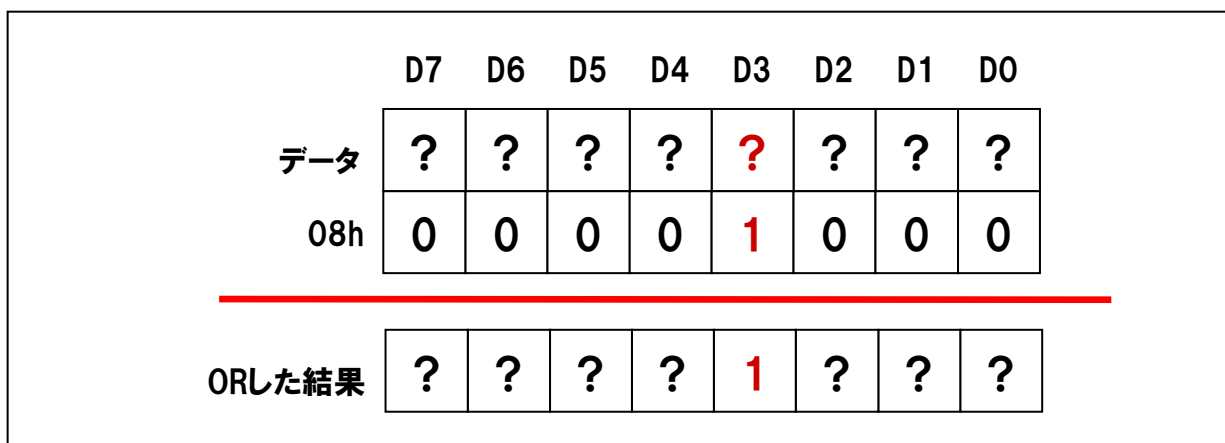
	D7	D6	D5	D4	D3	D2	D1	D0
データ	?	?	?	?	?	?	?	?
08h	1	1	1	1	0	1	1	1
<hr/>								
ANDした結果	?	?	?	?	0	?	?	?

③ 特定のビットのみを『1』にする操作

②では、AND演算を使用して、特定のビットを『0』にしましたが、論理演算のOR (論理和) 演算を利用して、特定のビットのみを『1』にする方法を解説します。『OR (論理和)』は、入力がすべて『0』の場合のみ、式の出力が『0』になります。どれか1つでも『1』があれば、出力は『1』になります。回路で表現すると並列回路です。

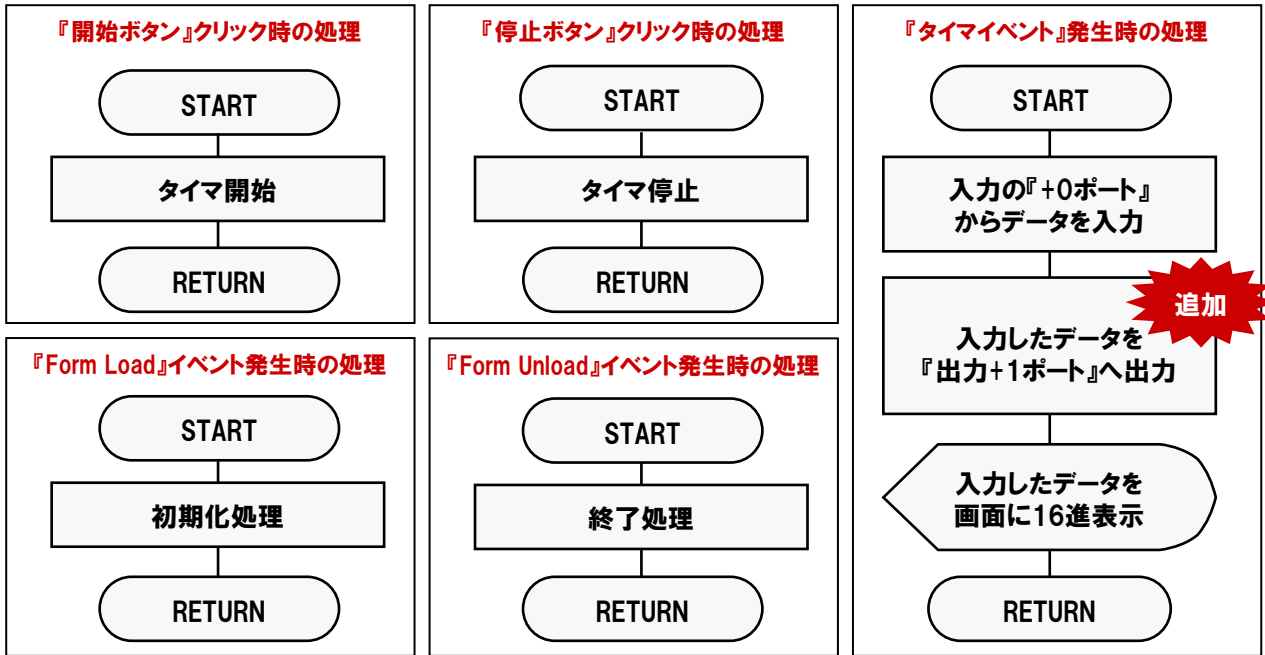


例えば、あるデータのD3ビットのみを『1』にしたい場合には、D3ビットに『1』を、その他すべてのビットを『0』にしたデータとOR演算を行います。D3ビットは、元のデータが0であっても1であっても、結果は必ず1になります。その他のビットは、現在の状態のままです。



## 5-3-15.デジタル出力処理の追加

最後にPIO-16/16L (CB) H (またはデモボード) の『出力+1ポート』へ、デジタルデータの出力を行う処理を追加します。ポート単位の出力関数がドライバソフトウェアで提供されていますので、その関数を実行します。作成したプログラムの『タイマイベント処理部:tmrTimer\_Timerイベント部』に処理を追加します。これにより、100msec (0.1秒) 毎に『入力+0ポート』からデータ (スイッチの状態) を入力し、そのデータをそのまま、『出力+1ポート』すなわちLEDに出力する処理が繰り返されます。



## ポート単位のデジタル出力関数 『DioOutByte』 リファレンス

- 機能 出力データが格納されている変数のデータをもとに、指定した1ポートへデジタルデータの出力を行います。
- 書式 Visual Basic6.0の場合

```
Dim hDrv           As Long
Dim OutPort       As Integer
Dim Buf           As Byte
Dim Ret           As Long
```

```
Ret = DioOutByte (hDrv,OutPort,Buf)
```

- 引数 hDrv: デバイスハンドルを指定します。このデバイスハンドルはDioOpenで取得したものです。
- OutPort: デジタル出力を行う論理ポート番号を指定します。  
指定可能な範囲:  $0 \leq \text{OutPort} \leq \text{論理ポートの最大値} - 1$   
設定範囲外である場合、戻り値606hを返します。
- Buf: デジタル出力データが格納されている変数を指定します。
- Ret: 終了情報 (戻り値) → 正常終了: 0、エラー終了: 0以外 (詳細はヘルプの「戻り値一覧」参照)。

※ 本書では、戻り値の確認 (エラー処理) は、誌面の都合上、割愛しております。  
実システムにおいては、関数を実行した後にエラー処理のコードを記述します。  
エラー処理の方法は、各サンプルプログラムを参照してください。



デジタル出力機能を追加した後の『変数宣言処理部』『タイマイベント処理部』のコードは下記のとおりになります。

■変数宣言処理部

Dim hDrv	As Long	'デバイスハンドル格納用変数
Dim Ret	As Long	'リターンコード(戻り値)格納用変数
Dim InPort	As Integer	'入力ポート指定用
Dim InPortData	As Byte	'入力データ用
Dim DeviceName	As String	'デバイス名格納用変数
Dim OutPort	As Integer	'出力ポート指定用



■タイマイベント処理部

InPort = 0	'入力ポート指定用変数に"0"を設定
Ret = DioInpByte (hDrv, InPort, InPortData)	'入力0ポートのデータを変数InPortDataに格納
OutPort = 1	'出力ポート指定用変数に"1"を設定
Ret = DioOutByte (hDrv, OutPort, InPortData)	'入力したデータをそのまま出力+1ポートへ出力
IblData.Caption = Hex (InPortData)	'IblDataに入力データ (InPortData) を表示



5-3-16.プログラムの実行

プログラムを保存した後、実行してみましょう。スイッチの状態(入力+0ポートのデータ)が『A5』のままならば、下記のように、CM-32(PC)Eの『GROUP 2』(ソフトウェアチェックメイトならば『Port1』)の『LED 0』、『LED 2』、『LED 5』、『LED 7』が点灯すると思います。出力ポートでは『1』を出力すると『ON』、『0』で『OFF』です。

ソフトウェアチェックメイト (CM\_64)

Port1: 7 6 5 4 3 2 1 0 (LEDs 0, 2, 5, 7 are lit)

Port0: 7 6 5 4 3 2 1 0 (LEDs 0, 2, 5, 7 are lit)

ポート単位のデジタル入出力

入力+0ポートの値 (16進数) A5

開始 停止

入力+0ポートの状態

D7	D6	D5	D4	D3	D2	D1	D0
1	0	1	0	0	1	0	1

出力+1ポートの状態

1	0	1	0	0	1	0	1
---	---	---	---	---	---	---	---

『スイッチ 7』『スイッチ 5』『スイッチ 2』『スイッチ 0』がONです。『LED 7』『LED 5』『LED 2』『LED 0』が点灯します。

画面上のデータ表示は『A5』です。

データをセット

## 5-3-17.ポート単位のデジタル入力 プログラムリスト

```
Dim hDrv      As Long      'デバイスハンドル格納用変数
Dim Ret       As Long      'リターンコード(戻り値)格納用変数
Dim InPort    As Integer   '入力ポート指定用
Dim InPortData As Byte     '入力データ用
Dim DeviceName As String   'デバイス名格納用変数
```

---

```
Private Sub Form_Load ()
```

```
    DeviceName = "DIO00"      'デバイス名を変数に格納
    Ret = DioOpenEx (DeviceName, hDrv) '初期化処理関数を実行
```

```
End Sub
```

---

```
Private Sub Form_Unload (Cancel As Integer)
```

```
    Ret = DioClose (hDrv)      '終了処理関数実行(デバイスハンドル開放)
```

```
End Sub
```

---

```
Private Sub cmdStart_Click ()
```

```
    tmrTimer.Enabled = True    'タイマを起動します
```

```
End Sub
```

---

```
Private Sub cmdStop_Click ()
```

```
    tmrTimer.Enabled = False   'タイマを停止します
```

```
End Sub
```

---

```
Private Sub tmrTimer_Timer ()
```

```
    InPort = 0                '入力ポート指定用変数に"0"を設定
    Ret = DioInpByte (hDrv, InPort, InPortData) '入力0ポートのデータを変数InPortDataに格納

    lblData.Caption = Hex (InPortData)      'lblDataに入力データ(InPortData)を表示
```

```
End Sub
```

## 5-3-18.ポート単位のデジタル入出力 プログラムリスト

Dim hDrv	As Long	'デバイスハンドル格納用変数
Dim Ret	As Long	'リターンコード(戻り値)格納用変数
Dim InPort	As Integer	'入力ポート指定用
Dim InPortData	As Byte	'入力データ用
Dim DeviceName	As String	'デバイス名格納用変数
Dim OutPort	As Integer	'出力ポート指定用

Private Sub Form\_Load ()

DeviceName = "DIO00"	'デバイス名を変数に格納
Ret = DioOpenEx (DeviceName, hDrv)	'初期化処理関数を実行

End Sub

Private Sub Form\_Unload (Cancel As Integer)

Ret = DioClose (hDrv)	'終了処理関数実行(デバイスハンドル開放)
-----------------------	-----------------------

End Sub

Private Sub cmdStart\_Click ()

tmrTimer.Enabled = True	'タイマを起動します
-------------------------	------------

End Sub

Private Sub cmdStop\_Click ()

tmrTimer.Enabled = False	'タイマを停止します
--------------------------	------------

End Sub

Private Sub tmrTimer\_Timer ()

InPort = 0	'入力ポート指定用変数に"0"を設定
Ret = DioInpByte (hDrv, InPort, InPortData)	'入力0ポートのデータを変数InPortDataに格納
OutPort = 1	'出力ポート指定用変数に"1"を設定
Ret = DioOutByte (hDrv, OutPort, InPortData)	'入力したデータをそのまま出力+1ポートへ出力
lblData.Caption = Hex (InPortData)	'lblDataに入力データ (InPortData) を表示

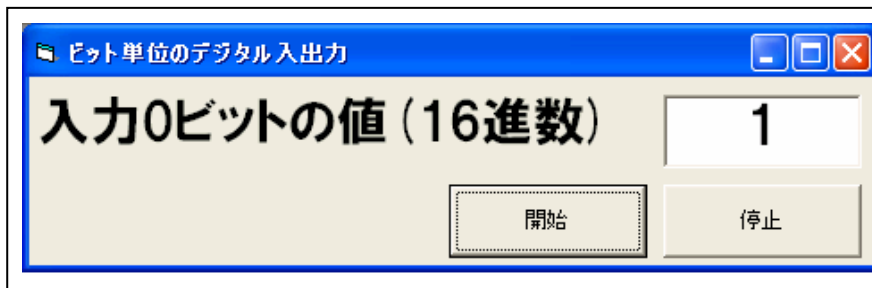
End Sub

## 5-4.ビット単位のデジタル入出力プログラム

デジタル信号の入出力は、今プログラミングを行った『ポート単位』で行う方法と『ビット単位』で行う方法があります。ドライバソフトウェアで『ビット単位の入力・出力』を行う関数が用意されておりますので、それらを使って『ビット単位の入出力プログラム』を作成してみましょう。実際には、先程作成した『ポート単位の入力と出力』プログラムの中で使用していた関数を置き換えるだけです。

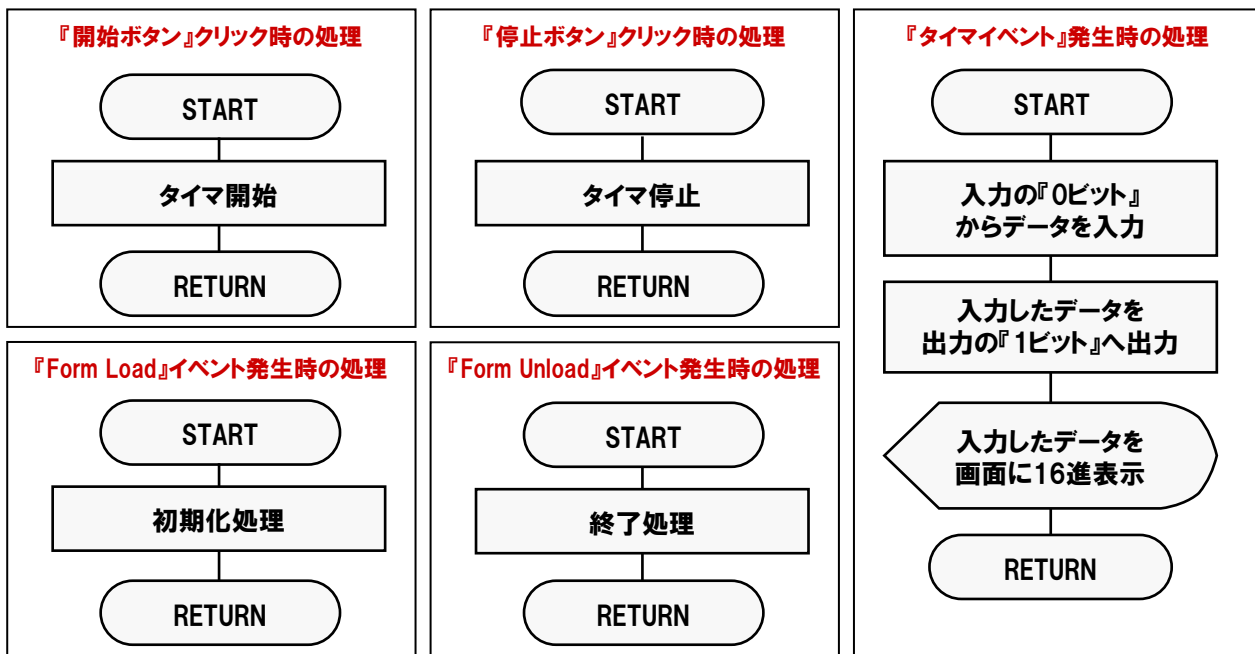
### 5-4-1.プログラム概要

100msec (0.1秒) の周期でPIO-16/16L (CB) Hの『入力0ビット』からデータを入力して、画面にデータを表示します。また、入力したデータをそのまま『出力1ビット』に出力します。

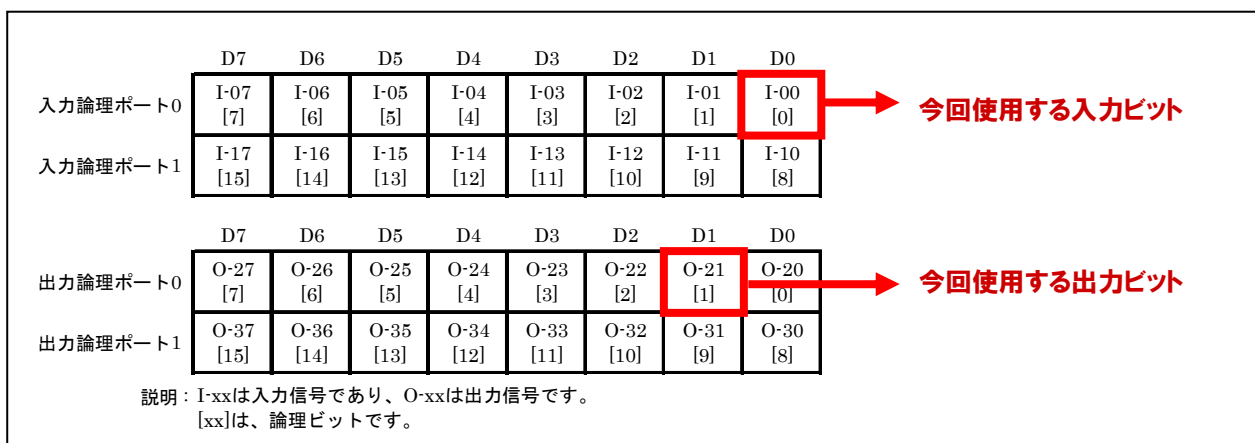


実行画面イメージ

### 5-4-2.プログラム フローチャート



### 5-4-3.PIO-16/16L (CB) Hのビット割付図



## 5-4-4.プログラム作成手順① (画面の作成:オブジェクトの配置とプロパティ設定)

【5-2-2.】～【5-2-7.】を参考にして、オブジェクトの配置と各オブジェクトのプロパティ設定を行ってください。

The screenshot shows a Visual Basic form titled 'ピット単位のデジタル入出力' (Digital I/O in Pit Units). The form contains several controls: a text box for '入力0ビットの値(16進数)' (Input 0 bit value in hexadecimal), two buttons labeled '開始' (Start) and '停止' (Stop), a timer control, and a title label. Red arrows point from the controls to their respective property settings listed in boxes on the right.

- Form**  
◎Caption = 『ピット単位のデジタル入出力』
- Labelコントロール**  
◎オブジェクト名 = 『lblData』
- 開始ボタン**  
◎Caption = 『開始』  
◎オブジェクト名 = 『cmdStart』
- 停止ボタン**  
◎Caption = 『停止』  
◎オブジェクト名 = 『cmdStop』
- タイマコントロール**  
◎オブジェクト名 = 『tmrTimer』  
◎Enabled = 『False』  
◎Interval = 『100』
- Labelコントロール**  
◎オブジェクト名 = 『lblTitle』  
◎Caption = 『入力0ビットの値(16進数)』

## 5-4-5.プログラム作成手順② (標準モジュールファイルの追加)

Visual BasicでAPI-PAC (W32) が提供する各関数を実行するには、使用する関数が参照するDLLの情報などが記述されている『標準モジュールファイル』をプロジェクトに追加する必要があります。

- ① Visual Basicのメニュー『プロジェクト』 - 『標準モジュールの追加』を選択します。
- ② ダイアログが表示されるので、『既存のファイル』のタブをクリックします。
- ③ 標準設定でインストールした場合には下記の場所に標準モジュールファイル (APIDIO.BAS) がありますので、選択して『開く』ボタンをクリックします。

**C:¥Program Files¥CONTEC¥API-PAC (W32) ¥Dio¥Samples¥Inc¥APIDIO.BAS**

- ④ プロジェクトエクスプローラ上に標準モジュールファイルが追加されていることを確認してください。

①

The screenshot shows the 'Project' menu open, with 'Add Standard Module' (標準モジュールの追加) selected. A red circle highlights this option.

②③

The screenshot shows the 'Add Standard Module' dialog box. The 'Existing Files' tab is active. 'APIDIO.BAS' is selected in the file list, and the 'Open' (開く) button is highlighted with a red circle.

④

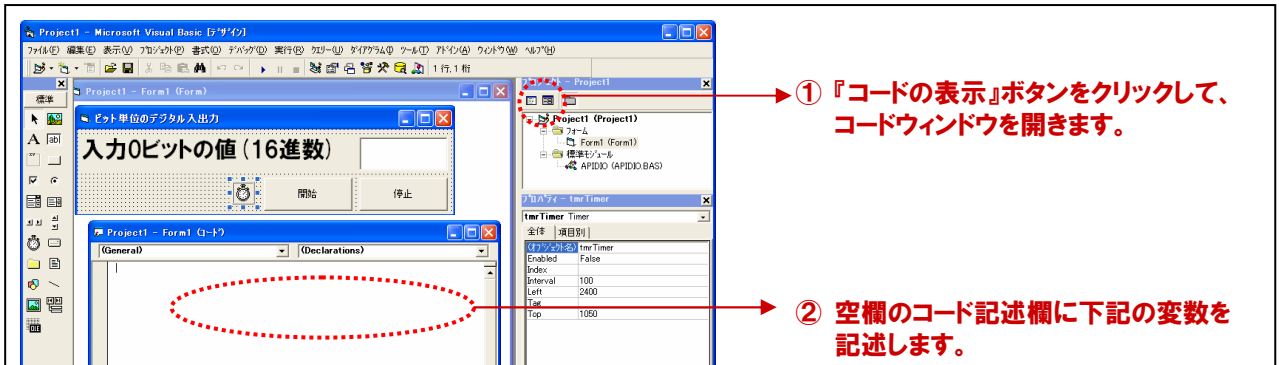
The top screenshot shows the Project Explorer with 'APIDIO.BAS' added to the project. The bottom screenshot shows the code window with the API-PAC declarations for the 'Dio' module, including functions like 'DioOpen', 'DioClose', 'DioRead', and 'DioWrite'.

APIDIO.BASをダブルクリックで開くと、ドライバソフトウェアが提供している関数の情報(参照先)や、構造体の宣言などがされているのが分かります。

### 5-4-6.プログラム作成手順③ (変数の追加)

ビット単位のデジタル入出力プログラム中で使用する変数を宣言 (追加) します。変数名は任意ですが、変数の型 (整数型やバイト型) は、プログラム中で使用するドライバソフトウェアの関数の仕様に合わせて決定します。関数リファレンスは、インストールしたオンラインヘルプファイルに記載されていますので参照してください。

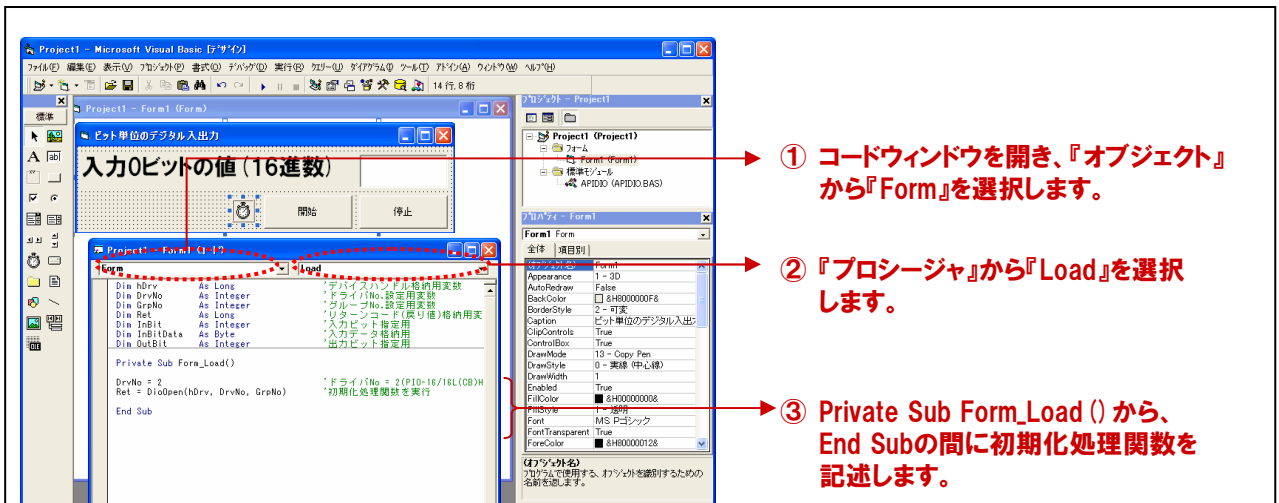
『スタートメニュー』 - 『プログラム』 - 『CONTEC API-PAC (W32)』 - 『DIO』 - 『API-DIO HELP』



Dim hDrv	As Long	'デバイスハンドル格納用変数
Dim Ret	As Long	'リターンコード (戻り値) 格納用変数
Dim InBit	As Integer	'入力ビット指定用
Dim InBitData	As Byte	'入力データ格納用
Dim OutBit	As Integer	'出力ビット指定用
Dim DeviceName	As String	'デバイス名格納用変数

### 5-4-7.プログラム作成手順④ (初期化処理の追加)

API-PAC (W32) は初期化処理ではじまり、終了処理で終了する決まりがあります。初期化処理は、『Form』の『Form\_Loadイベント』内で、初期化処理関数を実行します。『Form\_Loadイベント』は『Form』がロード (立ち上がる) 際に発生するイベントで、各コントロールの既定値の設定や、変数を初期化する際に使われます。



下記のコードを『Private Sub Form\_Load ()』から『End Sub』の間に記述してください。

```
DeviceName = "DIO00"
Ret = DioOpenEx (DeviceName, hDrv)
```

'デバイス名を変数に格納  
'初期化処理関数を実行



## 初期化処理関数(デバイス名) 『DioOpenEx』 リファレンス

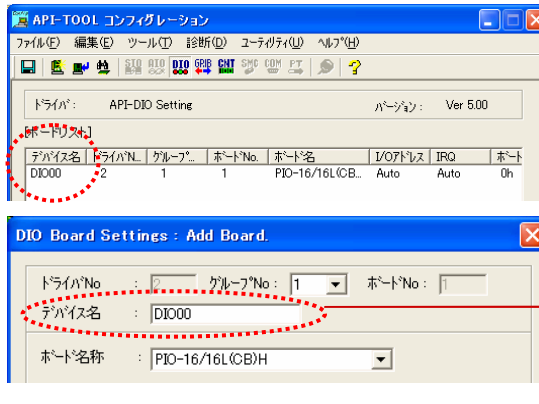
- 機能 使用するボード用のデバイスドライバをオープンし、ドライバの各関数を使用可能にします。
- 書式 Visual Basic6.0の場合

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

Ret = DioOpenEx (DeviceName,hDrv)
```

- 引数 hDrv : 使用するグループのデバイスハンドルが格納されます。  
他のDioXxxx関数を実行する時には、このデバイスハンドルを指定します。
- DeviceName : オープンするデバイス名を指定します。  
デバイス名は『API-TOOL Configuration』で設定します。

### デバイス名に関して



ドライバ番号、グループ番号で表される論理デバイスに対して付けられる名前をデバイス名と言います。デバイス名は『API-TOOL Configuration』で設定します。デバイス名は、[ボードの編集画面]で変更できますが、本書ではデフォルトの“DIO00”を使用します。

メニュー [編集] - [ボードの編集] の手順で、ボードの編集画面になります。この画面上で、デバイス名を任意に変更可能です。ただし、複数枚ボード/カードを使用する場合には重複しないように設定してください。

Ret: 終了情報(戻り値) → 正常終了:0、エラー終了: 0以外(詳細はヘルプの「戻り値一覧」参照)。

※ 本書では、戻り値の確認(エラー処理)は、誌面の都合上、割愛しております。  
実システムにおいては、関数を実行した後にエラー処理のコードを記述します。  
エラー処理の方法は、各サンプルプログラムを参照してください。

## 5-4-8.プログラム作成手順⑤ (終了処理の追加)

API-PAC (W32) は初期化処理ではじまり、終了処理で終了する決まりがあります。終了処理は、『Form』の『Form\_Unloadイベント』内で、終了処理関数を実行します。『Form\_Unloadイベント』は『Form』が画面から消去(アンロード)される際に発生するイベントです。

① コードウィンドウを開き、『オブジェクト』から『Form』を選択します。

② 『プロシージャ』から『Unload』を選択します。

③ Private Sub Form\_Unload () から、End Subの間に終了処理関数を記述します。

下記のコードを『Private Sub Form\_UnLoad ()』から『End Sub』の間に記述してください。

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

終了処理関数実行 (デバイスハンドル開放)

## 終了処理関数 『DioClose』 リファレンス

- 機能 指定したデバイスの終了処理を行います。
- 書式 Visual Basic6.0の場合

```
Dim hDrv As Long
Dim Ret As Long

Ret = DioClose (hDrv)
```

- 引数 hDrv: 終了するデバイスハンドルを指定します。このデバイスハンドルはDioOpenで取得したものです。
- Ret: 終了情報(戻り値) → 正常終了:0、エラー終了:0以外(詳細はヘルプの「戻り値一覧」参照)。

※ 本書では、戻り値の確認(エラー処理)は、誌面の都合上、割愛しております。実システムにおいては、関数を実行した後にエラー処理のコードを記述します。エラー処理の方法は、各サンプルプログラムを参照してください。

- 補足 終了処理実行後は、指定グループに対して、(DioOpen以外の)ドライバの各関数は実行できません。

## 5-4-9.プログラム作成手順⑥ (タイマ開始処理の記述)

100msec (0.1秒) 毎にデジタル入出力処理を行うために、【5-2.】にて使用したタイマコントロールを使用します。タイマコントロールは【5-2.】と同様に『開始ボタン』で開始、『停止ボタン』で停止の処理を行います。

先のプロパティ設定で『False (無効)』に設定したタイマコントロールを起動、すなわち開始させます。『開始ボタン』をクリックした時に『タイマコントロール (オブジェクト名: tmrTimer)』のEnabledプロパティを『False (無効)』から『True (有効)』に変える処理を記述します。フォーム上の『開始ボタン』をダブルクリックします。下記のようなコードウィンドウが開かれます。

① 『開始ボタン』をダブルクリックします。

② 『コードウィンドウ』が開きます。

③ Private Sub cmdStart\_Click () から、End Subまでの間に、何を行いたいかをコードで記述します。

```
Private Sub cmdStart_Click()
    tmrTimer.Enabled = True 'タイマを起動します
End Sub
```

Private Sub cmdStart\_Click () からEnd Subまでの間に、下記のコードを記述します。

`tmrTimer.Enabled = True`

『タイマを起動します』

## 5-4-10.プログラム作成手順⑦ (タイマ停止処理の記述)

『開始ボタン』で有効にしたタイマコントロールを停止させるための『停止ボタン (cmdStop)』の処理を記述していきます。この『停止ボタン』をクリックした時に『タイマコントロール (オブジェクト名: tmrTimer)』のEnabledプロパティを『True (有効)』から『False (無効)』に変える処理を記述します。フォーム上の『停止ボタン』をダブルクリックします。下記のようなコードウィンドウが開かれます。

① 『停止ボタン』をダブルクリックします。

② 『コードウィンドウ』が開きます。

③ Private Sub cmdStop\_Click () から、End Subまでの間に、何を行いたいかをコードで記述します。

```
Private Sub cmdStop_Click()
    tmrTimer.Enabled = False 'タイマを停止します
End Sub
```

Private Sub cmdStop\_Click () からEnd Subまでの間に、下記のコードを記述します

`tmrTimer.Enabled = False`

『タイマを停止します』

## 5-4-11.プログラム作成手順⑧ (デジタル入出力処理の追加)

デジタル入出力カード PIO-16/16L (CB) H (またはデモボード) の『入力0ビット』から、デジタルデータの入力を行う処理と、そのデータを『出力の1ビット』へ出力する処理を追加します。ビット単位の入力関数がドライバソフトウェアで提供されていますので、その関数を実行します。

① 『タイマコントロール』をダブルクリックします。

② 『コードウィンドウ』が開きます。

③ Private Sub tmrTimer\_Timer () から、End Subの間に下記のコードを記述します。

Private Sub tmrTimer\_Timer () から、End Subの間に記述します。

InBit = 0	'入力ビット指定用変数に"0"を設定
Ret = DioInpBit (hDrv, InBit, InBitData)	'入力0ビットのデータを変数に格納
OutBit = 1	'出力ビット指定用変数に"1"を設定
Ret = DioOutBit (hDrv, OutBit, InBitData)	'入力データを出力1ビットへ出力
lblData.Caption = Hex (InBitData)	'lblDataに入力データ (InBitData) を表示

## ビット単位のデジタル入力関数『DioInpBit』リファレンス

- 機能 指定した1ビットからデジタルデータの入力を行い、そのデータを変数に格納します。  
 ■書式 Visual Basic6.0の場合

Dim hDrv	As Long
Dim InBit	As Integer
Dim Buf	As Byte
Dim Ret	As Long

Ret = DioInpBit (hDrv,InBit,Buf)

- 引数 hDrv: デバイスハンドルを指定します。このデバイスハンドルはDioOpenで取得したものです。
- InBit: デジタル入力を行う論理ビット番号を指定します。  
 指定可能な範囲:  $0 \leq \text{InBit} \leq \text{論理ビットの最大値} - 1$   
 設定範囲外である場合、戻り値610hを返します。
- Buf: デジタル入力データを格納する変数を指定します。
- Ret: 終了情報 (戻り値) → 正常終了: 0、エラー終了: 0以外 (詳細はヘルプの「戻り値一覧」参照)。

※ 本書では、戻り値の確認 (エラー処理) は、誌面の都合上、割愛しております。  
 実システムにおいては、関数を実行した後にエラー処理のコードを記述します。  
 エラー処理の方法は、各サンプルプログラムを参照してください。

## ビット単位のデジタル出力関数『DioOutBit』リファレンス

- 機能 出力データが格納されている変数のデータをもとに、指定した1ビットへデジタルデータの出力を行います。  
 ■書式 Visual Basic6.0の場合

Dim hDrv	As Long
Dim OutBit	As Integer
Dim Buf	As Byte
Dim Ret	As Long

Ret = DioOutBit (hDrv,OutBit,Buf)

- 引数 hDrv: デバイスハンドルを指定します。このデバイスハンドルはDioOpenで取得したものです。
- OutBit: デジタル出力を行う論理ビット番号を指定します。  
 指定可能な範囲:  $0 \leq \text{OutBit} \leq \text{論理ビットの最大値} - 1$   
 設定範囲外である場合、戻り値610hを返します。
- Buf: デジタル出力データが格納されている変数を指定します。
- Ret: 終了情報 (戻り値) → 正常終了: 0、エラー終了: 0以外 (詳細はヘルプの「戻り値一覧」参照)。

※ 本書では、戻り値の確認 (エラー処理) は、誌面の都合上、割愛しております。  
 実システムにおいては、関数を実行した後にエラー処理のコードを記述します。  
 エラー処理の方法は、各サンプルプログラムを参照してください。

## 5-4-13.ビット単位のデジタル入出力プログラムリスト

Dim hDrv	As Long	'デバイスハンドル格納用変数
Dim Ret	As Long	'リターンコード(戻り値) 格納用変数
Dim InBit	As Integer	'入力ビット指定用
Dim InBitData	As Byte	'入力データ格納用
Dim OutBit	As Integer	'出力ビット指定用
Dim DeviceName	As String	'デバイス名格納用変数

Private Sub Form\_Load ()

DeviceName = "DIO00"	'デバイス名を変数に格納
Ret = DioOpenEx (DeviceName, hDrv)	'初期化処理関数を実行

End Sub

Private Sub Form\_Unload (Cancel As Integer)

Ret = DioClose (hDrv)	'終了処理関数実行(デバイスハンドル開放)
-----------------------	-----------------------

End Sub

Private Sub cmdStart\_Click ()

tmrTimer.Enabled = True	'タイマを起動します
-------------------------	------------

End Sub

Private Sub cmdStop\_Click ()

tmrTimer.Enabled = False	'タイマを停止します
--------------------------	------------

End Sub

Private Sub tmrTimer\_Timer ()

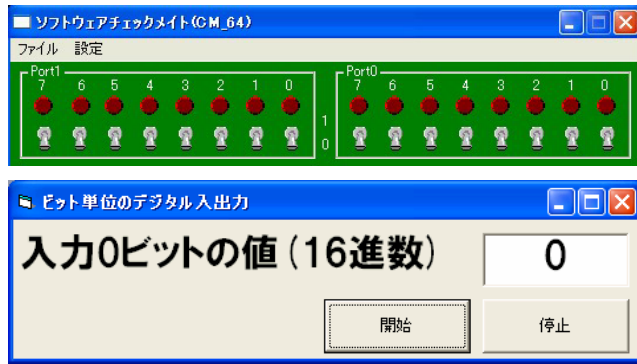
InBit = 0	'入力ビット指定用変数に"0"を設定
Ret = DioInpBit (hDrv, InBit, InBitData)	'入力0ビットのデータを変数に格納
OutBit = 1	'出力ビット指定用変数に"1"を設定
Ret = DioOutBit (hDrv, OutBit, InBitData)	'入力データを出力1ビットへ出力
lblData.Caption = Hex (InBitData)	'lblDataに入力データ (InBitData) を表示

End Sub



## 5-4-14.プログラムの実行

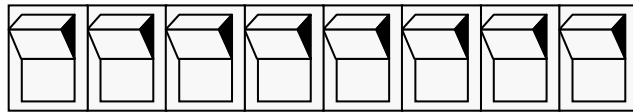
プログラムを保存した後、実行してみましょう。



『スイッチ 0』が『OFF』の場合は、  
『LED 1』は『消灯』状態です。

画面上のデータ表示は『0』です。

入力+0ポートの状態



D7 D6 D5 D4 D3 D2 D1 D0

出力+0ポートの状態

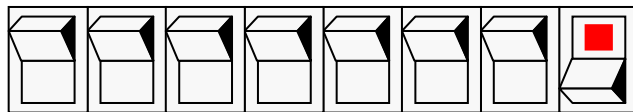
0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---



『スイッチ 0』が『ON』の場合は、  
『LED 1』は『点灯』状態です。

画面上のデータ表示は『1』です。

入力+0ポートの状態



D7 D6 D5 D4 D3 D2 D1 D0

出力+0ポートの状態

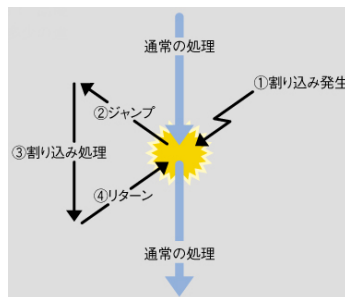
0	0	0	0	0	0	1	0
---	---	---	---	---	---	---	---

## 5-5. 割り込み処理 (割り込み入力プログラム)

これまでで、『タイマコントロールによるカウントアッププログラム』『ポート単位のデジタル入出力プログラム』『ビット単位のデジタル入出力プログラム』を作成しました。ここでは、『割り込み入力プログラム』を作成します。『割り込み処理』に関しては、第3章の用語解説で説明しましたが、もう一度おさらいしておきましょう。

### 5-5-1. 割り込み処理とは

特定の入力端子をパソコン (CPU) のIRQに接続して、外部から割り込みを発生させる機能です。外部装置の変化を検出して、特定の処理を実行するアプリケーションや、外部からの指令で高優先度の緊急処理などをする場合に使用します。今回使用しているPIO-16/16L (CB) HIは、入力の全点 (16点) が割り込みとして使用可能で、入力信号をまとめて、1つの割り込み信号を出力する仕様となっています。



### 5-5-2. Windowsにおける割り込み処理

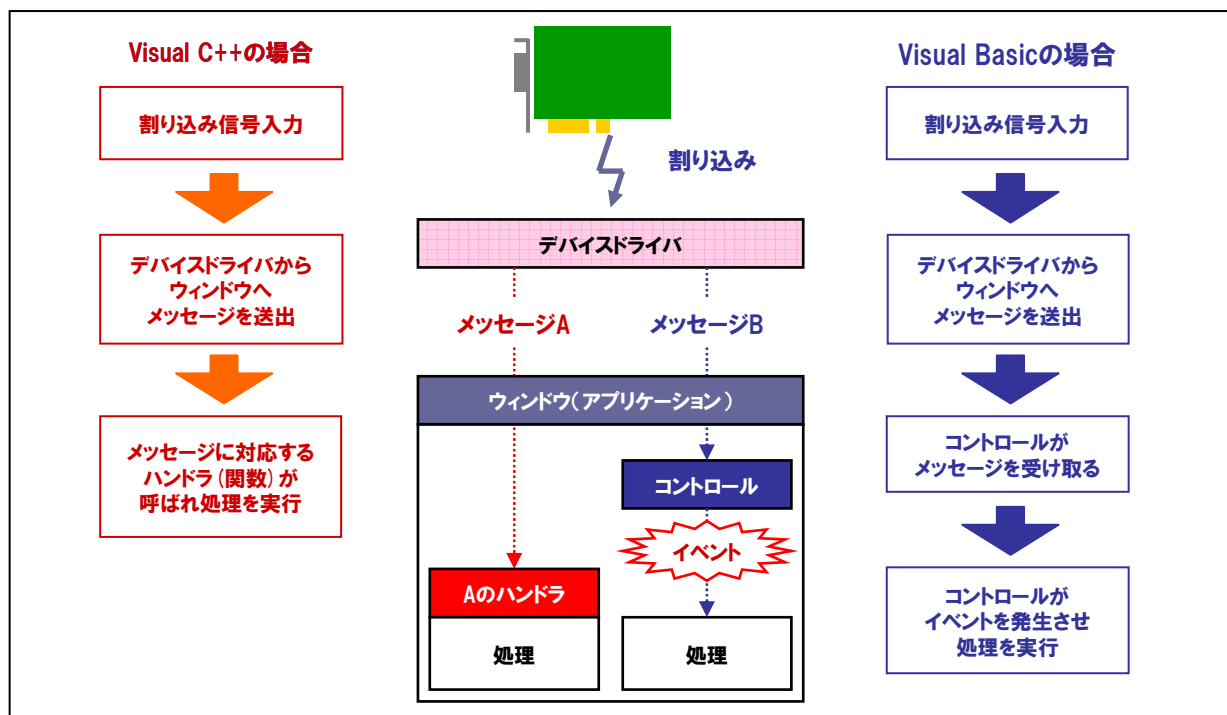
Windowsでは、外部機器から割り込み信号が入力されると、『デバイスドライバ』にその旨が通知されます。通知を受けた『デバイスドライバ』は『メッセージ』と呼ばれる32ビットの数値をアプリケーションに発信します。アプリケーションはそれらの送られてきた『メッセージ』に対応した各処理を行うことによって動作しています。

#### ①メッセージとは

接続された周辺機器からの入力などを知らせる32ビットの数値です。周辺機器から割り込み信号が入力された場合、デバイスドライバがその旨をアプリケーションへ通知するために使用します。標準的な装置はWindowsの仕様で、使用するメッセージ番号が決まっています (0~400 (16進数))。例えばマウスクリックを通知する番号は202 (16進数) です。新たに追加するデバイスで『割り込み入力』を使用する場合は、400 (16進数) 以降のメッセージ番号を重複しないように割り当てます。

#### ②Visual Basicにおける割り込み入力処理

Visual Basicは、Visual C++と異なり、『メッセージ』を直接受け取ることができません。そのため弊社では、Visual Basic上で割り込み処理を行う (デバイスドライバからのメッセージを受け取る) ために、『Msgecho.ocx』というコントロールを提供しています (コントロール (.OCX) は、メッセージを受け取ることができます)。この『Msgecho.ocx』はドライバソフトウェアと同時にインストールされています。



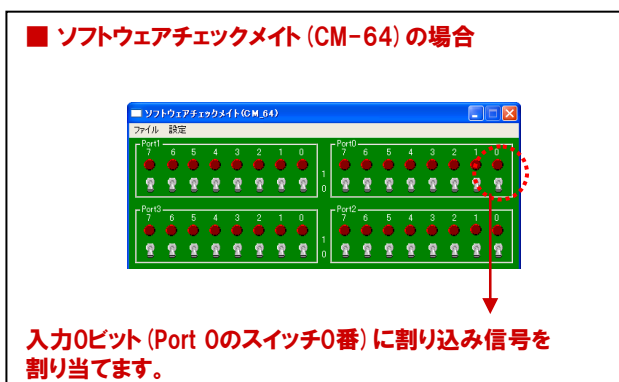
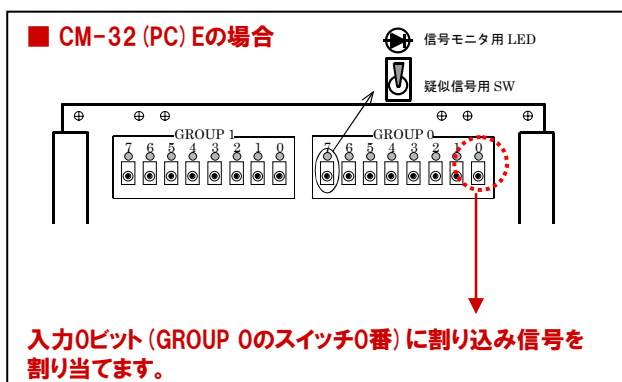
それでは、一番初めに作成したタイマコントロールによるカウントアッププログラムを利用して、割り込み入力プログラムを作成していきましょう。

### 5-5-3.割り込み入力プログラムの概要

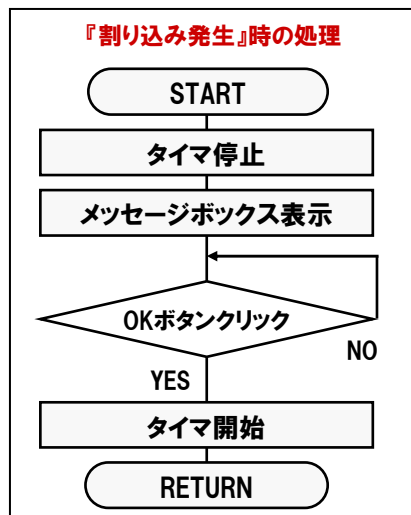
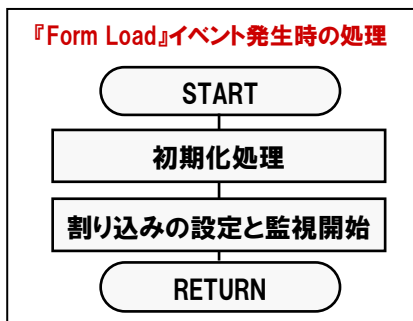
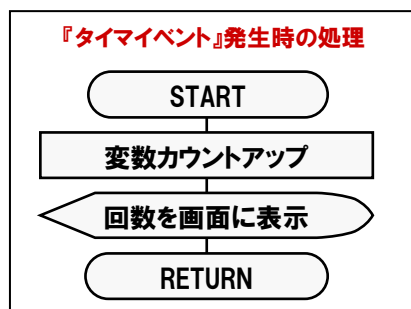
開始ボタンをクリックすると、100msec (0.1秒) 毎にタイマイベントが発生、その回数を画面上に表示します。タイマイベントが発生している最中に『スイッチ 0』に割り当てられた割り込み信号を入力 (スイッチをONする) すると『割り込みが発生しました!』のメッセージボックスが表示されます。メッセージボックス上の『OK』ボタンをクリックするとカウントアップが再開されるプログラムを作成します。



### 5-5-4.実行環境



### 5-5-5.プログラム フローチャート



## 5-5-6.プログラム作成手順① (画面の作成とプロパティ設定)

[5-2-2.]~[5-2-7.]を参考にして、オブジェクトの配置と各オブジェクトのプロパティ設定を行ってください。

**■ Form**  
 ◎Caption = 『割り込み入力プログラム』

**■ Labelコントロール**  
 ◎オブジェクト名 = 『lblData』

**■ 開始ボタン**  
 ◎Caption = 『開始』  
 ◎オブジェクト名 = 『cmdStart』

**■ 停止ボタン**  
 ◎Caption = 『停止』  
 ◎オブジェクト名 = 『cmdStop』

**■ タイマコントロール**  
 ◎オブジェクト名 = 『tmrTimer』  
 ◎Enabled = 『False』  
 ◎Interval = 『100』

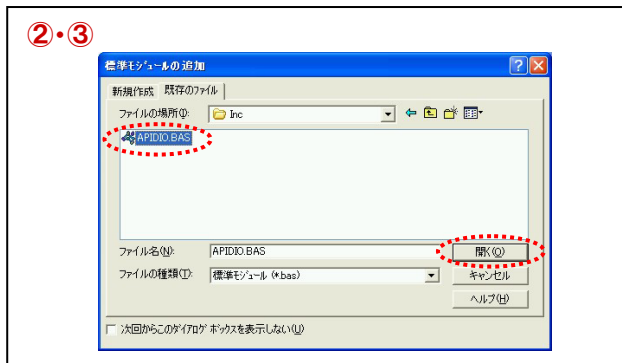
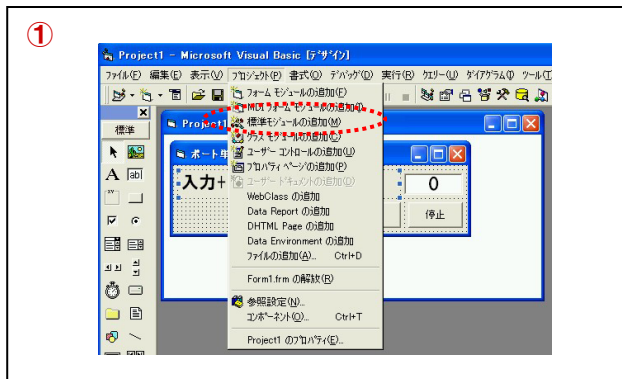
## 5-5-7.プログラム作成手順② (標準モジュールファイルの追加)

Visual BasicでAPI-PAC (W32) が提供する各関数を実行するには、使用する関数が参照するDLLの情報などが記述されている『標準モジュールファイル』をプロジェクトに追加する必要があります。

- ① Visual Basicのメニュー『プロジェクト』 - 『標準モジュールの追加』を選択します。
- ② ダイアログが表示されるので、『既存のファイル』のタブをクリックします。
- ③ 標準設定でインストールした場合には下記の場所に標準モジュールファイル (APIDIO.BAS) がありますので、選択して『開く』ボタンをクリックします。

**C:\¥Program Files¥CONTEC¥API-PAC (W32) ¥Dio ¥Samples ¥Inc ¥APIDIO.BAS**

- ④ プロジェクトエクスプローラ上に標準モジュールファイルが追加されていることを確認してください。



④

APIDIO.BASをダブルクリックで開くと、ドライバソフトウェアが提供している関数の情報(参照先)や、構造体の宣言などがされているのが分かります。

### 5-5-8.プログラム作成手順③ (変数の追加)

変数宣言部分に下記の変数を追加します。変数名は任意ですが、変数の型 (整数型やバイト型) は、プログラム中で使用するドライバソフトウェアの関数の仕様に合わせて決定します。関数リファレンスは、インストールしたオンラインヘルプファイルに記載されていますので参照してください。

『スタートメニュー』 - 『プログラム』 - 『CONTEC API-PAC (W32)』 - 『DIO』 - 『API-DIO HELP』

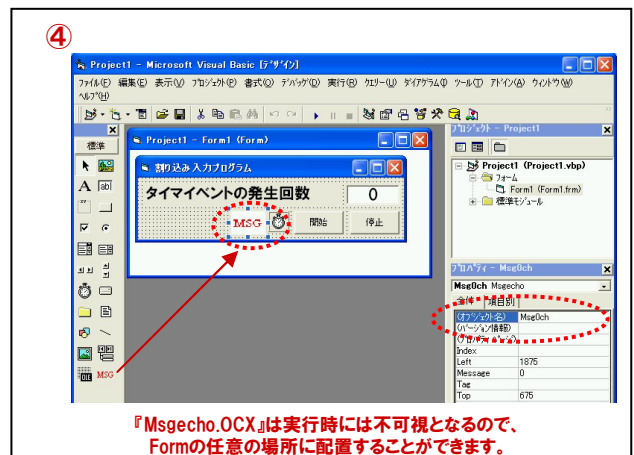
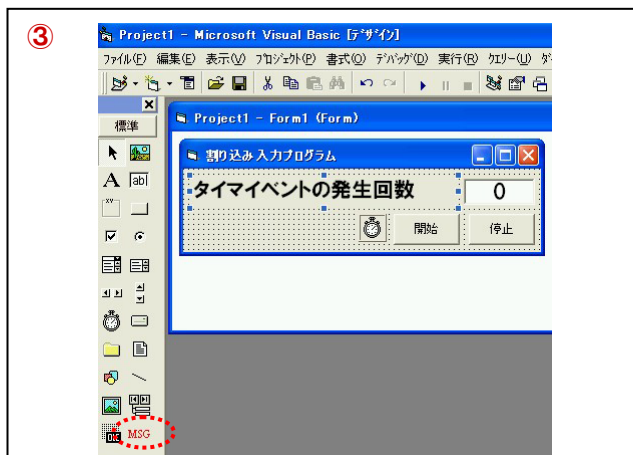
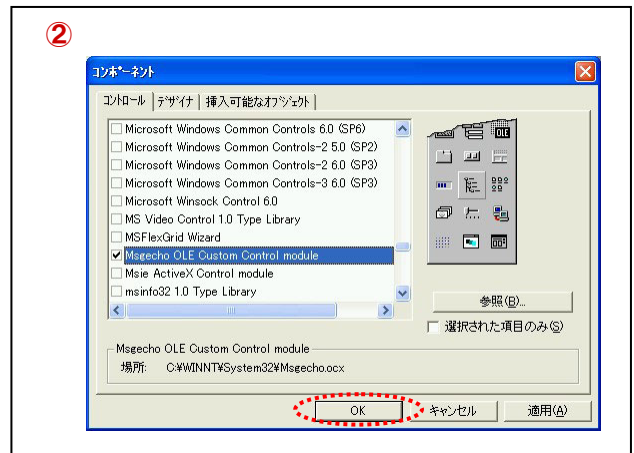
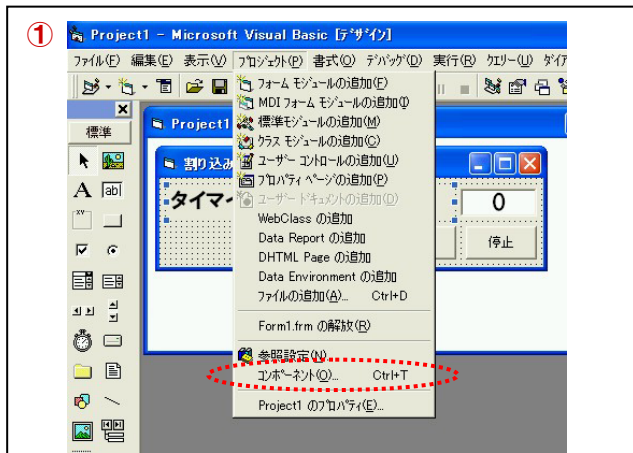
- ① 『コードの表示』ボタンをクリックして、コードウィンドウを開きます。
- ② 空欄のコード記述欄に下記の変数を記述します。

Dim hDrv	As Long	'デバイスハンドル格納用
Dim Ret	As Long	'リターンコード (戻り値) 格納用
Dim DeviceName	As String	'デバイス名格納用変数
Dim SigLog	As Integer	'フラグ (割り込みのエッジ) 指定用
Dim SigCh	As Integer	'割り込み入力端子設定用
Dim Counter	As Long	'カウントアップ用変数

### 5-5-9.プログラム作成手順④ (Msgecho.ocxコントロールの追加)

割り込みの『メッセージ』を受け取るためのコントロール『Msgecho.OCX』をプロジェクトに追加します。

- ① Visual Basicのメニュー『プロジェクト』 - 『コンポーネント』を選択します。
- ② 一覧から『Msgecho OLE Custom Control Module』を選択 (チェックを付ける) して、『OK』をクリックします。
- ③ 『コントロール・ツール・ボックス』に『Msgecho.OCX』が追加されます。
- ④ 『Msgecho.OCX』を選択して、Formに貼り付け、オブジェクト名を『Msg0ch』に変更します。



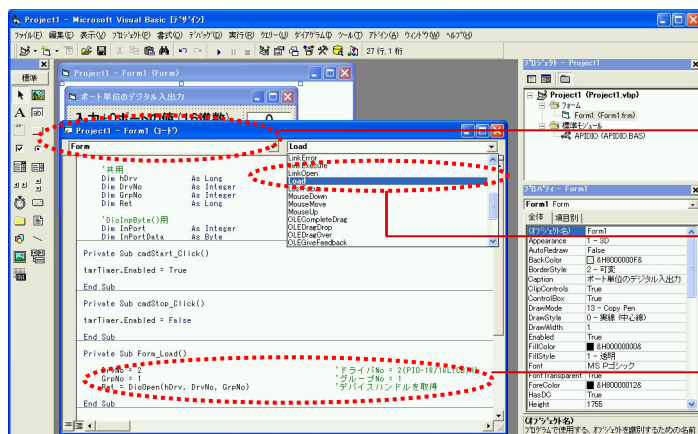
『Msgecho.OCX』は実行時には不可視となるので、Formの任意の場所に配置することができます。



## 5-5-10.プログラム作成手順⑤ (初期化処理の追加)

API-PAC (W32) は初期化処理ではじまり、終了処理で終了する決まりがあります。初期化処理は、『Form』の『Form\_Loadイベント』内で、初期化処理関数を実行します。

下記のコードを『Private Sub Form\_Load ()』から『End Sub』の間に記述してください。



① コードウィンドウを開き、『オブジェクト』から『Form』を選択します。

② 『プロシージャ』から『Load』を選択します。

③ Private Sub Form\_Load () から、End Subの間に初期化処理を記述します。

Private Sub Form\_Load () から、End Subの間に、下記のコードを記述します。

```
DeviceName = "DIO00"
Ret = DioOpenEx (DeviceName, hDrv)
```

'デバイス名を変数に格納  
'初期化処理関数を実行

## 初期化処理関数 (デバイス名) 『DioOpenEx』 リファレンス

- 機能 使用するボード用のデバイスドライバをオープンし、ドライバの各関数を使用可能にします。
- 書式 Visual Basic6.0の場合

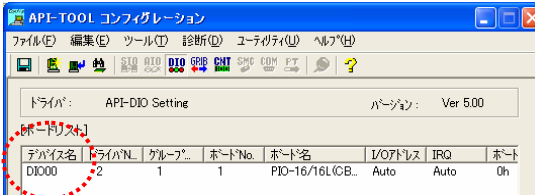
```
Dim hDrv As Long
Dim DeviceName As String
Dim Ret As Long

Ret = DioOpenEx (DeviceName,hDrv)
```

- 引数 hDrv : 使用するグループのデバイスハンドルが格納されます。他のDioXxxx関数を実行する時には、このデバイスハンドルを指定します。

DeviceName : オープンするデバイス名を指定します。  
デバイス名は『API-TOOL Configuration』で設定します。

### デバイス名に関して



ドライバ番号、グループ番号で表される論理デバイスに対して付けられる名前をデバイス名と言います。デバイス名は『API-TOOL Configuration』で設定します。デバイス名は、[ボードの編集画面]で変更できますが、本書ではデフォルトの『DIO00』を使用します。

メニュー [編集] - [ボードの編集] の手順で、ボードの編集画面になります。この画面上で、デバイス名を任意に変更可能です。ただし、複数枚ボード/カードを使用する場合には重複しないように設定してください。

Ret: 終了情報 (戻り値) → 正常終了:0、エラー終了: 0以外 (詳細はヘルプの「戻り値一覧」参照)。

※ 本書では、戻り値の確認 (エラー処理) は、誌面の都合上、割愛しております。実システムにおいては、関数を実行した後にエラー処理のコードを記述します。エラー処理の方法は、各サンプルプログラムを参照してください。



## 5-5-11.プログラム作成手順⑥（終了処理の追加）

API-PAC (W32) は初期化処理ではじまり、終了処理で終了する決まりがあります。終了処理は、『Form』の『Form\_Unloadイベント』内で、終了処理関数を実行します。『Form\_Unloadイベント』は『Form』が画面から消去（アンロード）される際に発生するイベントです。

① コードウィンドウを開き、『オブジェクト』から『Form』を選択します。

② 『プロシージャ』から『Unload』を選択します。

③ Private Sub Form\_Unload () から、End Subの間に終了処理関数を記述します。

下記のコードを『Private Sub Form\_UnLoad () 』から『End Sub』の間に記述してください。

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

終了処理関数実行（デバイスハンドル開放）

## 終了処理関数 『DioClose』 リファレンス

- 機能 指定したデバイスの終了処理を行います。
- 書式 Visual Basic6.0の場合

```
Dim hDrv As Long
Dim Ret As Long
Ret = DioClose (hDrv)
```

- 引数 hDrv: 終了するデバイスハンドルを指定します。このデバイスハンドルはDioOpenで取得したものです。
- Ret: 終了情報（戻り値） → 正常終了:0、エラー終了:0以外（詳細はヘルプの「戻り値一覧」参照）。

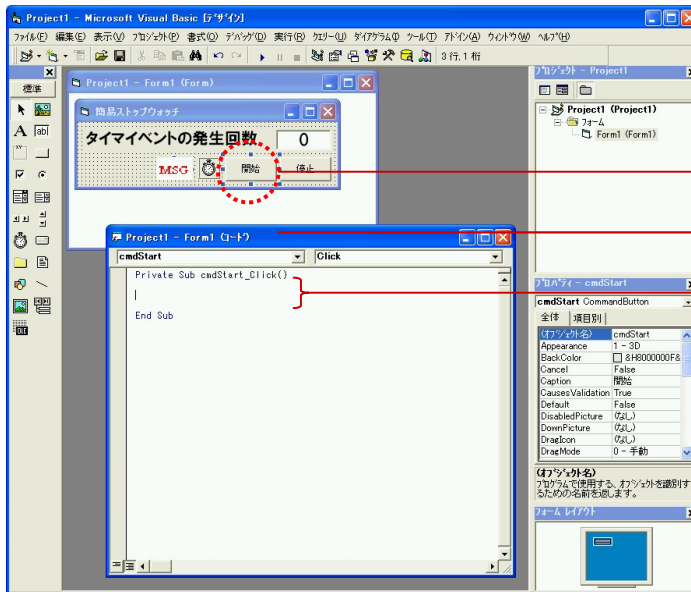
※ 本書では、戻り値の確認（エラー処理）は、誌面の都合上、割愛しております。実システムにおいては、関数を実行した後にエラー処理のコードを記述します。エラー処理の方法は、各サンプルプログラムを参照してください。

- 補足 終了処理実行後は、指定グループに対して、(DioOpen以外の)ドライバの各関数は実行できません。

## 5-5-12.プログラム作成手順⑦ (タイマ開始コードの記述)

ここからは、変数のカウントアップの機能を実現するためのコード(プログラム)を記述していきます。先に述べましたが、Visual Basicは『イベント駆動型(イベントドリブン)言語』です。『何を(か)』、『何した時に』、『何をするか』をコードで表現します。まず最初に行いたい処理は、先のプロパティ設定で『False(無効)』に設定したタイマコントロールを起動、すなわち開始させることです。そのために用意したのが『開始ボタン(オブジェクト名:cmdStart)』です。

まず最初に行いたい処理は、先のプロパティ設定で『False(無効)』に設定したタイマコントロールを起動、すなわち開始させることです。そのために用意したのが『開始ボタン(オブジェクト名:cmdStart)』です。この『開始ボタン』をクリックした時に『タイマコントロール(オブジェクト名:tmrTimer)』のEnabledプロパティを『False(無効)』から『True(有効)』に変える処理を記述します。フォーム上の『開始ボタン』をダブルクリックします。下記のようなコードウィンドウが開かれます。



- ① 『開始ボタン』をダブルクリックします。
- ② 『コードウィンドウ』が開きます。
- ③ Private Sub cmdStart\_Click() から、End Subまでの間に、何を行いたいかをコードで記述します。

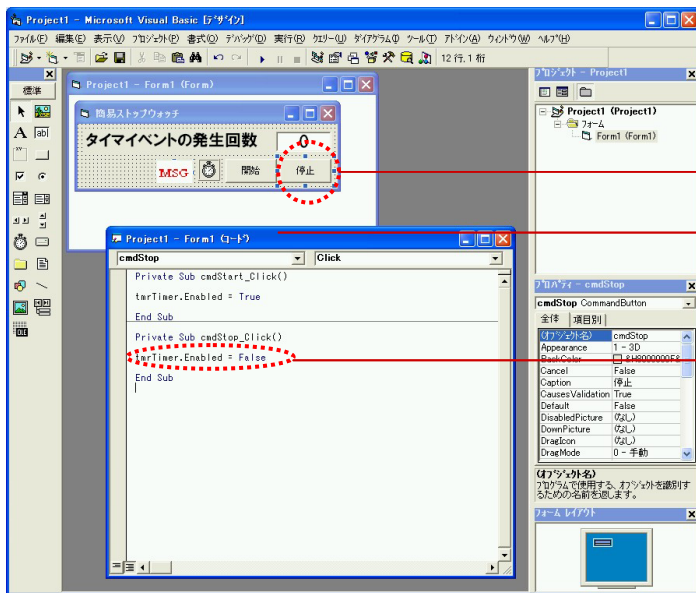
Private Sub cmdStart\_Click() からEnd Subまでの間に、下記のコードを記述します。『』から始まる部分は、コメント行として扱われますので、他の人がコードを見ても処理が分かりやすいようにコメントを書いておくのが望ましいです。文法は『何の』、『何を』、『何したいか』の順番で記述します。今回は、『タイマコントロール(tmrTimer)』の『Enabledプロパティ』を『True(有効)』にするという意味になります。

```
tmrTimer.Enabled = True
```

『タイマを起動します』

## 5-5-13.プログラム作成手順⑧ (タイマ停止コードの記述)

次は、『開始ボタン』で有効にしたタイマコントロールを停止させるための『停止ボタン(cmdStop)』の処理を記述していきます。この『開始ボタン』をクリックした時に『タイマコントロール(オブジェクト名:tmrTimer)』のEnabledプロパティを『True(有効)』から『False(無効)』に変える処理を記述します。フォーム上の『停止ボタン』をダブルクリックします。



① 『停止ボタン』をダブルクリックします。

② 『コードウィンドウ』が開きます。

③ Private Sub cmdStop\_Click () から、End Subまでの間に、何を行いたいかをコードで記述します。

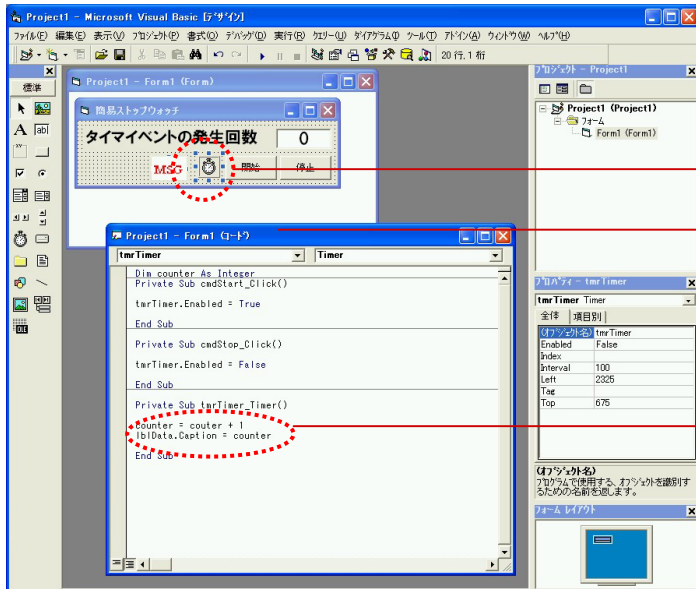
Private Sub cmdStop\_Click () からEnd Subまでの間に下記のコードを記述します。

`tmrTimer.Enabled = False`

'タイマを停止します

#### 5-5-14.プログラム作成手順⑨ (カウントアップ処理コードの記述)

タイマイベントが発生した際の処理を記述します。このタイマイベントは、タイマコントロールのEnabledプロパティが『True (有効)』の間、プロパティ『Interval』で設定した間隔毎に発生します。タイマコントロール (tmrTimer) をダブルクリックして、コードウィンドウを開きます。



① 『タイマコントロール』をダブルクリックします。

② 『コードウィンドウ』が開きます。

③ Private Sub tmrTimer\_Timer () から、End Subまでの間に、繰り返し何を行いたいかをコードで記述します。

Private Sub tmrTimer\_Timer () からEnd Subまでの間に、下記のコードを記述します。変数『Counter』のカウントアップ (+1の繰り返し) と『Counter』の現在値をLabelコントロール『lblData』に繰り返し表示するという処理です。

`Counter = Counter + 1`  
`lblData.Caption = Counter`

'変数に+1を行います  
'変数を画面表示します

現在までのコードは下記のとおりになります。このコードに割り込み処理を追加していくことになります。

Dim hDrv	As Long	'デバイスハンドル格納用
Dim Ret	As Long	'リターンコード(戻り値)格納用
Dim DeviceName	As String	'デバイス名格納用変数
Dim SigLog	As Integer	'フラグ(割り込みのエッジ)指定用
Dim SigCh	As Integer	'割り込み入力端子設定用
Dim Counter	As Long	'カウントアップ用変数

Private Sub Form\_Load ()

DeviceName = "DIO00"	'デバイス名を変数に格納
Ret = DioOpenEx (DeviceName, hDrv)	'初期化処理関数を実行

End Sub

Private Sub Form\_Unload (Cancel As Integer)

Ret = DioClose (hDrv)	'終了処理関数実行(デバイスハンドル開放)
-----------------------	-----------------------

End Sub

Private Sub cmdStart\_Click ()

tmrTimer.Enabled = True	'タイマを起動します
-------------------------	------------

End Sub

Private Sub cmdStop\_Click ()

tmrTimer.Enabled = False	'タイマを停止します
--------------------------	------------

End Sub

Private Sub tmrTimer\_Timer ()

Counter = Counter + 1	'変数に+1を行います
lblData.Caption = Counter	'変数を画面表示します

End Sub

## 5-5-15.プログラム作成手順⑩ (割り込み監視開始処理の追加)

割り込みイベントを有効にする関数を使用します。この関数を実行すると、ただちに割り込み信号が有効になります。今回は、タイマの起動と同時に(開始ボタンのクリック)に実行するようにします。開始ボタンクリック時処理の場所に下記4行を追加します。

```
Private Sub cmdStart_Click ()
```

```
tmrTimer.Enabled = True
```

'タイマを起動します

```
MsgOch.Message = &H470
```

'メッセージ番号の指定

```
SigLog = 0
```

'立ち上がり(0)を指定

```
SigCh = 0
```

'割り込み使用ビット番号を指定

```
Ret = DioEventEx (hDrv, SigLog, SigCh, MsgOch.hWnd, MsgOch.Message)
```

'監視開始

→ この4行を追加します。

```
End Sub
```

## 割り込み処理関数『DioEventEx』リファレンス

■機能 指定したデバイスの割り込みを利用して、バックグラウンドからアプリケーション側に割り込みイベントメッセージ信号を発生させます。割り込みとして使用可能な各ビット毎に別のメッセージを送出できます。

■書式 Visual Basic6.0の場合

```
Dim hDrv As Long
Dim SigLog As Integer
Dim SigCh As Integer
Dim Ret As Long
```

```
MsgOK.Message = &H470  メッセージ番号を指定
SigLog = 0             0 (信号の立ち上がり) を指定
SigCh = 0             入力端子を指定
```

```
Ret = DioEventEx(hDrv, SigLog, SigCh, MsgOK.hWnd, MsgOK.Message)
```

■引数 hDrv: 終了するデバイスハンドルを指定します。このデバイスハンドルはDioOpenで取得したものです。

SigLog: 割り込み入力信号の論理フラグを指定します。  
0: 入力信号値の0から1 (立ち上がり)  
1: 入力信号値の1から0 (立ち下がり)

SigCh: 割り込み信号端子の入力を行うビット番号を指定します。

hWnd: 割り込みイベントメッセージを受け取る側のウィンドウのハンドルを指定します。  
この引数は、ユーザー側で値を設定する必要はありません。

Message: 割り込みイベントメッセージで使用する識別メッセージ番号を指定します。メッセージを使用しない場合は0を設定します。先に説明したとおり、メッセージ番号の0~400 (16進数) は、Windowsにより使用されていますので、400 (16進数) 以降を設定します。弊社のサンプルプログラムでは、このメッセージ番号が重複しないよう、各デバイスドライバ毎に推奨番号を決めています。

```
SIO (シリアル通信) → 410H~42FH
GPIB (GPIB通信) → 430H~44FH
AIO (アナログ入出力) → 450H~46FH
DIO (デジタル入出力) → 470H~48FH
SMC (モータコントロール) → 490H~4AFH
CNT (カウンタ入力) → 4B0H~4CFH
```

今回はデジタル入出力ドライバを使用しているため、470 (16進数) を設定しています。

Ret: 終了情報 (戻り値) → 正常終了:0、エラー終了: 0以外 (詳細はヘルプの「戻り値一覧」参照)。



## 5-5-16.プログラム作成手順①(割り込み発生時処理の追加)

『DioEventEx』を実行すると、指定したビット番号の割り込み信号が有効となり、指定したビット番号から割り込み信号が入力されると、デバイスドライバは『DioEventEx』使用時に設定した470(16進数)番の『メッセージ』をアプリケーション(今回はVisual Basic)に送出します。この『メッセージ』を受け取って、イベント発生させるのが『Msgecho.OCX』です。この『Msgecho.OCX』は、指定した番号の『メッセージ』を受け取ると、『MessageEcho』というイベントを発生させます。このイベントプロシージャ内に割り込み入力時『何が(を)』したいのかを記述します。今回は、”割り込みが発生しました!”という文字をVisual Basicのメッセージボックス機能を使用して表示します。

① 『Msgecho.OCX』をダブルクリックします。

② 『コードウィンドウ』が開きます。

③ Private Sub MsgOch\_MessageEcho () から、End Subまでの間に、何を行いたいかをコードで記述します。

下記のコードを『Private Sub MsgOch\_MessageEcho ()』から『End Sub』までの間に記述します。

```

tmrTimer.Enabled = False          'タイマを停止します
Ret = MsgBox("割り込みが発生しました!") 'メッセージダイアログボックスを表示
If Ret = 1 Then                    'OKボタンが押されたら、
    tmrTimer.Enabled = True        'タイマ処理を再開します
End If
    
```

### TOPICS

#### 『DioEventとDioEventExの使い分け』

今回使用しているデジタル入出力カードドライバソフトウェアでは、2種類の割り込み処理関数が提供されています。両者は、どちらも割り込みを扱う関数ですが、外部仕様および内部的な動作が異なります。用途・環境により使い分けをします。

	DioEvent	DioEventEx
呼び出し単位	全ビットで1回	1ビット毎に1回
割り込みビット判別	DioIntSence関数	メッセージ番号で識別
信号論理	全ビット同一	1ビット毎に設定

#### ■DioEvent関数(速度優先)

DioEvent関数は、スレッドを作成し、内部でループしており、システムオブジェクトの状態変化を待つことにより、割り込みが入ったことを識別します。これにより、DioEventExより高速な応答が可能です。ただし、システム負荷が大きくなった場合に、ループ処理の間に入った割り込みを取りこぼす可能性があります。

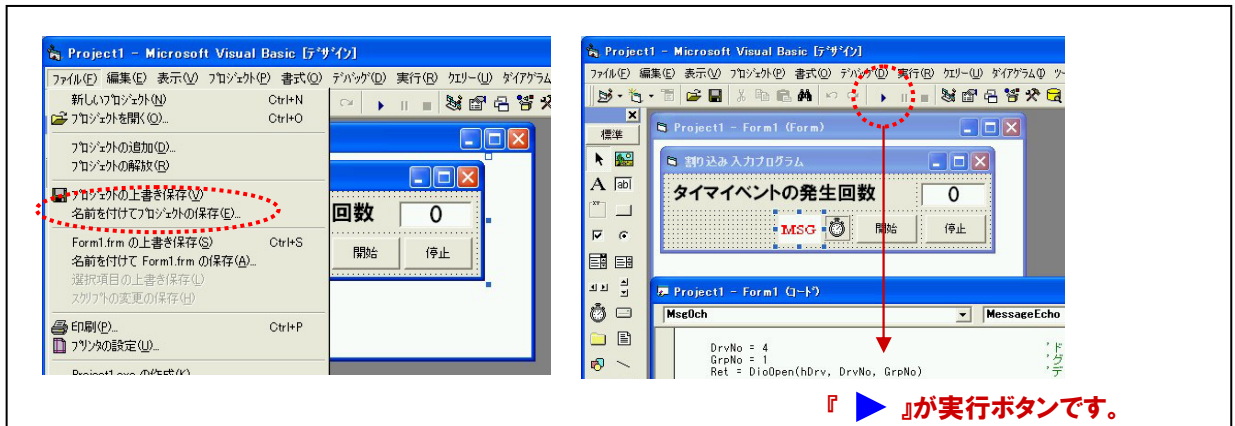
#### ■DioEventEx関数(回数優先)

DioEventEx関数は、スレッドを作成し、内部でループしており、割り込み処理ルーチンが走ったときに格納されるドライバワークを10ms間隔でポーリングしています。これにより、割り込み処理ルーチンが走った回数だけのメッセージを上げることが可能です。ただし、ポーリング周期が10ms単位のため、DioEvent関数よりも応答性が劣ります。



## 5-5-17.プログラムの実行

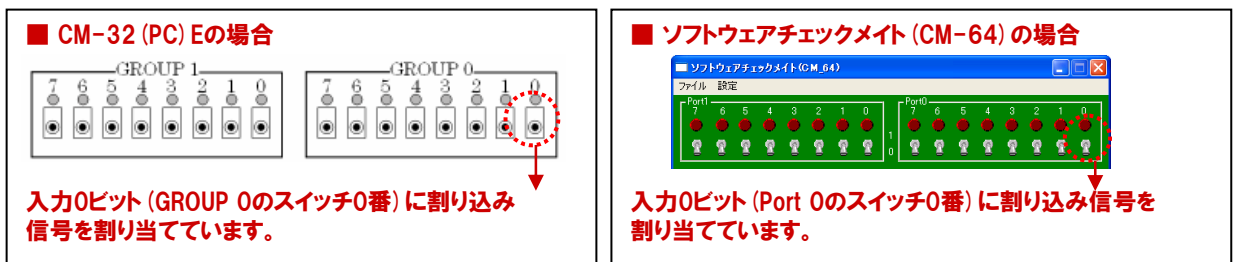
- ① 『ファイル』メニューの中から『名前を付けてプロジェクトの保存』を選択して、任意の場所に、任意のプロジェクト名称にて今回作成したプロジェクトを保存します。保存が完了したら、ツールバーの実行ボタンをクリックし、画面上の『開始ボタン』をクリックして実行してみましょう。



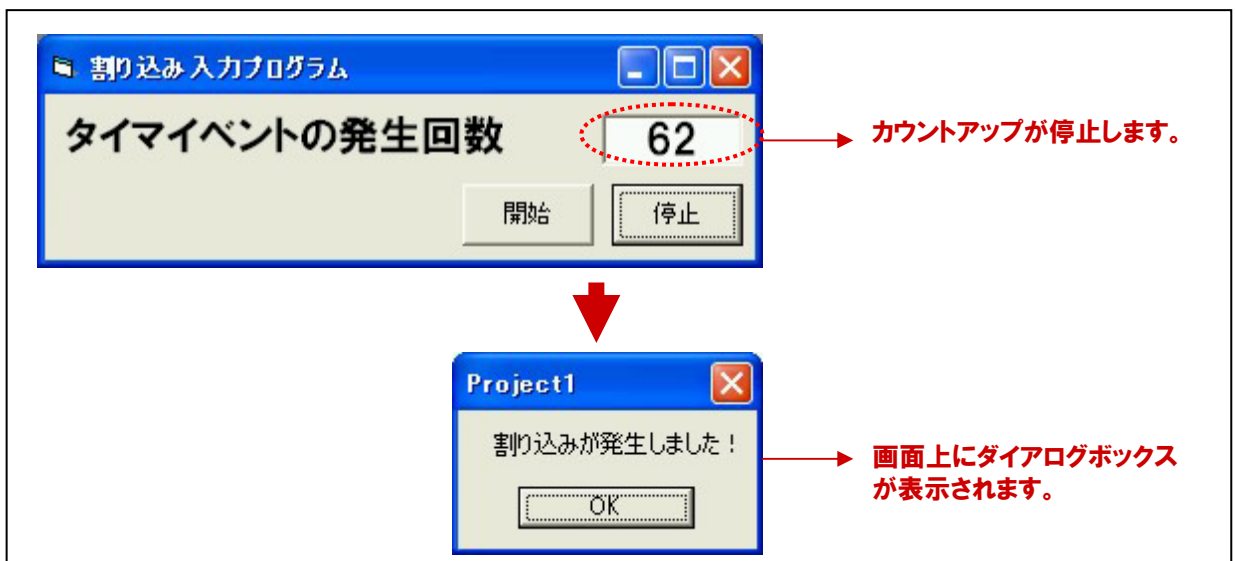
- ② プログラム実行後に『開始ボタン』をクリックするとタイマが起動し、カウントアップが始まります。同時に割り込み信号の監視状態に入ります。



- ③ カウントアップがされている状態で、『GROUP 0 (Port0)』の『スイッチ 0』を『ON』にしてみてください。



- ④ カウントアップが停止して、『タイマイベントが発生しました！』のメッセージボックスが表示されます。



- ⑤ カウントアップが停止して、“タイマイイベントが発生しました！”のメッセージボックスが表示されました。ダイアログボックスの『OKボタン』をクリックすると、ダイアログ表示処理が終わり、再びカウントアップが再開します。このように、割り込み機能を使用すれば、フォアグラウンドでプログラムを実行しながら、バックグラウンドで割り込み（スイッチ）の状態を監視し、優先的に割り込み発生時の処理を実行することができます。

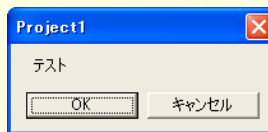


#### TOPICS『MsgBox関数について』

ダイアログボックスにメッセージを表示し、ボタンがクリックされるのを待って、どのボタンがクリックされたかを示す値を返すVisual Basicの標準関数です。関数の引数として、表示するメッセージや、使用するボタンの種類を指定して使用します。詳細は、Visual Basicの説明書やオンラインヘルプを参照願います。

例) ダイアログに『テスト』というメッセージを表示させ、『OKボタン』と『キャンセルボタン』を表示させる。

`Ret = MsgBox ("テスト", vbOKCancel)`



ここで、第2引数として指定しているボタンの種類は主なものとして以下のようなものがあります。

vbOKOnly	[OK] ボタンのみを表示します (省略した場合のデフォルト値です: 本書にて使用)。
vbOKCancel	[OK] ボタンと [キャンセル] ボタンを表示します。
vbAbortRetryIgnore	[中止]、[再試行]、および [無視] の 3 つのボタンを表示します。
vbYesNoCancel	[[はい]、[いいえ]、および [キャンセル] の 3 つのボタンを表示します。
vbYesNo	[[はい] ボタンと [いいえ] ボタンを表示します。

MsgBox関数が実行されて、表示されたボタンをクリックするとダイアログボックスは閉じられて、MsgBox関数は戻り値を返してくれます。通常は、この戻り値を判断し、『OKボタン』がクリックされたら何をする、『キャンセルボタン』がクリックされたら何をするというプログラムを組んでいきます。本書では、この戻り値=1 (vbOK) により、カウントアップを再開させる処理を行っています。その他の主な戻り値は、以下のとおりです。

vbOK	[OK] がクリックされた場合	1
vbCancel	[キャンセル] がクリックされた場合	2
vbAbort	[中止] がクリックされた場合	3
vbRetry	[再試行] がクリックされた場合	4
vbIgnore	[無視] がクリックされた場合	5

## 5-5-18.割り込み入力プログラムリスト

Dim hDrv	As Long	'デバイスハンドル格納用
Dim Ret	As Long	'リターンコード(戻り値)格納用
Dim DeviceName	As String	'デバイス名格納用変数
Dim SigLog	As Integer	'フラグ(割り込みのエッジ)指定用
Dim SigCh	As Integer	'割り込み入力端子設定用
Dim Counter	As Long	'カウントアップ用変数

Private Sub Form\_Load ()

DeviceName = "DIO00"	'デバイス名を変数に格納
Ret = DioOpenEx (DeviceName, hDrv)	'初期化処理関数を実行

End Sub

Private Sub Form\_Unload (Cancel As Integer)

Ret = DioClose (hDrv)	'終了処理関数実行(デバイスハンドル開放)
-----------------------	-----------------------

End Sub

Private Sub cmdStart\_Click ()

tmrTimer.Enabled = True	'タイマを起動します
MsgOch.Message = &H470	'メッセージ番号の指定
SigLog = 0	'立ち上がり(0)を指定
SigCh = 0	'割り込み入力端子を指定
Ret = DioEventEx (hDrv, SigLog, SigCh, MsgOch.hWnd, MsgOch.Message)	'監視開始

End Sub

Private Sub cmdStop\_Click ()

tmrTimer.Enabled = False	'タイマを停止します
--------------------------	------------

End Sub

Private Sub tmrTimer\_Timer ()

Counter = Counter + 1	'変数に+1を行います
lblData.Caption = Counter	'変数を画面表示します

End Sub

Private Sub MsgOch\_MessageEcho ()

tmrTimer.Enabled = False	'タイマを停止します
Ret = MsgBox ("割り込みが発生しました!")	'メッセージダイアログボックスを表示
If Ret = 1 Then	'OKボタンが押されたらタイマを開始
tmrTimer.Enabled = True	'して割り込み処理を抜けます
End If	

End Sub

## 5-6.ドライバソフトウェア提供関数一覧

デジタル入出力ドライバ『API-DIO』は、弊社製デジタル入出力ボード/カードを簡単に制御できる、さまざまな関数を用意しています。それぞれの関数は、処理ごとに分かりやすく分類されており、また処理の内容が一目で分かるような名称となっています。これらの関数を使用すれば、さらに高度な処理も可能です。

### ①共通関数 (初期化・終了処理、デジタルフィルタなど)

DioOpen	初期化処理
DioOpenEx	初期化処理 (デバイス名)
DioClose	終了処理
DioWait	指定時間の待ち時間処理
DioWaitEx	指定時間の待ち時間処理 (ms単位)
DioSFilter	指定グループのフィルタ処理
DioUseMutex	マルチプロセス時の排他処理

### ②8255搭載ボード/カード (双方向入出力) 専用関数

Dio8255Mode	8255モード設定
DioGet8255Mode	8255モード設定取得

### ③PIO-32D (PM) 専用関数

DioSetIoDirection	PIO-32D (PM) (Dual) の入出力設定
DioGetIoDirection	PIO-32D (PM) (Dual) の入出力設定取得

### ④簡易入出力関数

DioInpByte	指定ポートのデジタル入力処理
DioOutByte	指定ポートのデジタル出力処理
DioEchoBackByte	指定ポートのリードバックデータ入力処理
DioInpBit	指定ビットのデジタル入力処理
DioOutBit	指定ビットのデジタル出力処理
DioEchoBackBit	指定ビットのリードバックデータ入力処理

### ⑤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)
DioOut	指定ポートのデジタル出力処理 (複数バイト)
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)
DioBitInp	指定ビットのデジタル入力処理 (複数ビット)
DioBitOut	指定ビットのデジタル出力処理 (複数ビット)
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)

### ⑥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
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
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
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
DioSts	関数のバックグラウンドでの実行状態の取得
DioStop	バックグラウンドでのデジタル入出力処理の停止

## ⑦割り込み機能関数

DioEvent	割り込みイベントの設定
DioIntEnable	割り込みイベント信号入力の禁止/許可の設定
DioIntSence	割り込みコントロールポートのステータスの取得
DioEventEx	割り込みイベントの設定 (拡張版)
DioIntEnableEx	割り込みイベント信号入力の禁止/許可の設定 (拡張版)

## ⑧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
DioPtnStart	プロセスコントロール開始
DioPtnSts	プロセス開始時のステータスの取得

## ⑨トリガ監視機能関数

DioTrgSet	トリガ監視ビットの設定
DioTrgStart	トリガ開始
DioTrgSts	トリガイイベント発生時のステータスの取得

## ⑩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
DioOutBCD	指定ポートのデータをBCDコードに変換して出力
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)
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)

### TOPICS

#### 『BCD (2進化10進数) とは』

パソコン内部での数値の表現や演算には適している2進数ではありますが、日常10進数を扱い慣れている人間には余り便利な表現とは言えません。このような背景から、2進数を10進数値の感覚で扱える、2進化10進数という概念が生まれました。2進化10進数は、10進数を直接2進数で表すのではなく、10進数の各桁を4桁の2進数で表現したもので、2進化10進数またはBCD (Binary Coded Decimal) と呼ばれています。

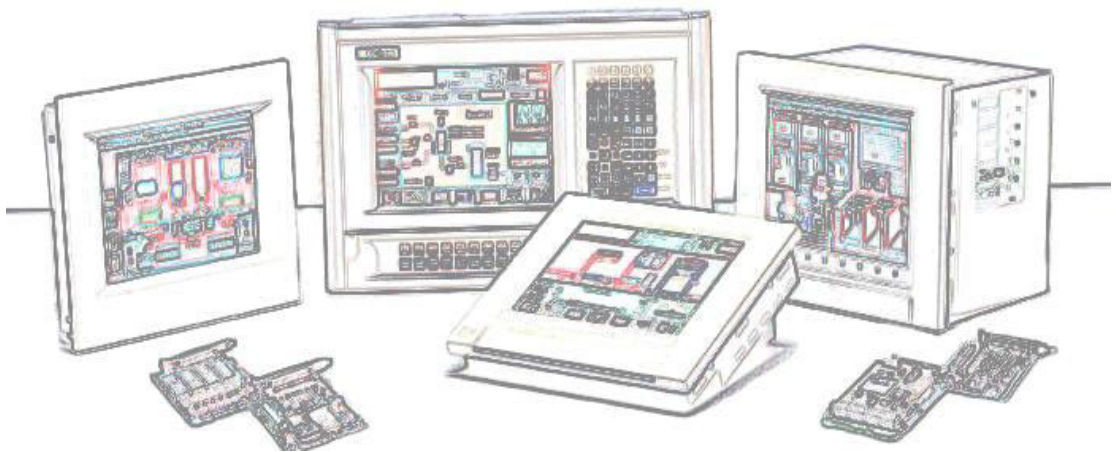


例えば、3桁のBCDコードを出力する機器から、データを入力する場合、4ビット (1桁) × 3 (桁数) = 12ビットが必要になると考えます。この場合、12点 (ビット) 以上の入力点数を持ったデジタル入力ボードが必要です。



# 第6章

## リモートI/O





## 第6章 リモートI/O

第5章では、ノートパソコンとPCカードを使用したシステムを作成しました。通常、PCカードやPCIバスボードを使用したシステムでは、計測や制御を行うパソコンと測定・制御対象（外部機器）とが近接している場合がほとんどです。しかし、システム形態によっては、計測や制御を行うパソコンと測定・制御対象（外部機器）との距離が遠隔になる場合もあります。このようなシステムで採用されるのが『リモートI/O』です。この章では、『リモートI/O』に焦点をあて、その概要と弊社のリモートI/O製品『F&EITシリーズ』の紹介および、プログラミングについての説明をしていきます。

### 6-1. リモートI/Oとは

現在、パソコンはあらゆるオートメーション分野において、その利用の幅を広げています。これらのシステムを構築するには、制御やデータ収集を行うパソコンと、その対象となる外部機器との距離、つまりパソコンの入出力インターフェイスと各種センサやアクチュエータとの間に、多数の信号線が必要となります。パソコンと外部機器とが近接している場合は問題になりませんが、機器が広域に点在している場合、ケーブルを敷設する作業には、大変な労力と多大な費用が必要となってしまいます。このような背景から『リモートI/O』と呼ばれるシステムが発展してきました。

『リモートI/O』とは、点在する計測・制御ポイントに信号入出力の機能を持った機器を配置して、パソコンと離れた場所で直接外部機器との信号入出力を行うシステムです。制御やデータ収集を行うパソコンと信号入出力機器との間は、シリアルケーブルやイーサネットケーブルなどで接続します。このため、大幅な省配線化が図られ、ローコスト&フレキシブルなシステムが実現するのです。

一口に『リモートI/O』と言っても、DeviceNetやCC-Linkのようなオープンフィールドネットワークやイーサネット、Dopa網などを利用した中・大規模なものから、RS-232CやUSBを利用した比較的規模の小さなシステムまで、その形態は様々です。本書では、パソコンの標準インターフェイスであり、利便性と汎用性が最も高いイーサネットベースのリモートI/Oについて説明をしていきます。

### 6-2. イーサネットベースのリモートI/O製品紹介

イーサネットベースのリモートI/Oは、広域に点在する設備の集中監視や制御に適したシステムです。

近年、企業活動の中でイーサネットのネットワークはごく当り前のものとなり、ネットワーク抜きでは仕事が進まない状況になっています。オートメーション設備・装置のインテリジェント化も進んでいます。品質管理や信頼性・稼働率の向上などの問題解決にも末端の情報を収集し、改善するための実用的ネットワークインフラの整備が求められています。

しかし、すべての設備をスクラップ&ビルドするような多大な設備投資は、企業にとってはリスクなことです。そのような実用的ネットワークインフラを実現するためには、いかに既設資産を活用し、設計・工事期間とトータルコストを抑えるかが最も重要なファクターです。

イーサネットベースのリモートI/Oシステムは、より遠隔にある機器の制御と情報収集ができるだけでなく、既設ネットワークインフラを最大限に活用し、情報系と制御系のシステムをシームレスに融合することができます。イーサネットの汎用性をフルに活かしたローコスト&フレキシブルなシステムを実現します。

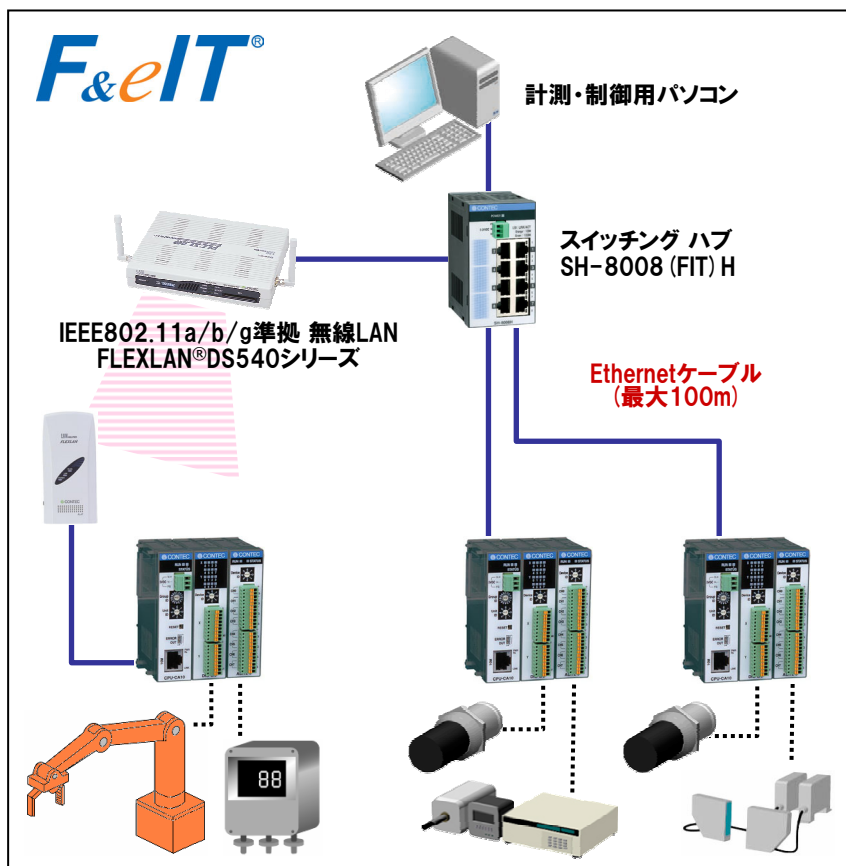
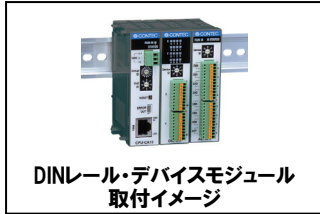


図6-1. 弊社のイーサネットベースリモートI/O製品を使用したシステム例

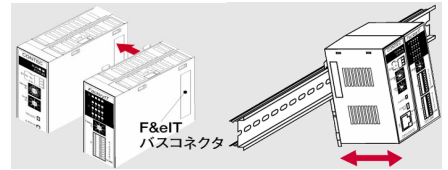
図6-1は、広域に点在する各種センサ・アクチュエータが複数のI/Oコントローラユニット(I/Oコントローラモジュール+I/Oデバイスモジュール)につながり、イーサネットケーブル⇒ハブ経由でパソコンへ接続、各種の計測制御処理を実行している例です。各セグメントを100m範囲で接続することができます。さらに、ケーブル敷設が困難な場所へのI/Oコントローラユニットの設置、建屋間・広域に点在する設備の集中監視・制御に無線LANを併用すれば、さらにフレキシブルなシステムが実現します。弊社のイーサネットベースのリモートI/Oは、遠隔地にある機器をあたかもパソコンの拡張バス(PCIやPCカードなど)をイーサネットで延長した感覚で制御できるようになっています。

## ① I/Oコントローラモジュール CPU-CA20 (FIT) GY

CPU-CA20 (FIT) GYは、コンパクト(ほぼ手のひらサイズ)設計のリモートI/Oコントローラモジュールです。各種I/Oデバイスモジュール(デジタル入出力、アナログ入出力、カウンタ入力)を組み合わせることで、装置にあわせた最適なI/Oを構成することができます(デバイスモジュールは最大8台まで接続可能です)。



- スタック接続したデジタル入出力などのデバイスモジュールの制御を行い、イーサネットを介してホストPCとのデータ送受信を行います。
- Windows用API関数ライブラリ[API-CAP (W32)]を使用すれば、『ネットワーク接続』ということ意識することなく、PCカードやPCバスボードを使用したシステムと同様の感覚でアプリケーションを組むことができます。
- 汎用ソケット関数を使用して、LinuxなどWindows以外のOSでも制御が可能です。
- DDEサーバ[FIT-SVR (W32)]を使用すると、ExcelやSCADA/HMIソフトでのモニタリングが可能です。
- 豊富なデバイスモジュールは、スタック方式のバス形状(F&EITバス)により簡単に着脱できます。DINレールへの取り付けも容易です。



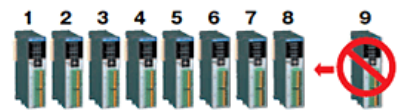
## ② I/Oデバイスモジュール

機能	型式
<b>絶縁型デジタル入出力モジュール</b>	
12~24VDC 入力16点 / 12~48VDC 出力16点	DIO-16/16 (FIT) GY
12~24VDC 入出力各8点	DIO-8/8 (FIT) GY
36~48VDC 入出力各8点	DIO-8/8H (FIT) GY
12~24VDC 入力4点 / 12~48VDC 出力4点	DIO-4/4 (FIT) GY
<b>非絶縁型デジタル入出力モジュール</b>	
TTL (5VDC) 入出力8点	DIO-8D (FIT) GY
<b>絶縁型デジタル入力モジュール</b>	
12~24VDC 入力32点	DI-32 (FIT) GY
12~24VDC 入力16点	DI-16 (FIT) GY
36~48VDC 入力16点	DI-16H (FIT) GY
12~24VDC 入力8点	DI-8 (FIT) GY
<b>絶縁型デジタル出力モジュール</b>	
12~48VDC 出力32点	DO-32 (FIT) GY
12~48VDC 出力16点	DO-16 (FIT) GY
12~48VDC 出力8点	DO-8 (FIT) GY
<b>絶縁型アナログ入力モジュール</b>	
絶縁型アナログ入力 12ビット 8チャンネル	ADI12-8 (FIT) GY
絶縁型アナログ入力 16ビット 4チャンネル	ADI16-4 (FIT) GY

機能	型式
<b>絶縁型アナログ出力モジュール</b>	
絶縁型アナログ出力 12ビット 4チャンネル	DAI12-4 (FIT) GY
絶縁型アナログ出力 16ビット 4チャンネル	DAI16-4 (FIT) GY
<b>Pt100温度センサ入力モジュール</b>	
Pt100温度センサ入力 4チャンネル	PTI-4 (FIT) GY
<b>絶縁型カウンタモジュール</b>	
24ビット UP/DOWN 5~12VDC 2チャンネル	CNT24-2 (FIT) GY
16ビットUP 12~24VDC 8チャンネル	CNT16-8 (FIT) GY
16ビットUP 5VDC 8チャンネル	CNT16-8L (FIT) GY
<b>リードリレー-接点出力モジュール</b>	
125VAC/30VDC 2A リードリレー-接点出力 4点	RRY-4 (FIT) GY
<b>シリアル コミュニケーションモジュール</b>	
RS-232C 2チャンネル	COM-2 (FIT) GY ※
RS-422A/485 1チャンネル	COM-1PD (FIT) GY ※
<b>GPIO コミュニケーションモジュール</b>	
GPIO (IEEE-488) 1チャンネル	GP-IB (FIT) GY ※

※：CPU-CA20 (FIT) GYでは使用できません。

1ユニットに最大8モジュールまでスタックできます。  
ただし、接続するデバイスモジュールの消費電流の総和が3Aを越える組み合わせはできません。  
詳しくは、ホームページまたは『F&EITカタログ』を参照してください。



最大8モジュール  
(消費電流総和3A以下)

**F&EITシリーズ スペシャルサイト** <http://www.contec.co.jp/fit/>



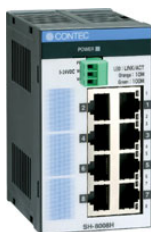


#### ④ F&eITシリーズ スwitching HUB SH-8008 (FIT) H

組み込みに便利な超小型・軽量の汎用Switching HUBです。外形幅はわずか52.4mmのコンパクト設計のため、一般的なHUBではスペース上で設置が困難な盤内、装置内にも設置可能です。

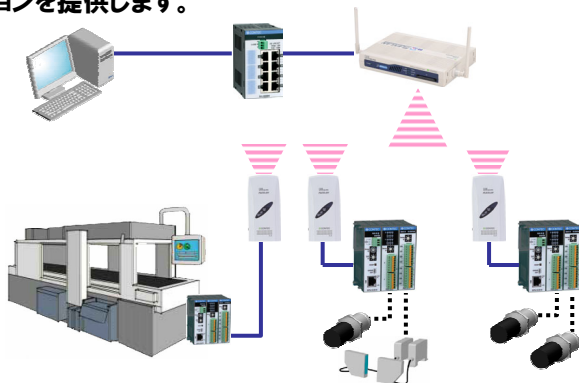
電源仕様は5~24VDCのワイド入力に対応しているため、専用の外部電源ユニットを使用せずに、装置内の電源を使用することもできるなど組み込み用途に適した仕様となっています。

- オートネゴシエーションによる通信速度・通信モード自動認識機能、Auto-MDI/MDI-X機能を備えた8ポートを搭載。
- 35mmDINレール取り付け機構を標準装備。



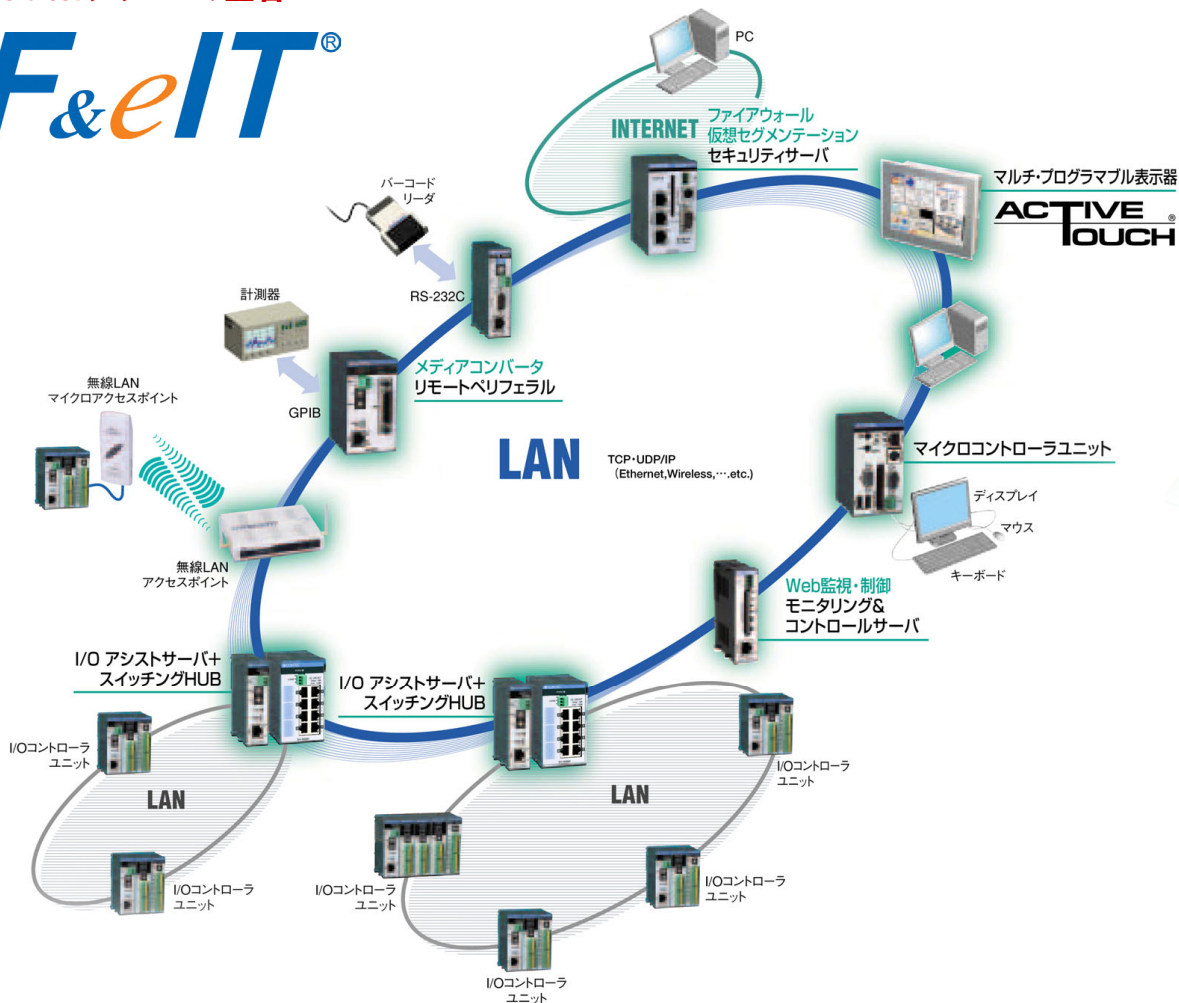
#### ⑤ 無線LANアクセスポイントとステーション FLEXLAN DS540シリーズ

FLEXLAN DS540シリーズは、IEEE802.11a/b/gに準拠した無線LAN製品シリーズ(アクセスポイント・ステーション)です。ハード・ソフトともに自社開発という利点を活かして、高度なセキュリティと機能、優れた信頼性と保守性を実現しています。ケーブル敷設が困難な場所へのI/Oコントローラ設置をはじめ、建屋間・広域に点在する設備の集中監視・制御に最適なソリューションを提供します。



#### ⑥ F&eITシリーズの全容

# F&eIT<sup>®</sup>



F&eIT<sup>®</sup>シリーズの詳細は、『F&eITカタログ』またはホームページを参照してください。

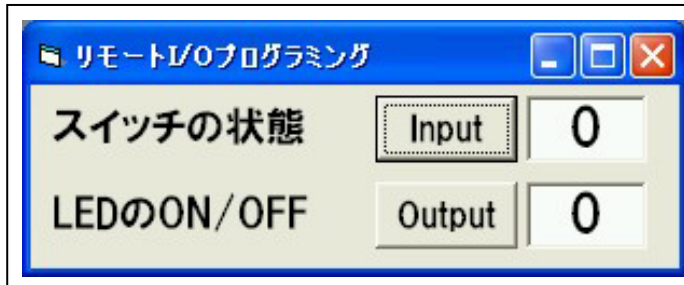
**F&eIT<sup>®</sup>シリーズ スペシャルサイト**  
<http://www.contec.co.jp/fit/>

## 6-3.リモートI/Oによるデジタル入出力プログラミング (ビット単位)

それでは、弊社F&EITシリーズを使用したデジタル入出力リモートI/Oシステムを作成していきます。初歩的なVisual Basicの知識さえあれば、PCカードやPCIバスボードと同様の感覚でリモートI/Oシステムの構築が可能なお分かりいただけると思います。

### 6-3-1.リモートI/Oによるデジタル入出力プログラム概要

遠隔地にあるスイッチのON/OFF情報の入力とLED (ランプ) のON/OFF制御を行うプログラムを作成します。



実行画面イメージ

### 6-3-2.使用機器

#### ① ホストPC



- OS : Windows XP Professional/Home Edition
- 開発言語 : Visual Basic 6.0
- ソフトウェア : I/Oコントローラモジュール用 Windows ドライブライブラリ API-CAP (W32)

#### ② I/Oコントローラモジュール CPU-CA20 (FIT) GY



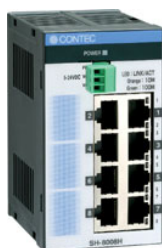
#### ③ I/Oデバイスモジュール 絶縁型デジタル入出力モジュール DIO-8/8 (FIT) GY (12~24VDC 入出力各8点)



#### ④ ACアダプタ電源×2台 入力90-264VAC 出力5VDC 2.0A POA200-20



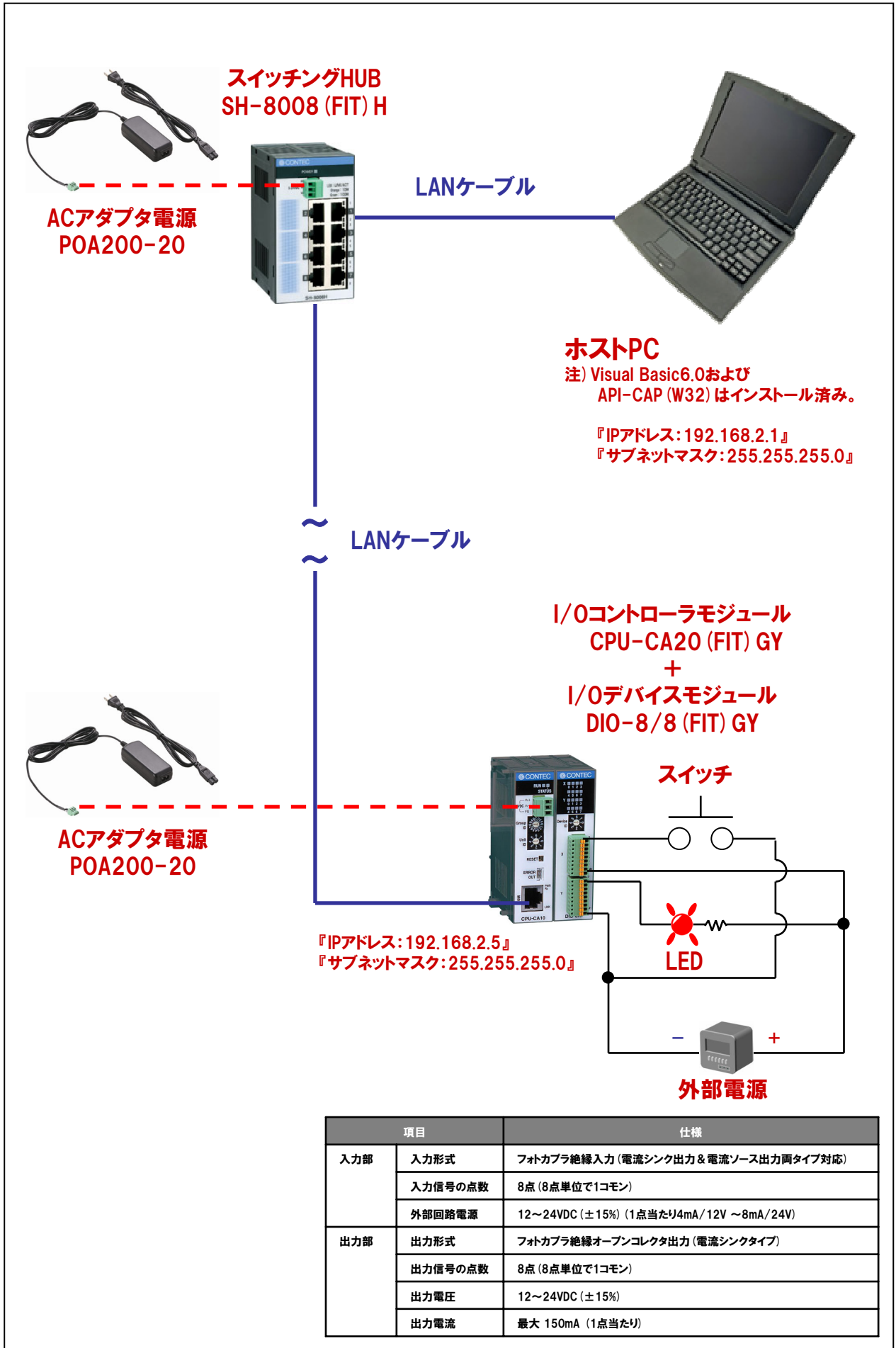
#### ⑤ スイッチングHUB 組み込み型10M/100M自動認識スイッチングHUB SH-8008 (FIT) H



#### ⑥ その他

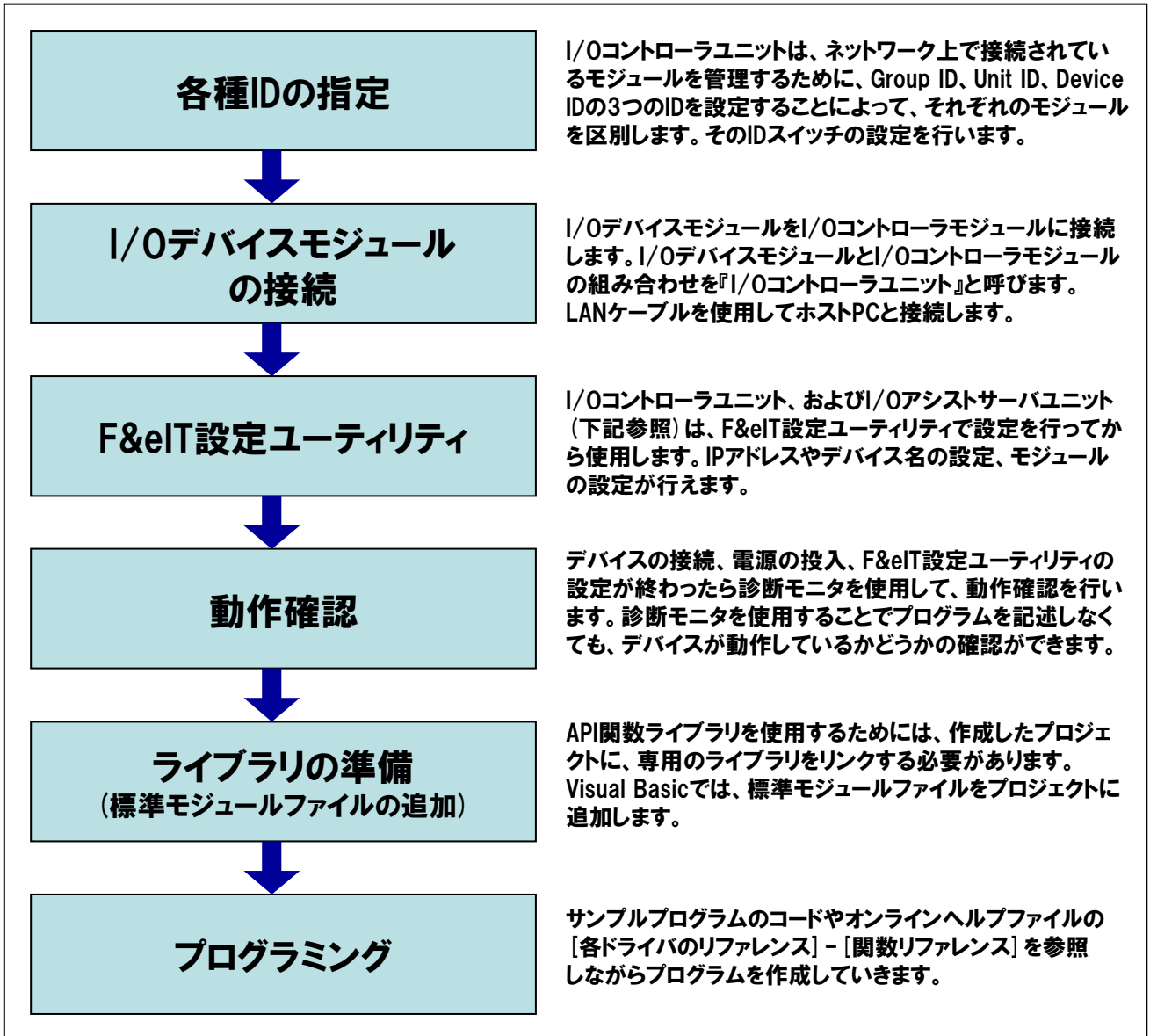
- (1) LANケーブル×2本
- (2) スイッチ・LED接続用ケーブル
- (3) スイッチ
- (4) LED
- (5) 外部電源





項目		仕様
入力部	入力形式	フォトブラ絶縁入力 (電流シンク出力 & 電流ソース出力両タイプ対応)
	入力信号の点数	8点 (8点単位で1コモン)
	外部回路電源	12~24VDC (±15%) (1点当たり4mA/12V ~8mA/24V)
出力部	出力形式	フォトブラ絶縁オープンコレクタ出力 (電流シンクタイプ)
	出力信号の点数	8点 (8点単位で1コモン)
	出力電圧	12~24VDC (±15%)
	出力電流	最大 150mA (1点当たり)

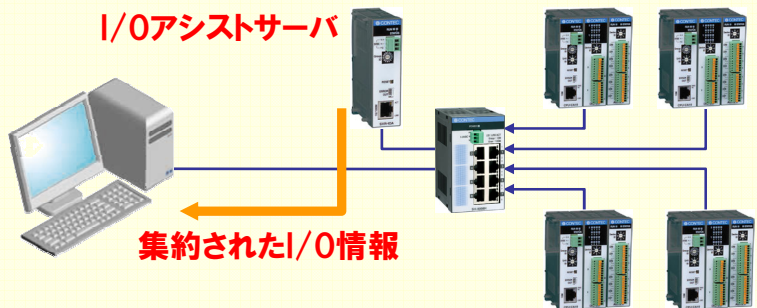
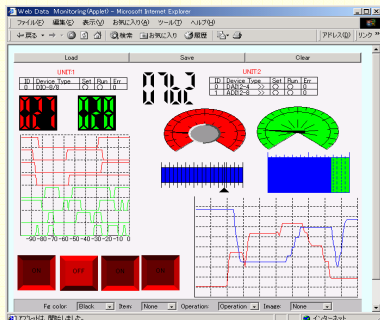




**TOPICS**

『I/Oアシストサーバユニット:SVR-IOA2 (FIT) GYとは』

最大8台のI/Oコントローラからの情報を集約し、上位ホストからは『1回のアクセス』で全てのI/O情報収集を可能にするF&eITシリーズ I/Oコントローラユニット統合管理&Webモニタリングサーバユニットです。回線の負荷軽減とプログラムレスWebモニタリングを実現できます。システムの規模が大きくなった時には、非常に有効ですが、本書の演習では使用しません。



## 6-3-5.ハードウェアの準備 (各種IDの設定)

I/Oコントローラユニット (I/Oコントローラモジュール+I/Oデバイスモジュール) は、ネットワーク上で接続されているモジュールを管理・区別するために、『Group ID』『Unit ID』『Device ID』の3つのIDを使用します。

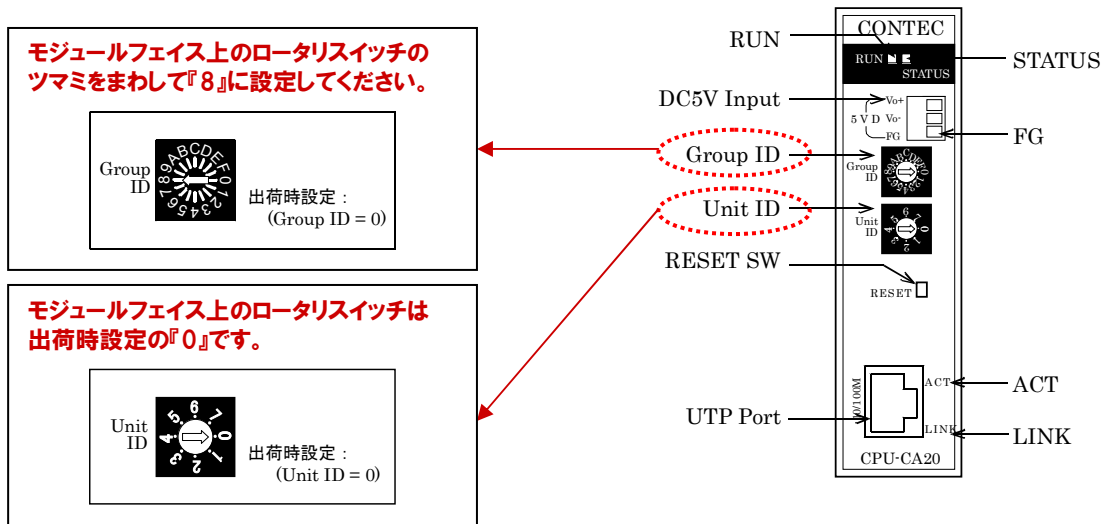
### ① I/Oコントローラモジュール CPU-CA20 (FIT) GYの設定

#### ■ Group IDの設定

『Group ID』は、I/Oアシストサーバユニットの使用有無 (Group ID:0~7はI/Oアシストサーバモード) と、I/Oアシストサーバユニットに管理させるI/Oコントローラユニットを特定するためのIDスイッチです。本書では、I/Oアシストサーバを使用しないため、『Group ID』は単独起動モードの『8』に設定します。

#### ■ Unit IDの設定

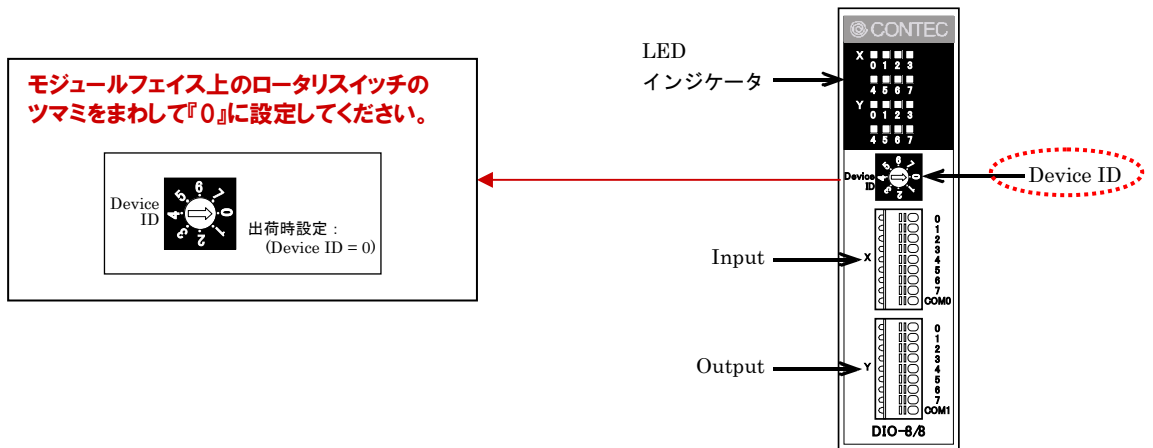
『Unit ID』は、同一『Group ID』内でI/Oコントローラユニットを区別するためのIDです。概念的には、『Group ID』の下位に属するものです。I/Oアシストサーバを使用しない場合、『Unit ID』によってI/Oコントローラユニットが区別され、『Unit ID』:0~7までの計8台が使用できます。本書では『Unit ID』は『0』に設定します。



### ② I/Oデバイスモジュール DIO-8/8 (FIT) GYの設定

#### ■ Device IDの設定

『Device ID』は、単一のI/Oコントローラユニットに接続されたデバイスモジュール同士を区別するためのIDです。『Device ID』は、0~7の範囲で設定でき、最大8台までのモジュールを区別できます。デバイスIDは一つのI/Oコントローラユニット内で重複しないように設定します。本書では、DIO-8/8 (FIT) GYの『Device ID』は『0』とします。

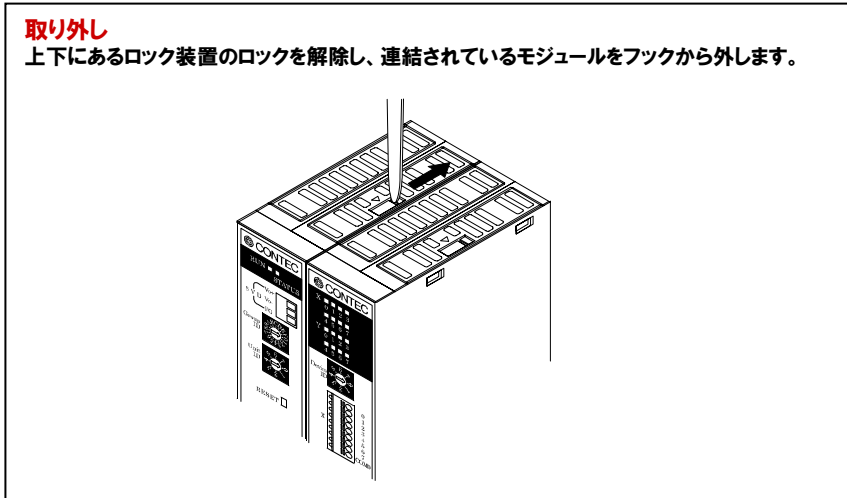
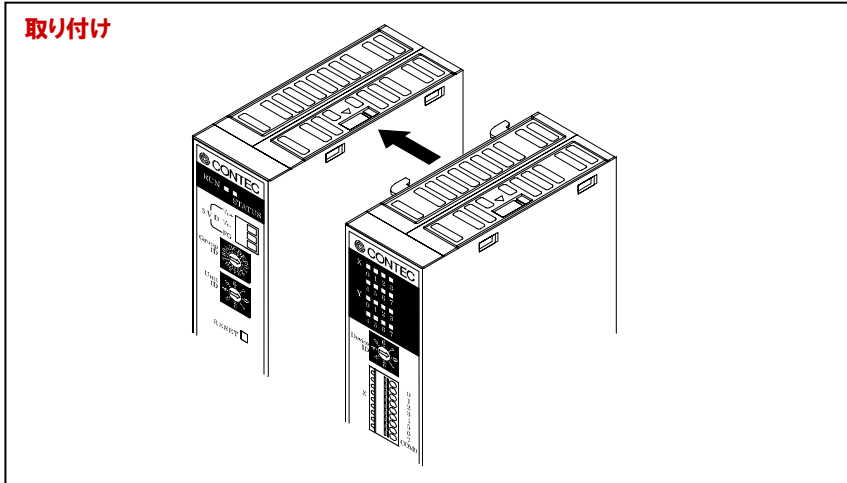


## 6-3-6.ハードウェアの準備 (I/Oデバイスモジュールの接続)

I/OコントローラモジュールとI/Oデバイスモジュールの接続および、電源ユニット、スイッチングHUBとの接続を行います。デバイスモジュールは、F&EITバスによるスタック接続で簡単に着脱が可能です。

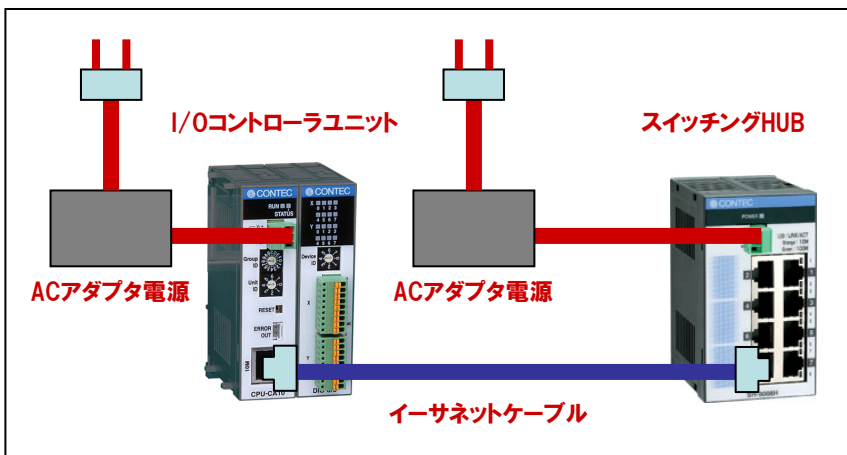
### ① I/OコントローラモジュールとI/Oデバイスモジュールとの接続

- (a) モジュールには、スタック接続するためのスタック用フックと連結するためのロック装置（上下2カ所）があります。スタック用フックを相手側のフック挿入口に合わせて差し込むと、モジュールは自動的にロックされます。モジュールを取り外す際は、上下にあるロック装置のロックを解除し、連結されているモジュールをフックから外します。



- (b) スwitchingHUBとの接続とACアダプタ電源の接続

I/OコントローラモジュールとスイッチングHUB（本書ではSH-8008 (FIT) H）とをイーサネットケーブルで接続します。また、それぞれのモジュールにACアダプタ電源（本書ではPOA200-20）を接続します。ただし、ACコンセントは、【6-3-7.】外部機器との接続完了後に入れてください。



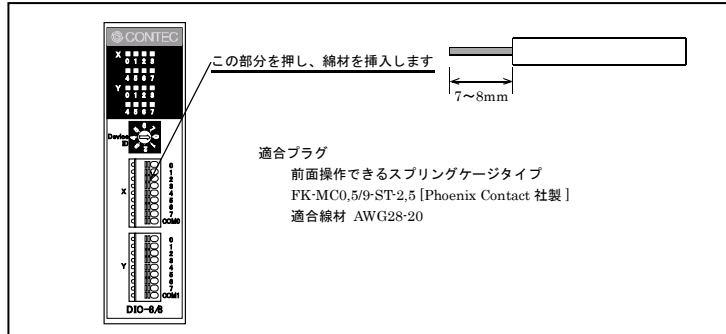
## 6-3-7.ハードウェアの準備 (I/Oデバイスモジュールと外部機器との接続)

I/Oデバイスモジュール (DIO-8/8 (FIT) GY) と外部機器 (本書では、スイッチ (入力) とLED (出力)) との接続を行います。すべての接続が完了した時点で各ユニットおよび外部機器の電源を投入します。

### ① I/Oデバイスモジュール (DIO-8/8 (FIT) GY) と外部機器との接続

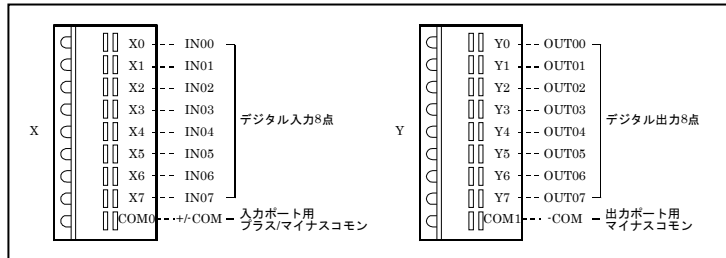
#### (a) 接続方法

DIO-8/8 (FIT) GYと外部機器を接続する場合は、添付されているコネクタプラグを使用します。配線を行う場合は、線材の被覆部を約7~8mm程度ストリップした後、コネクタプラグのオレンジ色のボタンを押しながら挿入します。挿入後オレンジ色のボタンをはなすと、線材は固定されます。適合線材はAWG28~20です。



#### (b) インターフェイスコネクタの信号配置

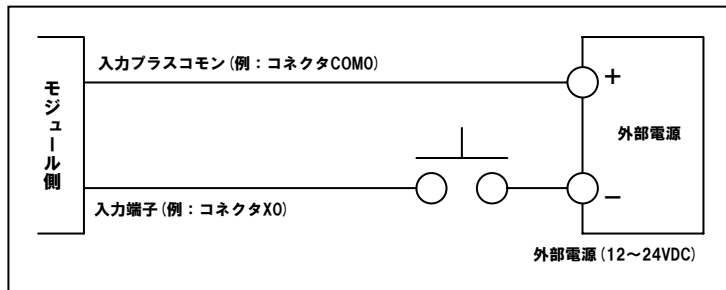
外部装置の接続は、モジュールのフェイスに装備された9ピンのコネクタで行います。



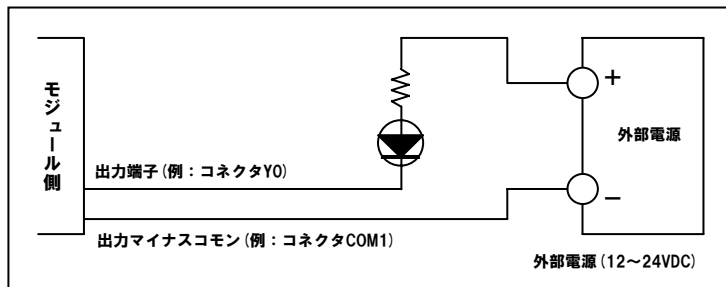
#### (c) 外部機器との接続例

本書のシステムでは、入力の+0ビット目 (信号名: X0) にスイッチを、出力の+0ビット目 (信号名: Y0) にLEDを接続します。

##### ● 入力部



##### ● 出力部



### 6-3-8.ハードウェアの準備 (ホストPCとI/Oコントローラユニットとの接続)

出荷時の設定ではI/OコントローラユニットのIPアドレスは、LANコントローラのMACアドレスを使用して自動生成されています。したがって、出荷時のIPアドレスが重複することはありません。機器の側面の製品ラベルにM/Aという項目が機器のIPアドレス (MACアドレス) を表しています。

例) M/A: 『00.80.4C.01.02.03』の場合

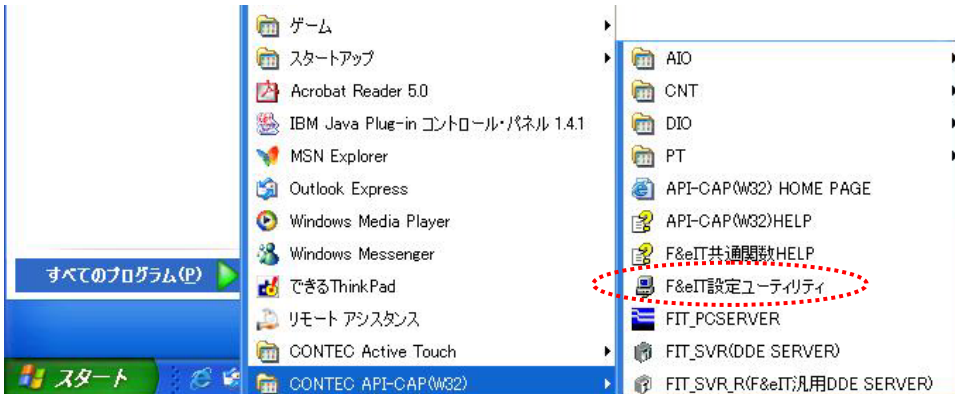
『00.80.4C』はベンダーコードと呼ばれ全機器共通になっており、IPアドレス生成では『10』として扱います。この機器の出荷時IPアドレスは、『00.80.4C』 = 『10』ですので、『10.01.02.03』に自動設定されています。なお、サブネットマスクは、『255.0.0.0』に設定されています。

- ① I/Oコントローラユニットの『Group ID』が『8』、『Unit ID』が『0』であることを確認します。  
また、デバイスモジュールの『Device ID』設定が重複していないことを確認します (今回は『0』です)。
- ② I/OコントローラユニットとスイッチングHUBにACアダプタおよびネットワークケーブルを接続します。

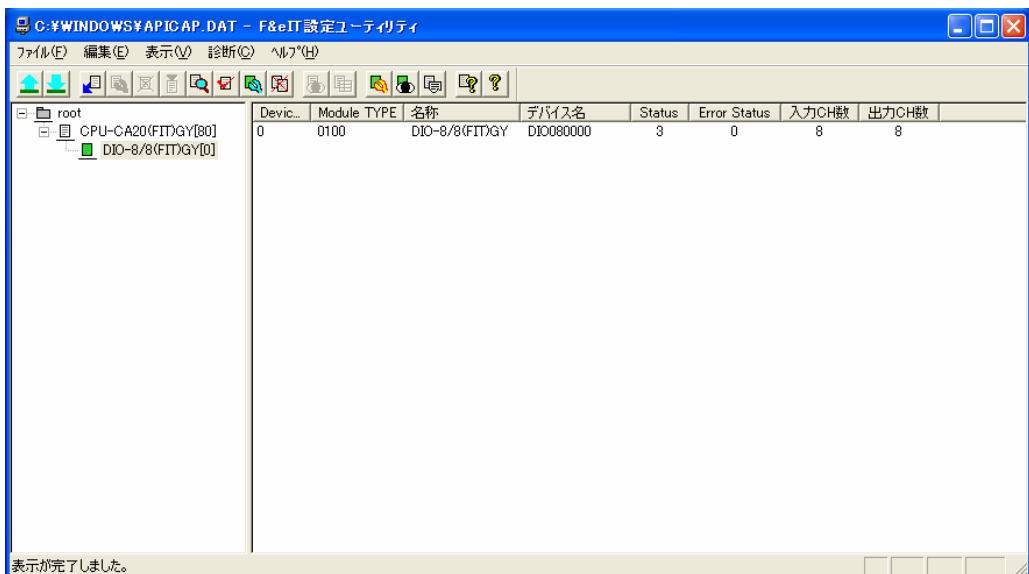
### 6-3-9.F&eIT設定ユーティリティによるセットアップ (ユーティリティの実行)

F&eIT設定ユーティリティは、I/OコントローラモジュールおよびI/OアシストサーバユニットをAPI-CAP (W32) で使用する場合に必要となる設定を行い、設定ファイルに保管する機能を持ったプログラムです。ネットワーク設定、デバイス名の設定、デバイスモジュールの診断・設定などを行うことができます。

『スタート』 - 『プログラム』 - 『CONTEC API-CAP (W32)』 - 『F&eIT設定ユーティリティ』の手順で、F&eIT設定ユーティリティを起動します。メイン画面が開かれ、同一ネットワーク上のF&eIT機器を自動検索し、ツリー形式にて一覧表示を行います。

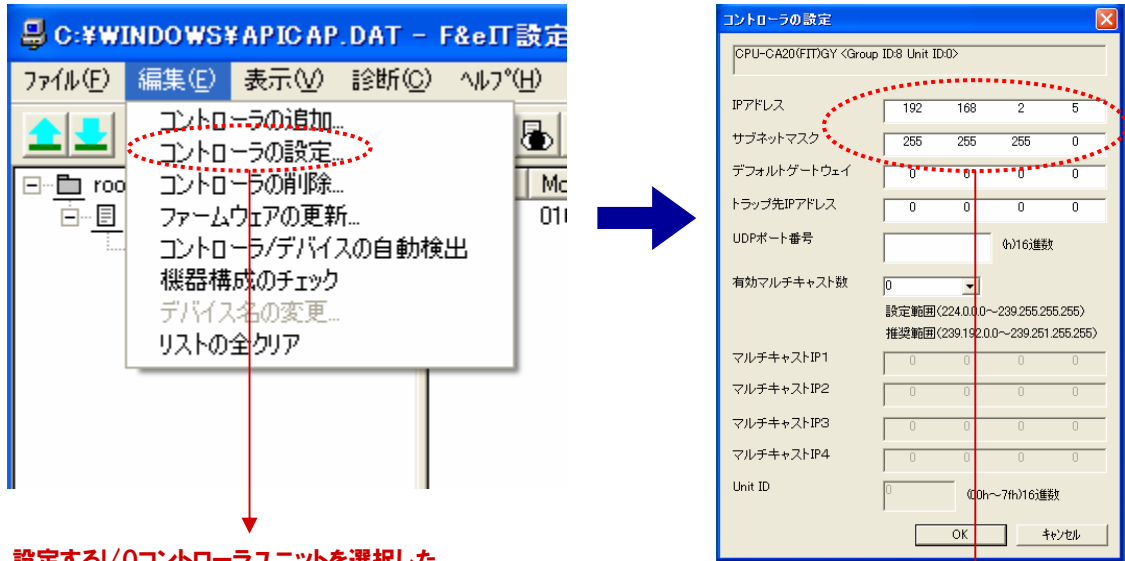


#### ■ F&eIT設定ユーティリティのメイン画面



### 6-3-10.F&eIT設定ユーティリティによるセットアップ(ネットワークの設定)

検出されたコントローラに対して、使用するネットワーク環境に合わせた、IPアドレスやサブネットマスクなどのコントローラ設定を行います。左のツリービューから設定するコントローラを選択し、右クリックするとポップアップメニューが開かれます。そのメニューから『コントローラの設定』を起動します(または、メニュー『編集』 - 『コントローラの設定』を選択します)。ネットワーク設定で設定された項目は、I/Oコントローラユニット本体に記憶されていますので、I/Oコントローラユニットの電源を落しても、次回起動時に有効になります



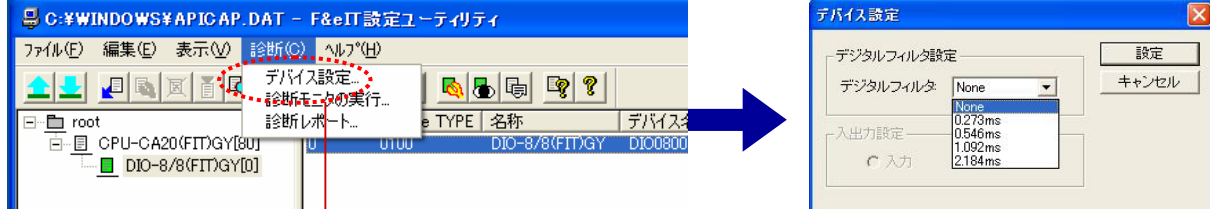
The screenshot shows the 'F&eIT 設定' utility. On the left, a tree view shows a controller selected. A context menu is open with 'コントローラの設定' circled in red. An arrow points to the 'コントローラの設定' dialog box on the right. The dialog shows network settings for 'CPU-CA20(FIT)GY <Group ID:8 Unit ID:0>'. The IP address is set to 192.168.2.5 and the subnet mask to 255.255.255.0, both circled in red. Below the dialog, red text specifies the settings used in the book.

設定するI/Oコントローラユニットを選択した状態で『編集』 - 『コントローラの設定』を選択します。

本書では、下記の通りの設定とします。  
IPアドレス:192.168.2.5  
サブネットマスク:255.255.255.0

### 6-3-11.F&eIT設定ユーティリティによるセットアップ(デバイス固有の設定)

デジタル入力デバイスにおけるデジタルフィルタや、アナログ入力デバイスの入力レンジなど、デバイス固有の設定を行うことができます。これらの設定は、API関数によっても行うことができますが、F&eIT設定ユーティリティでも設定可能です。



The screenshot shows the 'F&eIT 設定ユーティリティ' utility. The tree view shows a device 'DIO-8/8(FIT)GY[0]' selected. A context menu is open with 'デバイスの設定' circled in red. An arrow points to the 'デバイス設定' dialog box on the right. The dialog shows digital filter settings for 'デジタルフィルタ設定'. The digital filter is set to 'None', and the input range is set to 'None', both circled in red. Below the dialog, red text explains the setting.

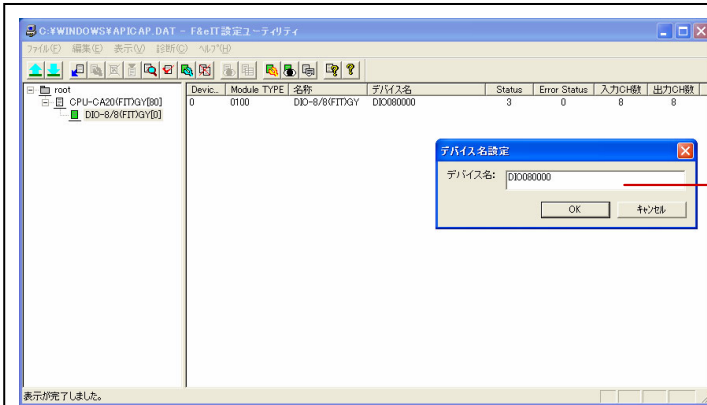
設定するI/Oデバイスモジュール(本書では、DIO-8/8(FIT)GY)を選択状態にして、『診断』 - 『デバイスの設定』を選択します。

DIO-8/8(FIT)GYは、デジタルフィルタの時定数を設定することができます。本書では『None』(使用しない)を設定します。



## 6-3-12.F&eIT設定ユーティリティによるセットアップ(デバイス名の設定)

API-CAP (W32) の関数でアクセスするためにデバイス毎に付ける論理名を設定します。デバイス名は、同一システム内で重複しないように設定する必要があります。API-CAP (W32) は、このデバイス名を使用したアクセスを実現することで、IPアドレスなどのネットワーク接続を意識しないプログラミングを可能にしています。



本書では、デフォルトで付けられている『DIO080000』を使用します。

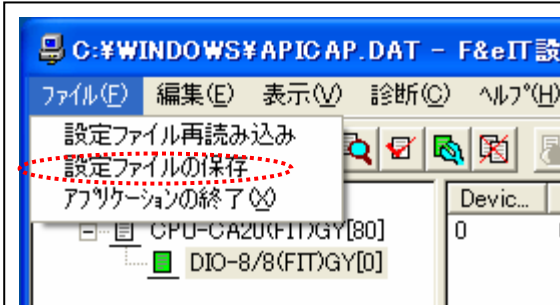
デフォルトの命名規則は下記の通りです。  
『デバイス機能、GroupID、UnitID、DeviceID』  
プログラミングでは各関数を実行する際、対象機器を指定するために使用しますので、分かりやすい名称に変更することもできます。

設定するI/Oデバイスモジュール(本書では、DIO-8/8 (FIT) GY) を選択状態にして、『編集』-『デバイス名の設定』を選択します。

## 6-3-13.F&eIT設定ユーティリティによるセットアップ(設定の保存)

設定が完了したら、メニューの『ファイル』-『設定ファイルの保存』を選択して設定を保存します。

デバイスの設定、診断を行ったりAPI-CAP (W32) を使用したアプリケーションを実行する場合は、設定ファイルを保存しておく必要があります。



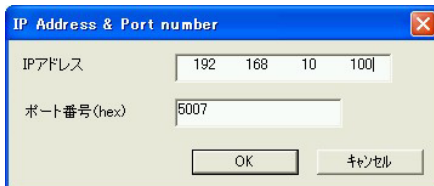
I/Oコントローラユニットを制御するパソコンで、使用するデバイスのデバイス名(デバイスの識別子)やIPアドレスを、設定ファイルとしてディスクに保存します。API関数ライブラリは、この設定ファイルを読み込んで動作します。

設定ファイルはテキストファイルであり、Windowsフォルダに『APICAP.DAT』というファイル名で保存されます。設定ユーティリティの入っていない他のパソコンでAPI関数を用いて作成したアプリケーションを動作させたい場合、この設定ファイルと必要なDLLをコピーして対象のマシンに持っていくことにより、F&eIT設定ユーティリティでの再操作なしで動作させることが可能です。

### ■他のネットワークにあるコントローラを使用する場合

F&eIT設定ユーティリティを起動すると、同一ネットワーク上にあるF&eIT機器のコントローラは自動検出されますが、他のネットワーク上にあるコントローラ(ルータを越える場合)は、IPアドレスを指定して、設定ファイルに情報を取り込まなければなりません。

- ①コントローラにアクセスするパソコンから、F&eIT設定ユーティリティを起動して下さい。
- ②『編集』メニューから『コントローラの追加』を選択してください。
- ③IPアドレスを入力して『OK』ボタンを押し、コントローラの情報を取り込んでください。



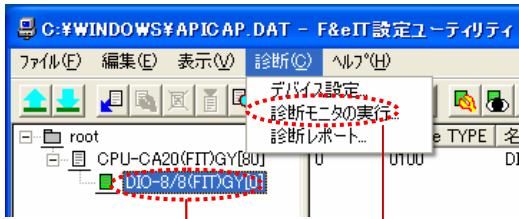
- ④『ファイル』メニューから『設定ファイルの保存』を実行してください。

注意: 他のネットワーク上にあるコントローラ(ルータを越える場合)を使用する場合には、コントローラのデフォルトゲートウェイが正しく設定されていなければなりません。

## 6-3-14.動作確認 (診断モニタの実行)

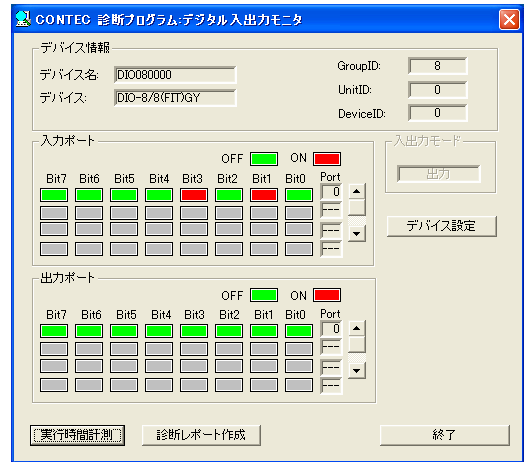
F&eIT設定ユーティリティの設定が終わったら診断モニタを使用し、デバイス動作確認と配線の確認を行います。診断モニタを使用することでプログラムを記述しなくても、デバイスが動作しているかどうかの確認ができます。

### ■ デジタル診断モニタの実行



DIO-8/8 (FIT) GYを  
選択状態にします。

メニューの『診断』 -  
『診断モニタの実行』  
を選択します。

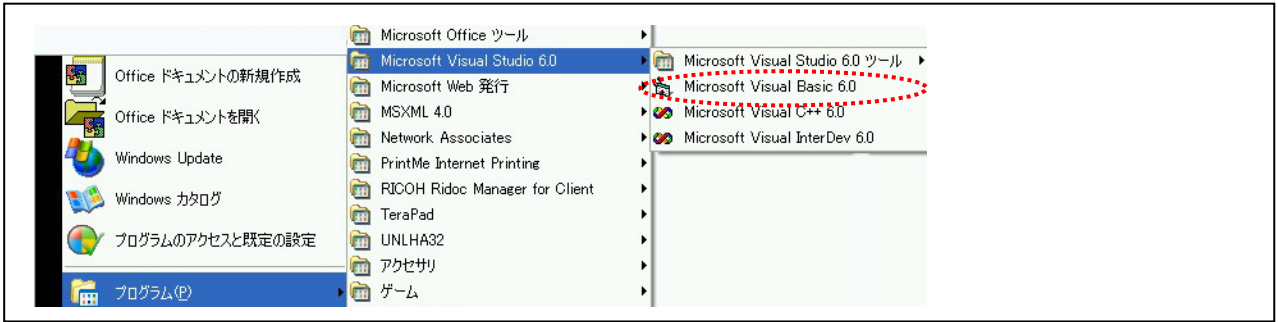


デジタル診断モニタ (CDIODIAG.EXE) は、100ms  
間隔でデジタルデバイスの入力ポートと出力ポート  
をモニタします。出力ポートの赤と緑の表示部分  
をクリックすると、出力データが切り替わります。  
レポート出力も可能です。

## 6-3-15. Visual Basicの起動と画面の作成

まず最初にVisual Basic6.0を起動しましょう。インストール時の設定が標準で、その後変更していなければ、次の手順で起動します。

『スタートメニュー』 - 『プログラム』 - 『Microsoft Visual Studio6.0』 - 『Microsoft Visual Basic6.0』を選択します。



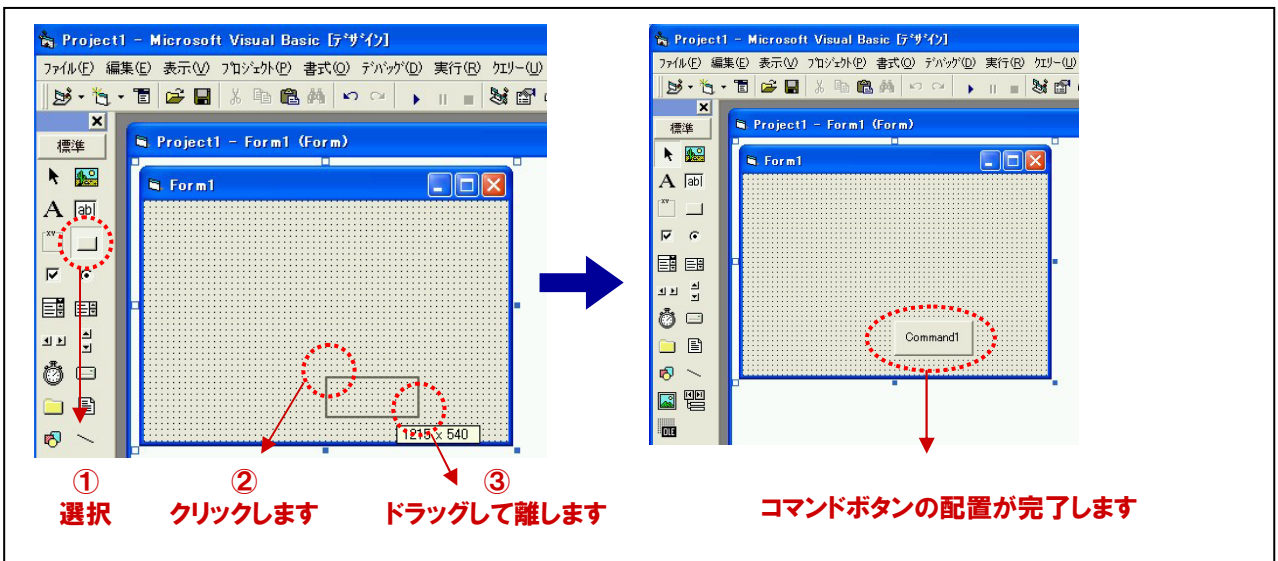
『新しいプロジェクト』ダイアログボックスが表示されますので(※)、『新規作成』タブ内の『標準EXE』を選択して、『開く』ボタンをクリックしてください。

(※: 『新しいプロジェクト』ダイアログボックスが表示されない場合は、メニューの『ファイル』 - 『新しいプロジェクト』を選択すると同様のダイアログボックスが表示されます。)



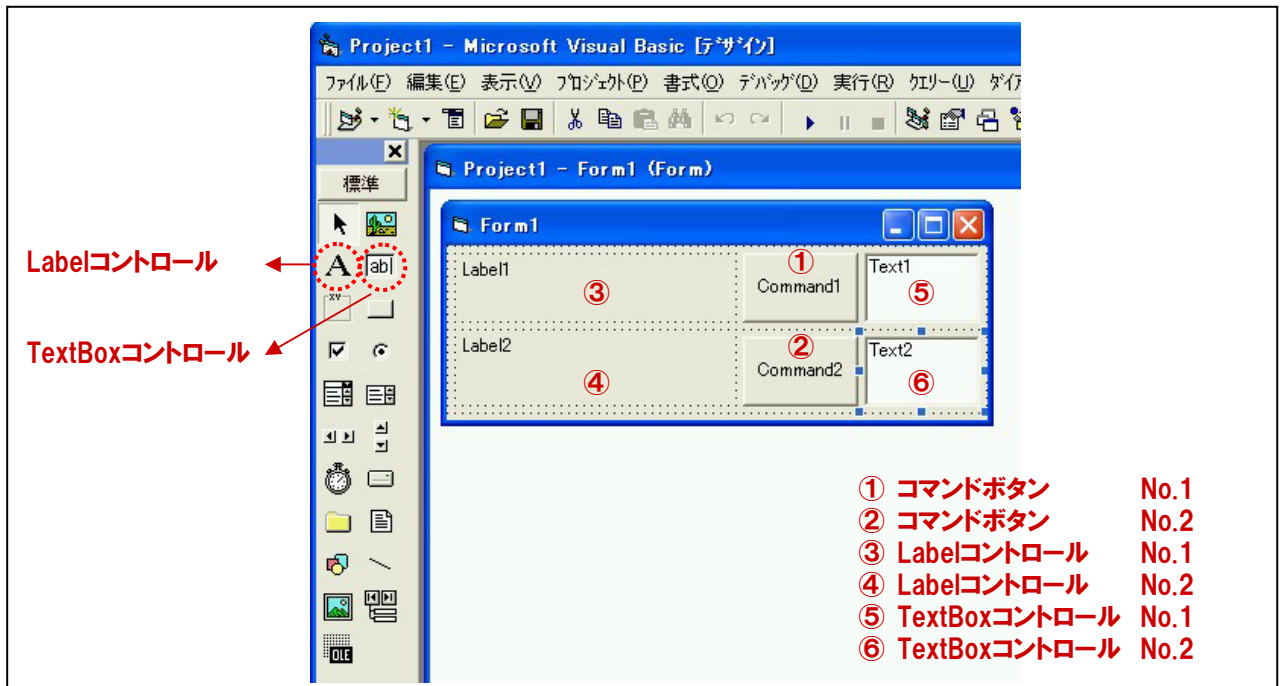
本プログラムにて使用する『コントロール』をフォーム上に配置します。まず最初に『コマンドボタン コントロール』をフォーム上に配置しましょう。

- ① コントロールツールボックスの『コマンドボタン』のアイコンをクリックします。
- ② フォーム上にマウスカursorを移動してクリックし、クリックしたまま右斜め下方向にドラッグ(移動)します。
- ③ クリックした左ボタンを離れた時点で、『コマンドボタン』が配置されます(大きさの変更も同様作業です)。
- ④ 配置後の移動は、『コマンドボタン』をクリックして選択した後、ドラッグ操作で任意の場所に移動できます。



## 6-3-16.画面構成

同様の手順で、2つ目の『コマンドボタン』と2つの『Labelコントロール』、2つの『TextBox』をフォーム上に配置します。



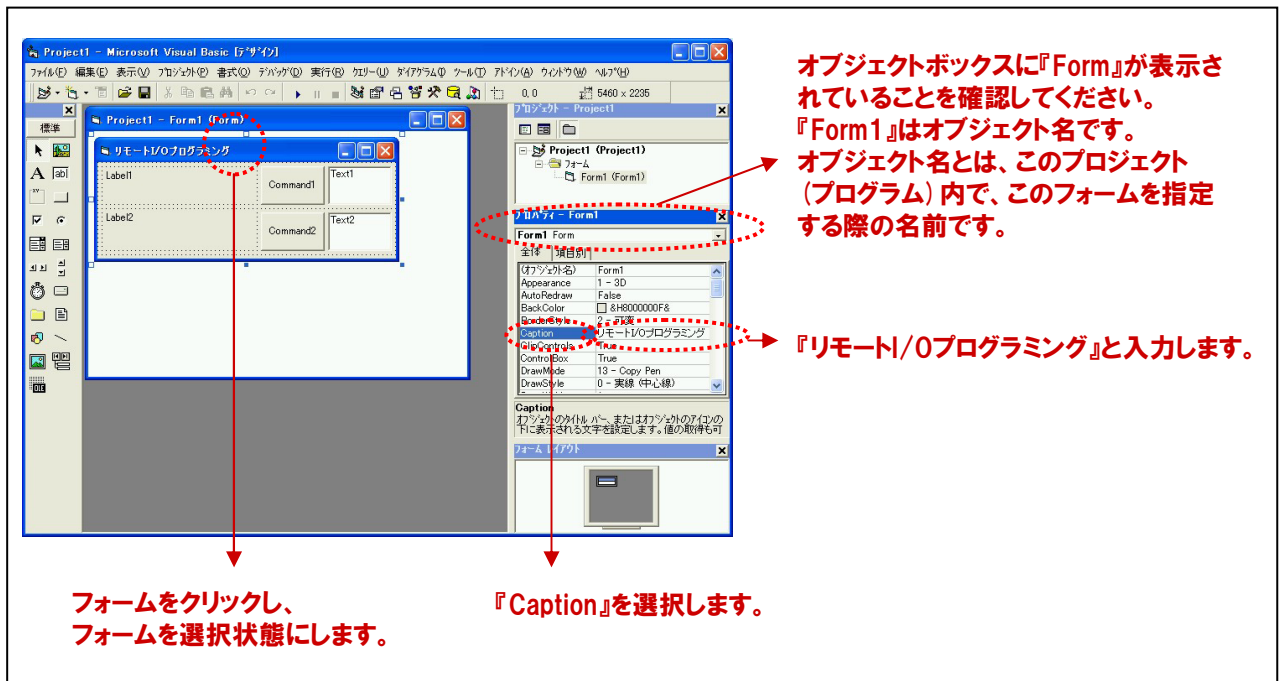
The screenshot shows the Microsoft Visual Basic IDE with a form titled 'Form1' containing six controls: Label1, Label2, Command1, Command2, Text1, and Text2. The controls are numbered 1 through 6. On the left, the '標準' (Standard) toolbox is visible, with 'Labelコントロール' and 'TextBoxコントロール' highlighted by red arrows. A legend on the right lists the controls and their IDs:

① コマンドボタン	No.1
② コマンドボタン	No.2
③ Labelコントロール	No.1
④ Labelコントロール	No.2
⑤ TextBoxコントロール	No.1
⑥ TextBoxコントロール	No.2

## 6-3-17.各オブジェクトのプロパティ設定

フォームおよび各コントロールのプロパティ設定を行います。プロパティの設定方法は、プロパティウィンドウで設定する方法と、コード(プログラム上)で設定する方法がありますが、本プログラムにおいてはすべてプロパティウィンドウで設定することとします。

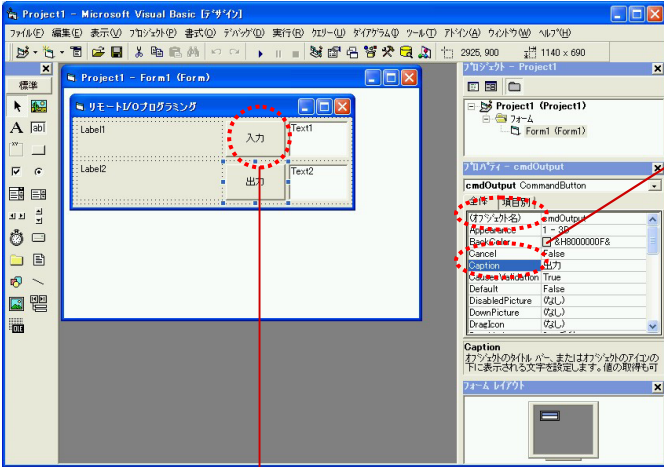
- ① フォームの『タイトル』を変更します。フォームをクリックして選択、プロパティウィンドウのプロパティリストの中から『Caption』プロパティを選択して、設定ボックスに『リモート/0プログラミング』と入力します。



The screenshot shows the Microsoft Visual Basic IDE with the 'リモート/0プログラミング' (Remote/0 Programming) property window open for 'Form1'. The 'Caption' property is selected, and the value 'リモート/0プログラミング' is entered. Red arrows and text provide instructions:

- オブジェクトボックスに『Form』が表示されていることを確認してください。『Form1』はオブジェクト名です。オブジェクト名とは、このプロジェクト(プログラム)内で、このフォームを指定する際の名前です。
- 『リモート/0プログラミング』と入力します。
- フォームをクリックし、フォームを選択状態にします。
- 『Caption』を選択します。

- ② コマンドボタンの『オブジェクト名』と『タイトル』を変更します。コマンドボタン①を選択、プロパティウィンドウのプロパティリスト『(オブジェクト名)』を選択して、『cmdInput』と入力します。続いて、プロパティリストの中から、『Caption』プロパティを選択して、設定ボックスに『入力』と入力します。同様にコマンドボタン②は、オブジェクト名に『cmdOutput』、Captionプロパティに『出力』と入力します。



**コマンドボタン①のプロパティ設定**

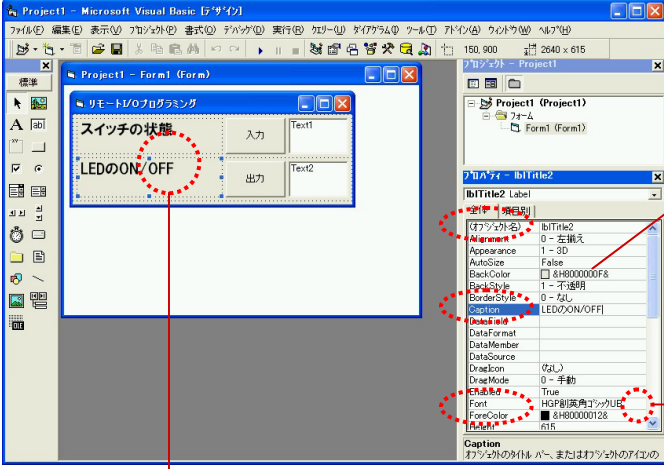
- オブジェクト名 : cmdInput
- Caption : 入力

**コマンドボタン②のプロパティ設定**

- オブジェクト名 : cmdOutput
- Caption : 出力

コマンドボタンをクリックし、選択状態にします。

- ③ Labelコントロールの『オブジェクト名』、『Caption』、『font』プロパティを変更します。Labelコントロール①を選択、プロパティウィンドウのプロパティリスト『(オブジェクト名)』を選択して、『lblTitle1』と入力します。続いて、プロパティリストの中から、『Caption』プロパティを選択して、設定ボックスに『スイッチの状態』と入力します。最後に『font』プロパティを変更します。『font』プロパティを選択するとプロパティリスト中に□ボタンが表示されますので、そのボタンをクリックします。『フォントの設定ダイアログ』が表示されますので、好みのフォントスタイルに変更してください。同様にLabelコントロール②は、オブジェクト名に『lblTitle2』、Captionプロパティに『LEDのON/OFF』と入力します。



**Labelコントロール①のプロパティ設定**

- オブジェクト名 : lblTitle1
- Caption : スwitchの状態
- font : 任意

**Labelコントロール②のプロパティ設定**

- オブジェクト名 : lblTitle2
- Caption : LEDのON/OFF
- font : 任意

Labelコントロールをクリックし、選択状態にします。

クリックすると『フォントの設定ダイアログ』が表示されますので、任意のフォントスタイルを設定してください。

**フォント**

フォント名(F): 標準

スタイル(S): 標準

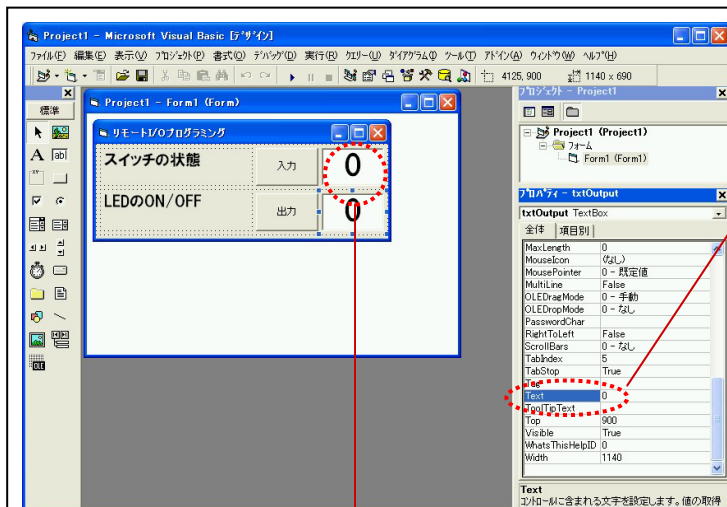
サイズ(S): 9

文字飾り: 無

文字セット(C): 日本語



- ④ TextBoxコントロールの『オブジェクト名』と『タイトル』を変更します。TextBoxコントロール①を選択、プロパティウィンドウのプロパティリスト『(オブジェクト名)』を選択して、『txtInput』と入力します。続いて、プロパティリストの中から、『Text』プロパティを選択して、設定ボックスに『0』と入力します。同様にTextBoxコントロール②は、オブジェクト名に『txtOutput』、Captionプロパティに『0』と入力します。



**TextBox①のプロパティ設定**  
 ○オブジェクト名 :txtInput  
 ○Text :0  
 ○font :任意

**TextBox②のプロパティ設定**  
 ○オブジェクト名 :txtOutput  
 ○Text :0  
 ○font :任意

TextBoxコントロールをクリックし、選択状態にします。

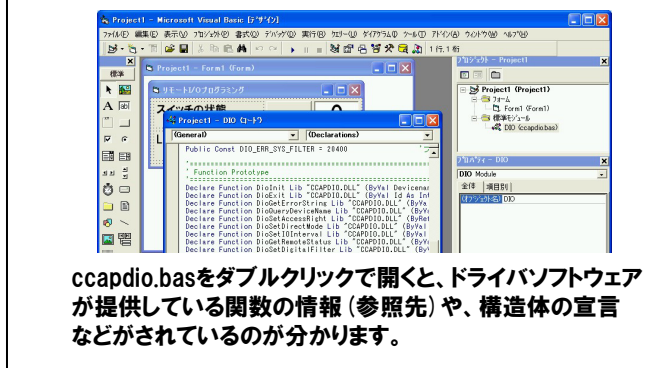
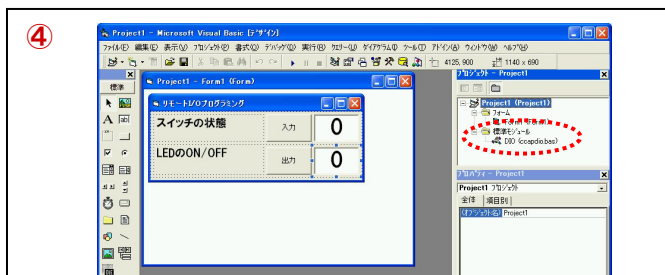
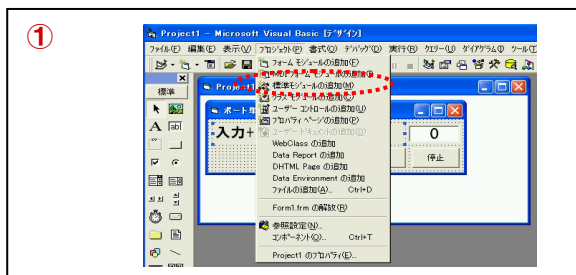
### 6-3-18.標準モジュールファイルの追加

Visual BasicでAPI-CAP (W32) を使用してプログラムを作成するには、API-CAP (W32) が提供するAPI関数を呼び出すための宣言が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。

- ① Visual Basicのメニュー『プロジェクト』 - 『標準モジュールの追加』を選択します。
- ② ダイアログが表示されるので、『既存のファイル』のタブをクリックします。
- ③ 標準設定でインストールした場合には下記の場所に標準モジュールファイルがあります。本書では、デジタル入出力の機能を使用しますので『ccapdio.bas』を選択して『開く』ボタンをクリックします。

**C:¥Program Files¥CONTEC¥API-CAP (W32) ¥Samples¥Inc¥ccapdio.bas**

- ④ プロジェクトエクスプローラ上に標準モジュールファイルが追加されていることを確認してください。

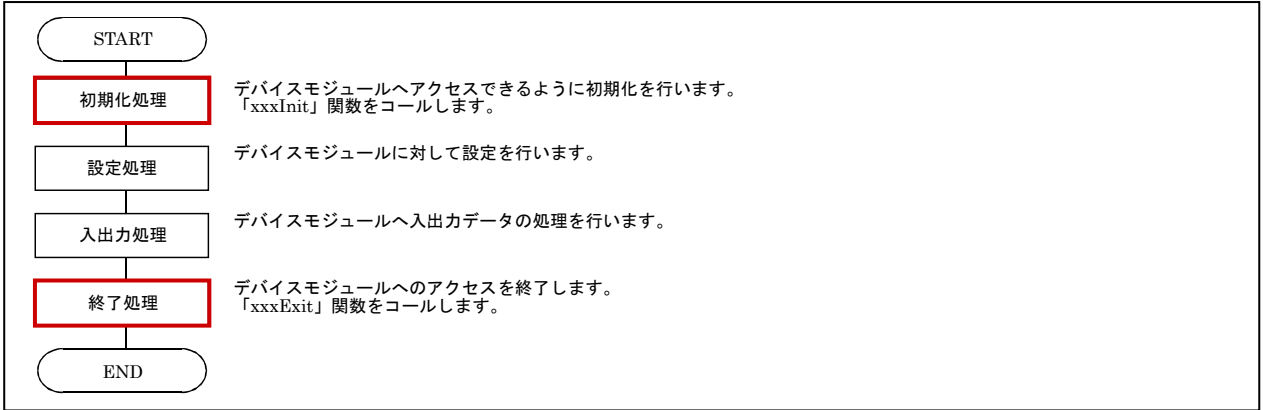




## 6-3-19.API-CAP (W32) の処理体系

API-CAP (W32) は、初期化処理ではじまり、終了処理で終了する決まりがあります。初期化処理および、終了処理の専用関数を用意していますので、それぞれのタイミングでその関数を実行します。

API-CAP (W32) のライブラリは、デジタル、アナログ、カウンタ、温度計測といった、I/Oコントローラユニットに接続して、使用するデバイスのカテゴリごとに特化した関数インターフェイスを持つDLLの集まりです。API-CAP (W32) のライブラリ関数を使用すれば、仮想アドレスマップやプロトコル、デバイス固有のコントロールシーケンスなどを意識せずに、カテゴリごとに用意された機能別関数を呼び出すだけで、I/Oコントローラユニットに接続されたデバイスを簡単に制御できます。



## 6-3-20.変数の宣言

プロジェクトエクスプローラの『コードの表示』ボタンをクリックして、コードウィンドウを開きます。本プロジェクトにて使用する変数を宣言します。下記のコードを記述してください。変数の型宣言に関しては、使用する関数の仕様により決定します。

- 『コードの表示』ボタンをクリックします。
- コードウィンドウが開きます。
- 変数を宣言(記述)します。

Dim Diold	As Integer	デバイスId格納用変数
Dim Ret	As Long	戻り値用変数
Dim InBit	As Integer	入力ビット指定用変数
Dim InpData	As Byte	入力データ格納用変数
Dim OutBit	As Integer	出力ビット指定用変数
Dim OutData	As Byte	出力データ格納用変数
Dim DeviceName	As String	デバイス名設定用変数

## 6-3-21.初期化処理の記述

API-CAP (W32) は初期化処理ではじまり、終了処理で終了する決まりがあります。初期化処理は、『Form』の『Form\_Loadイベント』内で、初期化処理関数を実行します。『Form\_Loadイベント』は『Form』がロード (立ち上がる) 際に発生するイベントで、各コントロールの既定値の設定や、変数を初期化する際に使われます。

① コードウィンドウを開き、『オブジェクト』から『Form』を選択します。

② 『プロシージャ』から『Load』を選択します。

③ Private Sub Form\_Load () から、End Subの間に初期化処理関数を記述します。

```
DeviceName = "DIO080000"
Ret = Diolnit (DeviceName, Diold)
```

‘デバイス名を変数に格納  
‘デバイス名のIDをDioldに格納

## 初期化処理関数『Diolnit』リファレンス

■機能 デバイスIDを取得して、デバイスをアクセス可能にします。以降の関数では、このデバイスIDを使用して、デバイスにアクセスします。複数のデバイスをアクセスする場合、IDを格納する変数を複数用意してください。Diolnitが正常終了した後、DioExitが呼び出されるまで、各機能関数が使用できます。

■書式 Visual Basic6.0の場合

```
Dim Diold As Integer
Dim DeviceName As String
Ret = Diolnit (DeviceName, Diold)
```

■引数 DeviceName : 設定ユーティリティで設定したデバイス名を指定します。デバイス名は最大50文字です。

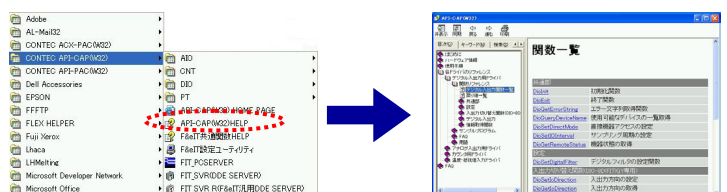
Diold : 取得したデバイスIDを格納する変数を指定します。以降の関数では、このデバイスIDを使用して、デバイスにアクセスします。

Ret : 終了情報 (戻り値) → 正常終了:0, エラー終了: 0以外 (詳細はヘルプ参照)。

※ 本書では、戻り値の確認 (エラー処理) は、誌面の都合上、割愛しております。実システムにおいては、関数を実行した後にエラー処理のコードを記述します。エラー処理の方法は、各サンプルプログラムを参照してください。

オンラインヘルプファイルは使用手順をはじめ、API-CAP (W32) が提供する各関数のリファレンスなどが、記載されています。関数の使い方は [各ドライバのリファレンス] - [関数リファレンス] に記載されています。この関数リファレンスとサンプルプログラムのコードを参照しながらプログラミングを進めていきます。

『スタート』 - 『プログラム』 - 『CONTEC API-CAP (W32)』 - 『API-CAP (W32) HELP』の手順で、オンラインヘルプファイルを立ち上げます。



## 6-3-22. 終了処理の記述

API-CAP (W32) は初期化処理ではじまり、終了処理で終了する決まりがあります。終了処理は、『Form』の『Form\_Unloadイベント』内で、終了処理関数を実行します。『Form\_Unloadイベント』は『Form』が画面から消去(アンロード)される際に発生するイベントです。

① コードウィンドウを開き、『オブジェクト』から『Form』を選択します。

② 『プロシージャ』から『Unload』を選択します。

③ Private Sub Form\_Unload () から、End Subの間に終了処理関数を記述します。

Ret = DioExit (Diold) 終了処理実行

## 終了処理関数 『DioExit』 リファレンス

■機能 Diolnit (初期化処理関数) で取得したデバイスIDを無効にし、デバイスに対してアクセス不能にします。この関数を呼び出した後は、Diolnit以外の各機能関数は使用できません。

■書式 Visual Basic6.0の場合

```
Dim Diold As Integer
Ret = DioExit (Diold)
```

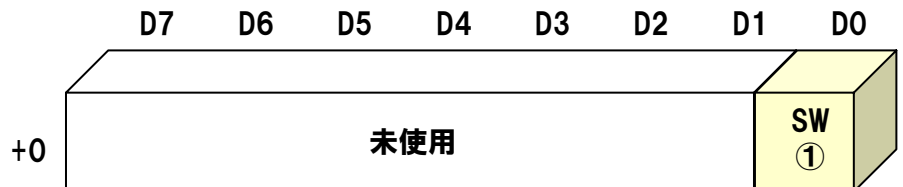
■引数 Diold : Diolnit (初期化処理関数) で取得したデバイスIDを指定します。

Ret : 終了情報(戻り値) → 正常終了:0、エラー終了: 0以外(詳細はヘルプ参照)。

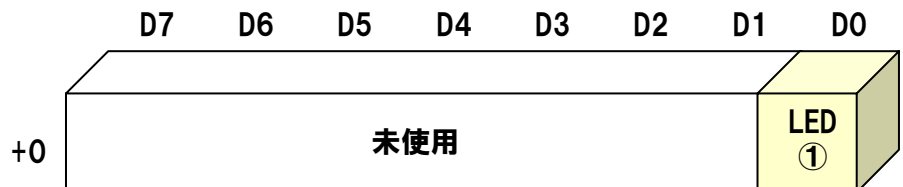
※ 本書では、戻り値の確認(エラー処理)は、誌面の都合上、割愛しております。実システムにおいては、関数を実行した後にエラー処理のコードを記述します。エラー処理の方法は、各サンプルプログラムを参照してください。

## 実習環境の論理ポート・論理ビット マップ図 (DIO-8/8 (FIT) GY)

### 入力ポート

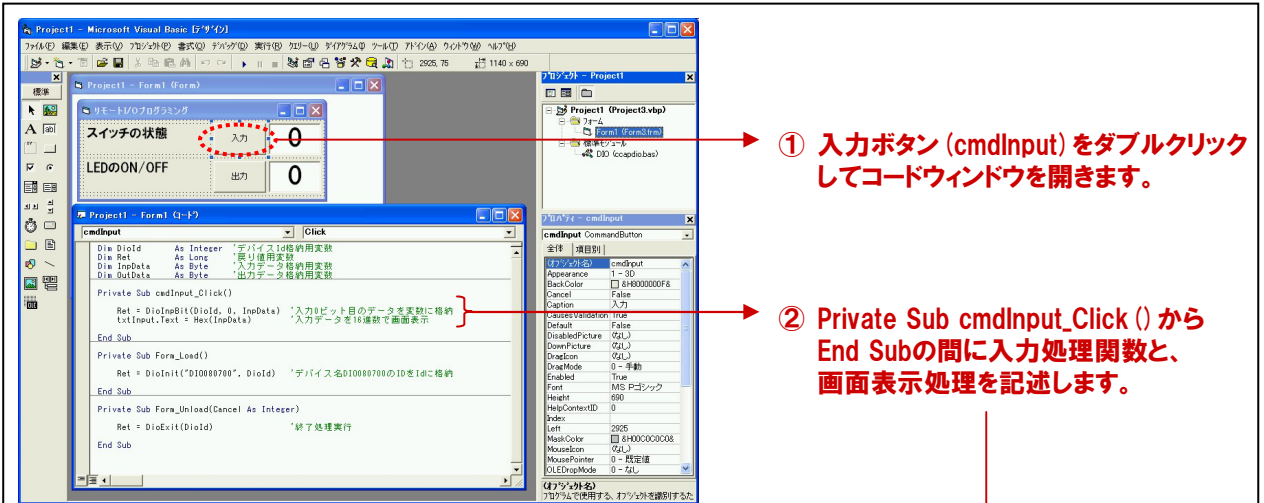


### 出力ポート



## 6-3-23.入力処理の記述

DIO-8/8 (FIT) GYの『入力0ビット (接続されているスイッチ)』からデータの入力を行う処理を追加します。API-CAP (W32) では、ポート単位とビット単位の入力関数が提供されています。今回は、1つのスイッチの状態を入力するだけですので、ビット単位の入力関数を使用します。



InBit = 0

Ret = DiolnBit (Diold, InBit, InpData)

txtInput.Text = Hex (InpData)

‘入力ビットに0を指定

‘入力0ビットのデータを変数に格納

‘入力データを16進数で表示 (オブジェクト=txtInput)

## ビット単位の入力処理関数 『DiolnBit』 リファレンス

■機能 指定した論理ビットからデジタルデータの入力を行います。ビットデータは0 (OFF) または1 (ON) です。

■書式 Visual Basic6.0の場合

```
Dim Diold As Integer
Dim InBit As Integer
Dim InpData As Byte

Ret = DiolnBit (Diold, InBit, InpData)
```

■引数 Diold : Diolnit (初期化処理) で取得したデバイスIDを指定します。

InBit : 入力論理ビット番号を指定します。入力論理ビット番号は、デバイスの入力ビットの先頭を0として最後の入力ビットまで、連続で付けられた番号です。

InpData : 入力データを格納する変数を指定します。

Ret : 終了情報(戻り値) → 正常終了:0, エラー終了: 0以外 (詳細はヘルプ参照)。

※ 本書では、戻り値の確認 (エラー処理) は、誌面の都合上、割愛しております。実システムにおいては、関数を実行した後にエラー処理のコードを記述します。エラー処理の方法は、各サンプルプログラムを参照してください。

## 6-3-24.出力処理の記述

DIO-8/8 (FIT) GYの『出力0ビット (接続されているLED)』ヘデータの出力を行う処理を追加します。  
API-CAP (W32) では、ポート単位とビット単位の出力関数が提供されています。今回は、1つのLEDのON/OFF制御を行うだけですので、ビット単位の出力関数を使用します。

① 出力ボタン (cmdOutput) をダブルクリックしてコードウィンドウを開きます。

② Private Sub cmdOutput\_Click () から End Subの間にデータ設定処理と、データ出力処理を記述します。

```
OutData = Val (txtOutput.Text)      'テキストボックスのデータを数値変換、変数へ格納
OutBit = 0                          '出力ビットに0を指定
Ret = DioOutBit (Diold, OutBit, OutData) 'データを出力0ビットへ出力
```

## ビット単位の出力処理関数 『DioOutBit』 リファレンス

■機能 指定した論理ビットヘデジタルデータの出力を行います。データは0 (OFF) または1 (ON) です。

■書式 Visual Basic6.0の場合

```
Dim Diold As Integer
Dim OutBit As Integer
Dim OutData As Byte

Ret = DioOutBit (Diold, OutBit, OutData)
```

■引数

- Diold : Diolnit (初期化処理) で取得したデバイスIDを指定します。
- OutBit : 出力論理ビット番号を指定します。出力論理ビット番号は、デバイスの出力ビットの先頭を0として最後の出力ビットまで、連続で付けられた番号です。
- OutData : 出力するデータを指定します。データは0 (OFF) または1 (ON) です。
- Ret : 終了情報(戻り値) → 正常終了:0, エラー終了: 0以外 (詳細はヘルプ参照)。

※ 本書では、戻り値の確認(エラー処理)は、誌面の都合上、割愛しております。実システムにおいては、関数を実行した後にエラー処理のコードを記述します。エラー処理の方法は、各サンプルプログラムを参照してください。

## TOPICS

### 『Val関数 (Visual Basic標準関数)』

指定された文字列中の数値として解釈できるものを数値データに変換するVisual Basicの関数です。今回の例では、テキストボックスに入力されたデータを数値の変換する役割をしています。



## 6-4. デジタル入出力カード (PIO-16/16L (CB) H) とのプログラム比較

今回作成したリモートI/Oでのデジタル入出力と、第5章で作成したPIO-16/16 (CB) Hを使用したプログラムを比較してみましょう。F&EITシリーズのイーサネットリモートI/O機器とAPI-CAP (W32) を使用すれば、ネットワーク接続やプロトコルを意識せず、同様の感覚でリモートI/Oシステムが実現できることが分かります。

### ① F&EIT機器とAPI-CAP (W32) を使用したビット入出力プログラミング例 (今回のプログラムリスト)

```
Dim Diold As Integer 'デバイスId格納用変数
Dim Ret As Long '戻り値用変数
Dim InBit As Integer '入力ビット指定用変数
Dim InpData As Byte '入力データ格納用変数
Dim OutBit As Integer '出力ビット指定用変数
Dim OutData As Byte '出力データ格納用変数
Dim DeviceName As String 'デバイス名設定用変数

Private Sub Form_Load ()
    DeviceName = "DIO080000" 'デバイス名を変数に格納
    Ret = DioInit (DeviceName , Diold) '初期化処理
End Sub

Private Sub Form_Unload (Cancel As Integer)
    Ret = DioExit (Diold) '終了処理実行
End Sub

Private Sub cmdInput_Click ()
    InBit = 0 '入力ビットに0を指定
    Ret = DioInpBit (Diold, InBit , InpData) '入力0ビットのデータを変数に格納
    txtInput.Text = Hex (InpData) '入力データを16進数で画面表示
End Sub

Private Sub cmdOutput_Click ()
    OutData = Val (txtOutput.Text) 'テキストボックスのデータを数値変換、変数へ格納
    OutBit = 0 '出力ビットに0を指定
    Ret = DioOutBit (Diold, OutBit , OutData) 'データを出力0ビットへ出力
End Sub
```

### ② PIO-16/16 (CB) HとAPI-PAC (W32) を使用したビット入出力プログラミング例 (第5章: タイマ処理部省略)

```
Dim hDrv As Long 'デバイスハンドル格納用変数
Dim Ret As Long 'リターンコード (戻り値) 格納用変数
Dim InBit As Integer '入力ビット指定用変数
Dim InBitData As Byte '入力データ格納用変数
Dim OutBit As Integer '出力ビット指定用変数
Dim DeviceName As String 'デバイス名格納用変数

Private Sub Form_Load ()
    DeviceName="DIO00" 'デバイス名を変数に格納
    Ret = DioOpenEx (DevieName,hDrv) '初期化処理 (デバイスハンドルの取得)
End Sub

Private Sub Form_Unload (Cancel As Integer)
    Ret = DioClose (hDrv) '終了処理 (デバイスハンドルを開放)
End Sub

Private Sub tmrTimer_Timer ()
    InBit = 0 '入力ビットに0を指定
    Ret = DioInpBit (hDrv, InBit, InBitData) '入力0ビットのデータを変数InBitDataに格納

    OutBit = 1 '出力ビットに1を指定
    Ret = DioOutBit (hDrv, OutBit , InBitData) '出力1ビットにInBitDataを出力

    lblData.Caption = Hex (InBitData) 'lblDataに入力データ (InBitData) を16進数で表示
End Sub
```



## 6-5.API-CAP (W32) デジタル入出力用ドライバ 提供関数一覧

イーサネットベースのリモートI/O用ドライバソフトウェア『API-CAP (W32)』は、ネットワークを意識させないプログラミングを実現しています。本書にて使用したデジタル入出力用ドライバには、弊社製デジタル入出力リモートI/O機器を簡単に制御可能な、さまざまな関数を用意しています。それぞれの関数は、処理ごとに分かりやすく分類されており、また処理の内容が一目で分かるような名称となっています。これらの関数を使用すれば、さらに高度な処理も可能です。

### ① 共通部

DioInit	初期化関数
DioExit	終了関数
DioGetErrorString	エラー文字列取得関数
DioQueryDeviceName	使用可能なデバイスの一覧取得
DioSetDirectMode	直接機器アクセスの設定
DioSetIOInterval	サンプリング周期の設定
DioGetRemoteStatus	機器状態の取得

### ② 設定

DioSetDigitalFilter	デジタルフィルタの設定関数
---------------------	---------------

### ③ 入出力切り替え関数 (DIO-8D (FIT) GY専用)

DioSetIoDirection	入出力方向の設定
DioGetIoDirection	入出力方向の取得

### ④ デジタル入出力

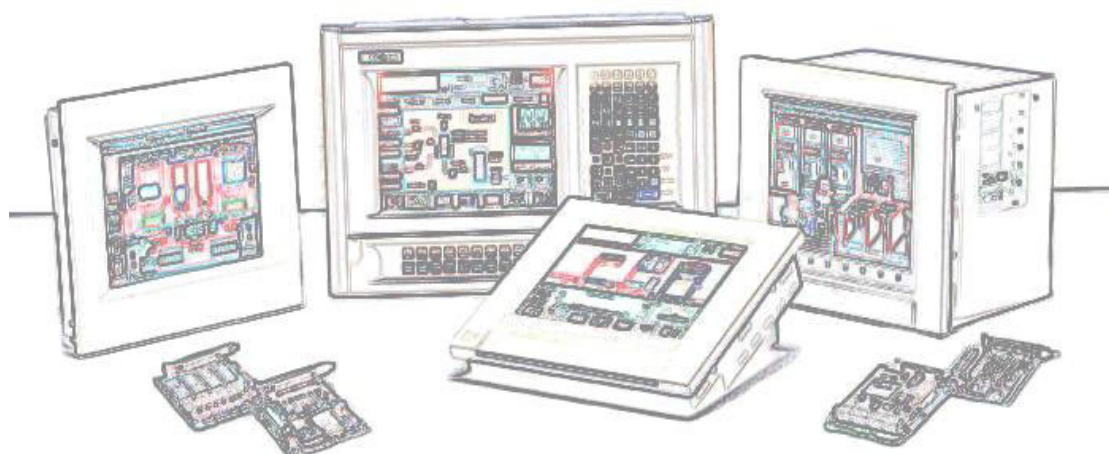
DioInpByte	1バイト入力関数
DioInpBit	1ビット入力関数
DioOutByte	1バイト出力関数
DioOutBit	1ビット出力関数
DioEchoBackByte	1バイトエコーバック関数
DioEchoBackBit	1ビットエコーバック関数
DioInpMultiByte	複数バイト入力関数
DioInpMultiBit	複数ビット入力関数
DioOutMultiByte	複数バイト出力関数
DioOutMultiBit	複数ビット出力関数
DioEchoBackMultiByte	複数バイトエコーバック関数
DioEchoBackMultiBit	複数ビットエコーバック関数

### ⑤ 情報取得関数

DioGetMaxPorts	使用可能な入出力ポート数取得関数
DioGetMaxBits	使用可能な入出力ビット数取得関数

# 付録

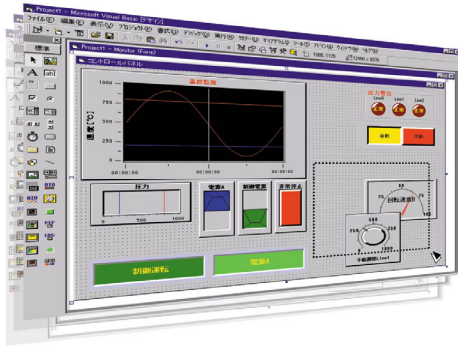
## 各種開発ツール紹介



# ①計測システム開発用ActiveXコンポーネント集【ACX-PAC (W32) Ver.4.0】

計測・制御システムの開発を強力にサポート！

for Windows



本製品は、200種類以上の弊社計測制御用インターフェイスボード/カード/USBモジュールに対応した計測システム開発支援ツールです。計測用途に特化したソフトウェア部品集で画面表示（各種グラフ、スライダ他）、解析・演算（FFT、フィルタ他）、ファイル操作（データ保存、読み込み）などのActiveXコンポーネントを多数収録しています。

アプリケーションプログラムの作成は、ソフトウェア部品を貼り付けて、関連をスクリプトで記述する開発スタイルで、効率よく短期間でできます。また、データロガーや波形解析ツールなどの実例集（アプリケーションプログラム）が収録されていますので、プログラム作成なしでパソコン計測がすぐに始められます。『実例集』はソースコード（Visual Basic）付きですので、お客様によるカスタマイズも可能です。

画面作成用コンポーネント	
X-Yグラフ	1次元、または2次元配列データのグラフを同時に32ライン表示。マウスによる拡大・縮小・カーソル・移動やサブ軸などの機能
棒グラフ	ヒストグラム表示に最適な棒グラフを表示。
トレンドグラフ	スクロールしていくグラフを8ラインまで表示。モニタリングに最適。
ランプ	デジタル入力状態の表示に最適なランプを表示。
スイッチ	デジタル出力や様々な設定のON/OFF等の表示に最適なスイッチ。
アナログメータ	取得データをアナログメータで表示。
レベルメータ	取得データをレベルメータで表示。
スライダ	データ設定に最適なスライダスイッチ。
ボリューム	データ設定に最適なボリュームスイッチ。
ファイル操作用コンポーネント	
ロギング	配列データをファイルに保存。ファイル名に日付や番号を付加して、複数のファイルに自動的に保存が可能。
リプレイ	ファイルからデータを読み込んで配列に格納。

ボード制御用コンポーネント	
アナログ入出力	弊社製アナログ入出力ボード/カード/USBモジュール制御用。
デジタル入出力	弊社製デジタル入出力ボード/カード/USBモジュール制御用。
GPIB通信	弊社製 GPIB通信ボード/カード/USBモジュール制御用。
カウンタ入力	弊社製カウンタ入力ボード/カード/USBモジュール制御用。
プロトコル変換用コンポーネント	
プロトコル変換	文字列データから必要な部分のみを数値データへ変換。
演算・解析用コンポーネント	
キャリブレーション	JIS規格に対応した熱電対のキャリブレーション演算が可能。
デジタルフィルタ	FIRフィルタによる入力データのフィルタリングが可能。
周波数分析	周波数特性を解析するFFT演算用。
統計解析	平均値、最大・最小値、ヒストグラム演算などの統計解析用。

対応日本語OS
<ul style="list-style-type: none"> <li>Microsoft Windows XP Professional</li> <li>Microsoft Windows XP Home Edition</li> <li>Microsoft Windows 2000 Professional</li> <li>Microsoft Windows NT Ver.4.0 (SP3 以上)</li> <li>+Internet Explorer 4.01以上</li> <li>Microsoft Windows Me</li> <li>Microsoft Windows 98およびSecond Edition</li> <li>Microsoft Windows 95 (SP1 以上)</li> <li>+Internet Explorer 4.01以上</li> </ul>
詳しくは、弊社ホームページをご確認ください。

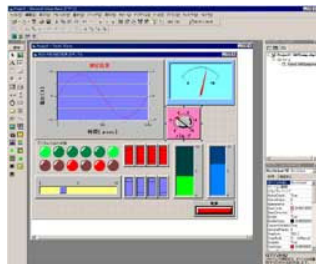
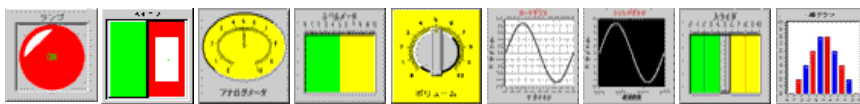
対応開発環境
<ul style="list-style-type: none"> <li>Microsoft Visual Basic .NET 2003, 2002</li> <li>Microsoft Visual Basic Ver.6.0, 5.0</li> <li>Microsoft Visual C++ .NET 2003, 2002</li> <li>Microsoft Visual C# .NET 2003, 2002</li> <li>Microsoft Visual C++ Ver.6.0, 5.0</li> <li>Microsoft Excel 2003 (VBA 6.4)、2002 (VBA 6.3)、2000 (VBA 6.0)、97 (VBA 5.0)</li> <li>Borland Delphi Ver.7, Ver.5, Ver.4</li> <li>National Instruments LabVIEW 7.1, 7.0, 6.1, 6i</li> </ul>
詳しくは、弊社ホームページをご確認ください。

適応パソコン
<ul style="list-style-type: none"> <li>IBM PC/AT 互換機、DOS/V 機</li> </ul>
その他
<ul style="list-style-type: none"> <li>Pentium100MHz以上のCPUを推奨</li> <li>プログラミング言語（コンテナ）が正常に動作する環境</li> </ul>

2006年2月現在

## 開発スタイル

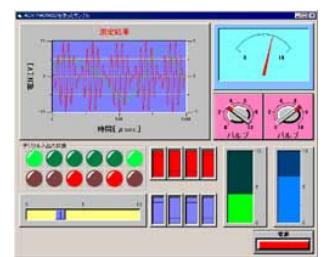
インターフェイスボード制御用の部品やメータ・グラフなどのGUI部品を・・・



Visual Basic やExcel などに貼り付けます。



各種設定はプロパティページにより、プログラムレスで行えます。

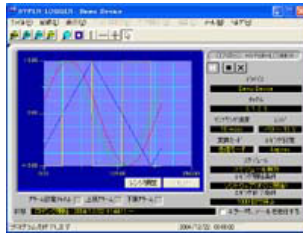


コントロールを動作させるための簡単なメソッドを記述して完成！

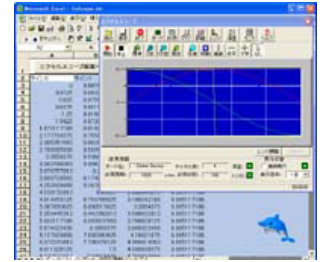
## 充実の実例集 (実用アプリケーション・プログラム)

Visual Basicの完全ソースコード付きでカスタマイズも容易な実例集です。すべての実例は計測/入力値および演算結果をファイルに保存することが可能です。また、アラーム情報のメール送信機能や、マウス操作によるグラフの拡大・縮小・カーソル・サブ軸などが追加され、さらに使いやすくなりました。アナログ出力に対応したプログラムやVisual Basic .NET 2003、2002に対応したソースコード、英語版のVisual Basic 6.0、5.0、Visual Basic .NET 2003、2002にも対応しています。

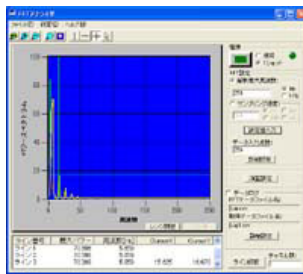
- **ハイパーロガー**  
アナログ信号の高速連続サンプリングとファイリングができます。



- **エクセルスコープ**  
アナログ信号をサンプリングし、フォームウィンドウにグラフ表示するとともにワークシートに書き出します。  
※Microsoft Excel 2003,2002,2000または97対応



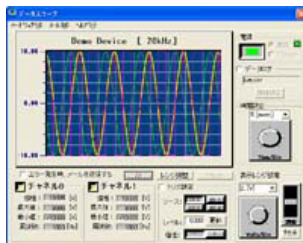
- **FFTアナライザ**  
アナログ信号をサンプリングし、FFT解析を行うことができます。



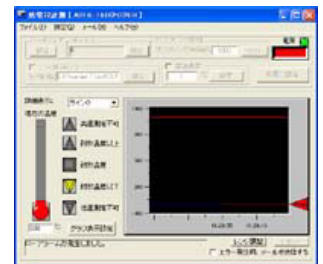
- **ペンレコーダ**  
複数のアナログ信号をサンプリングし、信号をペンレコーダのようにグラフ描画していきます。



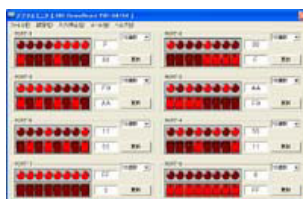
- **データスコープ**  
オシロスコープのように、アナログ信号を高速(有限)サンプリングし、波形表示します。



- **熱電対計測**  
熱電対センサによる温度計測を行い、温度の表示と変異をグラフ表示します。



- **デジタルモニタ**  
デジタル入出力とスイッチ、ランプを組み合わせた表示例です。



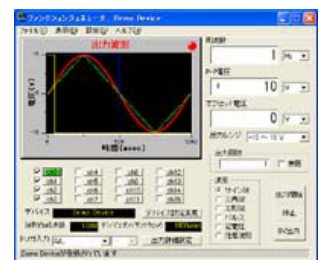
- **カウンタモニタ**  
カウンタ入力および周波数を計測し、表示します。



- **システムモニタ**  
パソコンのCPU使用率、メモリ使用率をヒストグラムで表示します。



- **ファンクションジェネレータ**  
連続アナログ出力(任意波形出力)を行います。



ACX-PAC (W32) の最新情報は、下記URLへアクセスしてください！

<http://www.contec.co.jp/acxpac/>

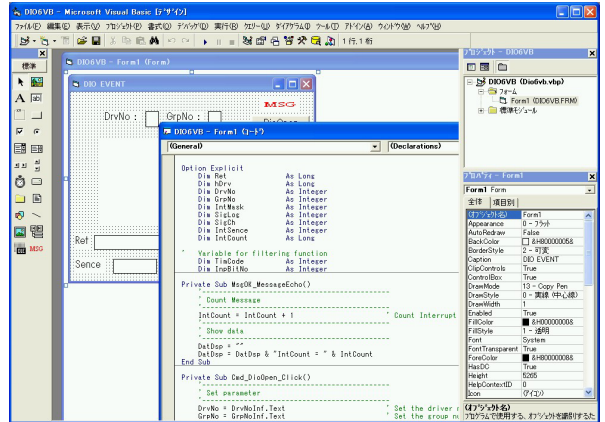


## ②Windows版 ドライブライブラリ【API-TOOL for Windows】

充実のサンプルプログラム、分かりやすいヘルプで開発をサポート！

for Windows

弊社製インターフェイスボード/カードへのコマンドをWindows標準のWin32API関数(DLL)の形式でご提供するライブラリソフトウェアです。Visual BasicやVisual C/C++などのWin32API関数をサポートした各種プログラミング言語で、弊社製インターフェイスボード/カードの特色を活かした高速なアプリケーションが作成できます。最新バージョンは、弊社ホームページの『API-TOOL Developers' Site』からダウンロード可能です。



### 特長

#### ① 統一されたAPI

各シリーズはそれぞれ同種のインターフェイスボード(RS-232C系、アナログボード系、デジタルボード系等)をまとめて一つの統一されたDLLで構成されています。それゆえ、ボードの機種変更に対して、再登録のみでソフトの変更が不要な流用性の高いアプリケーションの制作が可能です。

#### ② イベント駆動でのデータ収集が可能

イベントドリブン型制御が可能な関数をサポートしているため、ユーザー任意のタイミングでのデータ収集が可能です。

#### ③ 論理デバイスでアクセス

煩わしいI/Oポートアドレスを意識しないプログラミングが可能です。

#### ④ 分かりやすい関数名

各APIは処理機能が分かる名称を用いており、読みやすいプログラミングが実現します。

#### ⑤ 診断プログラム

インターフェイスボード/カードとドライバソフトウェアの状態を診断するプログラムが各ドライブライブラリに付属しています。診断プログラムを使用することにより簡単にボード/カードのセットアップやドライバソフトウェアが正常かどうかを確認することができます。

#### ⑥ 充実したサンプルプログラム

各ドライブライブラリにはサポートする各言語に対応したサンプルプログラムを多数付属しています。関数の使い方を確認するだけでなく、ボード/カードの動作を確認できるため、ドライブライブラリを使用したアプリケーションの開発効率が向上します。

カテゴリ	ライブラリ名称(型式)
シリアル通信	API-SIO (98/PC) NT, API-SIO (98/PC) W95
GPiB通信	API-GPiB (98/PC) NT, API-GPiB (98/PC) W95, API-GPLV (W32)
アナログ入出力	API-AIO (WDM), API-AIO (98/PC) NT, API-AIO (98/PC) W95
デジタル入出力	API-DIO (98/PC) NT, API-DIO (98/PC) W95
カウンタ	API-CNT (98/PC) NT, API-CNT (98/PC) W95
モータコントロール	API-SMC (98/PC) NT, API-SMC (98/PC) W95
タイマ	API-TIMER (W32)

#### 対応OS(日本語版/英語版)

- Microsoft Windows XP Professional
- Microsoft Windows XP Home Edition
- Microsoft Windows 2000 Professional
- Microsoft Windows NT 4.0
- Microsoft Windows NT 3.51
- Microsoft Windows Me
- Microsoft Windows 98およびSecond Edition
- Microsoft Windows

詳しくは、弊社ホームページをご確認ください。

#### 対応開発環境

- Visual C++ Ver.6.0, 5.0, 4.x, 2.0
- Borland C++ Ver.5.0, 4.5x
- Visual Basic Ver.6.0, 5.0, 4.0 (32ビットのみ)
- Visual C++ .NET 2003, 2002
- Visual C# .NET 2003, 2002
- Visual Basic .NET 2003, 2002
- Borland C++ Builder 6.0, 5.0
- Borland Delphi 6.0, 4.0, 3.0

詳しくは、弊社ホームページをご確認ください。

2006年2月現在

対応ハードウェアなどの詳細は、下記URLへアクセスしてください！

<http://www.contec.co.jp/apipac/>

### ③Linux版 ドライブライブラリ【API-TOOL for Linux】

充実のサンプルプログラム、分かりやすいヘルプで開発をサポート！

for Linux

弊社製アドオンボード/カードへのコマンドをモジュール形式のデバイスドライバとシェアードライブラリ形式でご提供する、開発・ランタイムともにライセンスフリーのドライバソフトウェアです。

#### 特長

- ① ヘルプファイルにより、プログラム開発を行いながら使用する関数の説明を画面上で見ることができます。
- ② サポートする各言語に対応したサンプルプログラムで、使用する関数の使い方やボードの動作を確認できるため、開発効率が上がります。
- ③ コンフィグレーションにより、実行環境へ移行を容易にする設定ファイルとドライバ起動スクリプト、停止スクリプトを出力できます。
- ④ ドライバに組み込んで実行できるユーザー割り込み処理ソースコードを添付しています。

カテゴリ・型式	主な特長
デジタル入出力ドライバ API-DIO (LNX) ※1	<ul style="list-style-type: none"><li>● モジュール形式のドライバとシェアードライブラリにより、弊社製デジタル入出力ボードを制御するための関数群を提供しています。</li><li>● 入出力、割り込み、タイマによるトリガ監視といった基本的な機能を提供しています。</li><li>● ドライバに組み込んで実行できるユーザー割り込み処理ソースコードが添付されています。</li></ul>
アナログ入出力ドライバ API-AIO (LNX)	<ul style="list-style-type: none"><li>● モジュール形式のドライバとシェアードライブラリにより、弊社製アナログ入出力ボードを制御するための関数群を提供しています。</li><li>● アナログ入出力の基本的な機能を提供しています。</li><li>● 弊社製アナログ入出力ボードの機能の違いを意識しないプログラミングが可能です。</li><li>● 弊社製アナログ入出力ボードへの設定パラメータをデフォルト値で保持。パラメータの設定なしで動作が可能です。</li><li>● 設定プログラムは実行環境へ移行を容易にする設定ファイルとドライバ起動スクリプト、停止スクリプトを出力します。</li></ul>
GPIB通信ドライバ API-GPIB (LNX)	<ul style="list-style-type: none"><li>● モジュール形式のドライバとシェアードライブラリにより、弊社製GPIBボードを制御するための関数群を提供しています。</li><li>● IEEE-488規格に準拠しています。</li><li>● マスタモード、スレーブモードなどの設定をすべてソフトウェアにて簡単に行えます。</li></ul>
カウンタ入力ドライバ API-CNT (LNX)	<ul style="list-style-type: none"><li>● モジュール形式のドライバとシェアードライブラリにより、弊社製カウンタボードを制御するための関数群を提供しています。</li><li>● モード設定、カウント値取得、カウント一致割り込み、タイマ割り込みといった基本的な機能を提供しています。</li></ul>
汎用入出力ドライバ IO-LIB (LNX) ※1	<ul style="list-style-type: none"><li>● 任意の指定I/Oポートアドレスに対し、1 / 2 / 4バイトの単位でアクセスが可能です。</li><li>● PCIバス / CompactPCIバス (Plug and Play対応) ボードのリソース情報取得が可能です。</li><li>● 割り込みイベント処理を行うことができます。</li><li>● コンソールおよびX-Window (kylinx) のサンプルプログラムを付属しています。</li><li>● HTML形式の関数リファレンスを付属しています。</li><li>● ドライバおよびシェアードライブラリのソースコードが付属しています。</li></ul>

対応言語	動作確認済みのカーネル / ディストリビューション
<ul style="list-style-type: none"><li>● gcc</li><li>● kylinx2 ※2</li></ul>	<ul style="list-style-type: none"><li>● 2.4.21 / RedHat Linux Professional Workstation</li><li>● 2.4.20 / RedHat Linux 9</li><li>● 2.4.18 / RedHat Linux 8.0</li><li>● 2.4.18 / RedHat Linux 7.3</li><li>● 2.4.7 / RedHat Linux 7.2</li><li>● 2.4.2 / RedHat Linux 7.1</li><li>● 2.2.16 / RedHat Linux 7.0</li><li>● 2.2.14 / RedHat Linux 6.2</li><li>● 2.4.18 / TurboLinux 8</li><li>● 2.4.5 / TurboLinux 7.0</li><li>● 2.2.13 / TurboLinux 6.0</li></ul>

※1 : API-DIO (LNX) 、 IO-LIB (LNX) は、Kernel2.6.xx、RedHat Enterprise Linux4、Turbo Linux10Server1に対応しております。

※2 : API-AIO (LNX) 、 API-GPIB (LNX) は対応していません。

2006年2月現在

対応ハードウェアなどの詳細は、下記URLへアクセスしてください！

<http://www.contec.co.jp/apipac/>



## ④LabVIEW対応サポートソフトウェア

for LabVIEW

National Instrument社LabVIEWは、計測分野で最も多く使用されているソフトウェアのひとつです。LabVIEWで弊社製アドオンボード／カードを使用する方法として、弊社では以下のサポートソフトウェアの使用を推奨および提供しています。豊富なラインアップと確実な実績を誇る弊社製アドオンボード／カードが使用できるだけでなく、LabVIEWを使用した計測システムをより安価に構築することができます。

●LabVIEWで弊社製デジタル入出力、アナログ入出力、カウンタの各アドオンボード／カードを使用する場合

LabVIEW対応データ集録用VIライブラリ VI-DAQ

●LabVIEWで弊社製 GPIB 通信ボード／カードを使用する場合

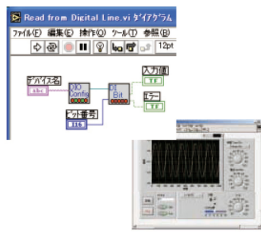
GPIB通信ボード用 LabVIEW対応 GPIBドライバ API-GPLV (W32)

●LabVIEWで弊社製シリアル通信ボード／カードを使用する場合

標準COMドライバ「COM-DRV (W32)」を使用し、標準COMポートにセットアップ

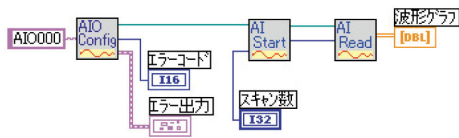
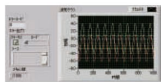
### LabVIEW対応データ集録用VIライブラリ VI-DAQを使用する

VI-DAQ (ブイアイ・ダック) とは、弊社の豊富なアナログ入出力、デジタル入出力、カウンタ入力デバイス (PCIバスボード/PCカード/USBモジュール) を、National Instruments社のLabVIEWで使用するためのVIライブラリです。LabVIEWの『データ集録VI』に似た関数形態で作成されているため、複雑な設定をすることなく、簡単に各種デバイスが使用できます。



#### 分かりやすいインターフェイス

VI-DAQは、LabVIEWの『データ集録VI』に似た関数形態で作成されています。ドライバ形式のような複雑な設定が必要ないため、弊社デバイスを使用したシステムが短期間で作成できます。



#### 用途と段階に応じたVI関数を提供

よく利用する機能を基本VI、特殊な条件設定などを拡張VIとして提供しています。まず基本VIで使い方を理解し、その後、拡張VIを必要に応じて追加していくなど、用途と段階に応じた使い分けが可能です。

#### 用途に応じた実用的なサンプルを提供

VIを利用した、シンプルで分かりやすい構成のサンプルを多数ご用意しています。例えば、アナログ入力では、「簡単な入力」「トリガの使用」「連続サンプリング」のように、用途に応じたサンプルをご用意しております。実際の動作を確認しながら、必要な個所だけを変更するなど、VI-DAQを使用したシステム開発を強力にサポートします。

#### ハードウェアに依存しないライブラリ

PCIバスボード/PCカード/USBモジュールといった異なるデバイスを使用する際も、同じVIで使用することができます。

#### 英語環境に対応

英語環境に対応したVIライブラリ/ヘルプ/サンプルをご用意。海外向けシステムの開発も可能です。

#### ■対応OS [日本語/英語版]

Windows XP、Windows 2000、Windows Me/98SE/98

#### ■LabVIEW対応バージョン

National Instruments LabVIEW 7.1 / 7.0 / 6.1 / 6i

LabVIEW対応データ集録用VIライブラリVI-DAQの詳細は、

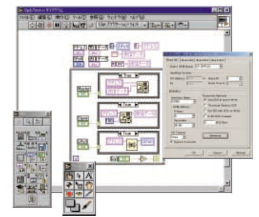
<http://www.contec.co.jp/vidaq>

LabVIEW対応GPIBドライバ API-GPLV (W32)の無償ダウンロードと詳細は、

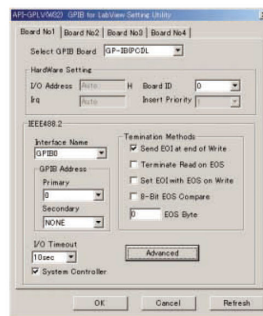
<http://www.contec.co.jp/gplv/>

### GPIB通信ボード用 LabVIEW対応 GPIBドライバ API-GPLV (W32)を使用する

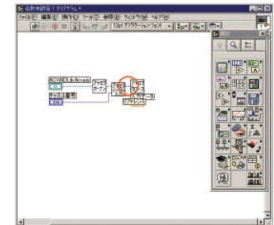
National Instruments社のLabVIEWで弊社製GPIB通信ボード／カードを使用するためのドライバソフトウェアです。本ソフトウェアをインストールすることにより、弊社製GPIB通信ボード／カードを使用したLabVIEWでのプログラム開発や、完成したGPIB通信プログラムを弊社製GPIB通信ボード／カードを使用して動作させることが可能です。また、National Instruments社のAPI関数スタイルで作成されているため、Microsoft Visual Basicをはじめとする他のプログラム言語で使用することもできます。



#### 設定ユーティリティと診断ユーティリティの提供



#### LabVIEW GPIB機器ライブラリの使用



LabVIEW提供のGPIB機器ライブラリがそのままご使用頂けます。

ハードウェアおよびパラメータ (IEEE488.2) をユーティリティを使用して簡単に設定できます。また、設定後に簡易動作確認を行うための診断ユーティリティが付属しています。

#### ■対応OS [日本語版/英語版]

Windows XP Professional、Windows XP Home Edition、Windows Server 2003  
Windows 2000 Professional、Windows NTWorkstation 4.0 + SP3以降、Windows Me/98SE/98/95

#### ■対応言語

**National Instruments** LabVIEW 7.1 / 7.0 / 6.1 / 6i / 5.1 / 5.0  
**Microsoft** Visual Basic 6.0 / 5.0 / 4.0、  
Visual C++ 6.0 / 5.0 / 4.x / 2.0  
Visual Basic .NET 2003 / 2002  
Visual C++ .NET 2003 / 2002  
Visual C# .NET 2003 / 2002  
**Borland** C++ Builder 6.0 / 5.0、Delphi 6.0 / 5.0 / 4.0

#### ■対象ボード

##### CompactPCIバスボード

GP-IB (CPCI) F\*1

##### PCIバスボード

GP-IB (LPCI) F\*1、GP-IB (LPCI) FL\*1、GP-IB (PCI)、  
GP-IB (PCI) F\*1、GP-IB (PCI) FL\*1、GP-IB (PCI) L

##### ISAバスボード

GP-IB (PC) L

##### PCカード

GP-IB (CB) F\*1、GP-IB (CB) FL\*1、GP-IB (PM)

※1:Windows 98、Windows 2000以降のOSで使用できます。

※National Instruments社のGPIB通信ボードドライバとの併用はできません。

※弊社GPIB通信ボードカード用ドライバライブラリ API-GPIB (98/PC)との互換性はありません。

※同一ハードウェアで、本ソフトウェアを使ったプログラムと、API-GPIB (98/PC)を使ったプログラムを同時に動作させることはできません。

## ⑤ MATLAB対応データ収録用ライブラリ【ML-DAQ】

for MATLAB

ML-DAQとは、The MathWorks 社のMATLABで弊社製アナログ入出力ボードを使用するためのライブラリソフトウェアです。各機能は、MATLABのData Acquisition Toolboxで統一されたインターフェイスに合わせて提供されます。



### 主な特長

#### ① MATLABの標準的なインターフェイスに対応

国内のみならず、ワールドワイドで使用されている MATLABで、弊社製アナログ入出力ボードを使用可能にします。MATLABの標準的なインターフェイスで使用できるので、MATLABユーザーにとって親和性に優れている他、他社ボードからの置き換えも容易です。

#### ② アナログ入出力ボードの各機能に対応

アナログ入出力ボードのアナログ入出力機能、デジタル入出力機能のそれぞれに対応しています。

#### ③ MATLABから直接データ収録

MATLABから直接生の測定データにアクセスできます。測定したデータは MATLABの強力な解析機能を使用することができます。

#### ④ 高機能なデータ収録

単純入出力ができるほか、各種トリガを使用した周期的な計測をすることができます。また「データ収集の終了」などの各種条件でイベントを発生させることができます。

動作環境	
対応OS	<ul style="list-style-type: none"><li>● Microsoft Windows XP Professional</li><li>● Microsoft Windows XP Home Edition</li><li>● Microsoft Windows 2000 Professional</li></ul>
MATLAB 対応バージョン	<ul style="list-style-type: none"><li>● MATLAB R14 以上</li><li>● Data Acquisition Toolbox 2.5 以上</li></ul>
Windows版 高機能アナログ入出力ドライバ API-AIO (WDM) 対応バージョン	<ul style="list-style-type: none"><li>● API-AIO (WDM) Ver.1.70 以上</li></ul>

2006年2月現在

対応ハードウェアなどの詳細は、下記URLへアクセスしてください！  
<http://www.contec.co.jp/mldaq/>

## お問い合わせ先

### ■本書の内容に関するお問い合わせ

デバイス&コンポーネント事業部 総合サービス部 プロモーション課  
E-mail: promote@contec.co.jp TEL: 03-5628-0255

### ■技術的なお問い合わせ

総合インフォメーション テクニカルサポートセンター  
E-mail: tsc@contec.co.jp  
FAX: 03-5628-9344  
TEL: 03-5628-9286 (弊社営業日9:30~12:00、13:00~17:00)

## デジタル入出力 ビギナーズ・ガイドブック Visual Basic6.0編 Ver.1

発行

株式会社コンテック  
〒555-0025 大阪市西淀川区姫里3-9-31

本書の著作権は、株式会社コンテックにあります。  
本書の内容の一部または全部を無断で転載、複写、  
電子化することはできません。  
無断転載・複製はかたくお断りします。  
製品の仕様、その他の記載事項については、予告なく  
変更する場合がありますのでご了承ください。

